Saarland University
Faculty of Natural Sciences and Technology I
Department of Computer Science

Master’s Thesis

Model Checking of Oscillatory and Noisy Periodic Behavior in Markovian Population Models

submitted by
David Spieler

submitted on
2009-12-03

Supervisor
Prof. Dr.-Ing. Holger Hermanns

Advisor
Dr. Verena Wolf

Reviewers
Prof. Dr.-Ing. Holger Hermanns
Dr. Verena Wolf
Eidesstattliche Erklärung
Ich erkläre hiermit an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement in Lieu of an Oath
I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Einverständniserklärung
Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent
I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, 

(Datum / Date) 

(Unterschrift / Signature)
During the time of writing this thesis I have been supported by several people whom I would like to thank sincerely.

Holger Hermanns has been mentoring me since my Bachelor’s studies and it has always been a pleasure to discuss various ideas with him. I also want to thank Verena Wolf who offered the lecture ”Stochastic Dynamics in Systems Biology” which sparked my interest in the topic of this work. Also, I really enjoyed the time writing this thesis under their guidance.

Special thanks goes to Silke Jansen who supported me throughout the whole time and always reminded me of not taking everything too serious.

Last but not least, I would like to thank Christa Schäfer for her great organizational backup.
# Contents

<table>
<thead>
<tr>
<th>Table of Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>1 Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Contributions</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Organization</td>
<td>2</td>
</tr>
<tr>
<td><strong>2 Preliminaries</strong></td>
<td>4</td>
</tr>
<tr>
<td>2.1 Continuous Time Markov Chains</td>
<td>4</td>
</tr>
<tr>
<td>2.1.1 F Bisimulation</td>
<td>6</td>
</tr>
<tr>
<td>2.1.2 Continuous Stochastic Logic</td>
<td>7</td>
</tr>
<tr>
<td>2.2 Markovian Population Models</td>
<td>8</td>
</tr>
<tr>
<td>2.2.1 Continuous Stochastic Logic with Comparisons</td>
<td>9</td>
</tr>
<tr>
<td>2.2.2 F,μ Bisimulation</td>
<td>9</td>
</tr>
<tr>
<td>2.2.3 Markovian Population Models of Biological Reaction Networks</td>
<td>11</td>
</tr>
<tr>
<td><strong>3 Defining and Detecting Oscillatory and Periodic Behavior</strong></td>
<td>13</td>
</tr>
<tr>
<td>3.1 Continuous Deterministic Solutions of Biological Systems</td>
<td>14</td>
</tr>
<tr>
<td>3.2 Fourier Transform</td>
<td>17</td>
</tr>
<tr>
<td><strong>4 Mathematical Definitions of Oscillatory, Periodic and Noisy Periodic Behavior</strong></td>
<td>22</td>
</tr>
<tr>
<td>4.1 Oscillatory Behavior</td>
<td>22</td>
</tr>
<tr>
<td>4.2 Periodic Behavior</td>
<td>23</td>
</tr>
<tr>
<td>4.3 Noisy Periodic Behavior</td>
<td>23</td>
</tr>
<tr>
<td><strong>5 Logical Characterizations</strong></td>
<td>27</td>
</tr>
<tr>
<td>5.1 Logical Characterization of Oscillation</td>
<td>27</td>
</tr>
<tr>
<td>5.2 Logical Characterization of Periodicity</td>
<td>29</td>
</tr>
<tr>
<td>5.3 Logical Characterization of Noisy Periodicity</td>
<td>29</td>
</tr>
<tr>
<td><strong>6 Model Checking and Optimizations</strong></td>
<td>31</td>
</tr>
<tr>
<td>6.1 Optimizing Model Checking of Oscillatory Behavior</td>
<td>31</td>
</tr>
<tr>
<td>6.1.1 Delta Observation MPM Construction</td>
<td>32</td>
</tr>
<tr>
<td>6.1.2 Model Checking Complexity</td>
<td>34</td>
</tr>
<tr>
<td>6.1.3 Correctness</td>
<td>35</td>
</tr>
<tr>
<td>6.2 Further Improvements on Model Checking Oscillations</td>
<td>37</td>
</tr>
<tr>
<td>6.2.1 MPM Predicate Expansion</td>
<td>38</td>
</tr>
<tr>
<td>6.2.2 Model Checking Complexity</td>
<td>39</td>
</tr>
<tr>
<td>6.2.3 Correctness</td>
<td>39</td>
</tr>
<tr>
<td>6.3 Model Checking Noisy Periodicity</td>
<td>39</td>
</tr>
<tr>
<td>6.3.1 Period Detector Expansion</td>
<td>40</td>
</tr>
<tr>
<td>6.3.2 Model Checking Complexity</td>
<td>43</td>
</tr>
<tr>
<td>6.3.3 Correctness</td>
<td>43</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>6.4 Quantifying the Period Length in Noisy Periodic Systems</td>
<td>46</td>
</tr>
<tr>
<td>7 Tool: BioToPrism</td>
<td>53</td>
</tr>
<tr>
<td>7.1 Input File Format</td>
<td>54</td>
</tr>
<tr>
<td>7.2 Prism Model Generation and Model Checking</td>
<td>55</td>
</tr>
<tr>
<td>7.2.1 Basic Model Generation</td>
<td>56</td>
</tr>
<tr>
<td>7.2.2 Change Detector Expansion</td>
<td>58</td>
</tr>
<tr>
<td>7.2.3 Period Detector Expansion</td>
<td>58</td>
</tr>
<tr>
<td>8 Case Studies</td>
<td>63</td>
</tr>
<tr>
<td>8.1 3-way Oscillator</td>
<td>63</td>
</tr>
<tr>
<td>8.1.1 3-way Oscillator without doping</td>
<td>63</td>
</tr>
<tr>
<td>8.1.2 3-way Oscillator with doping</td>
<td>64</td>
</tr>
<tr>
<td>8.2 Repressilator</td>
<td>68</td>
</tr>
<tr>
<td>9 Conclusions</td>
<td>73</td>
</tr>
<tr>
<td>9.1 Future Work</td>
<td>73</td>
</tr>
<tr>
<td>A Matlab Routines</td>
<td>75</td>
</tr>
<tr>
<td>A.1 3-way Oscillator ODE</td>
<td>75</td>
</tr>
<tr>
<td>B Proofs</td>
<td>76</td>
</tr>
<tr>
<td>B.1 Proof of Proposition 3</td>
<td>76</td>
</tr>
<tr>
<td>B.2 Proof of Lemma 1</td>
<td>78</td>
</tr>
<tr>
<td>B.3 Proof of Lemma 2</td>
<td>80</td>
</tr>
<tr>
<td>B.4 Proof of Lemma 3</td>
<td>82</td>
</tr>
<tr>
<td>B.5 Proof of Proposition 6</td>
<td>83</td>
</tr>
<tr>
<td>B.6 Proof of Lemma 4</td>
<td>88</td>
</tr>
<tr>
<td>C XML descriptions</td>
<td>91</td>
</tr>
<tr>
<td>C.1 3-way Oscillator without doping</td>
<td>91</td>
</tr>
<tr>
<td>C.2 3-way Oscillator with doping</td>
<td>92</td>
</tr>
<tr>
<td>C.3 Repressilator</td>
<td>94</td>
</tr>
<tr>
<td>List of Formulas</td>
<td>97</td>
</tr>
<tr>
<td>List of Tables</td>
<td>98</td>
</tr>
<tr>
<td>List of Figures</td>
<td>99</td>
</tr>
<tr>
<td>References</td>
<td>101</td>
</tr>
</tbody>
</table>
1 Introduction

Oscillation is a prevalent phenomenon that can be observed within biological systems at all kinds of granularity, e.g. on a microscopic level within individual cells [MM09] as well as on a macroscopic level within the growth of Savannah patches [MWWM07]. An example for oscillatory behavior is the day/night rhythms of many living organisms, with a period length, i.e. the time needed for one cycle, of approximately 24 hours. Oscillatory biological networks are the underlying structure of rhythmic behavior on a cellular level. In the case of the day/night cycle, the primary mechanism is based on circadian clocks [BL00].

It is the task of systems biology, an biology-based inter-disciplinary field, to study those complex biological systems, in order to understand their underlying basic mechanisms. In the systems biology research cycle, the system under consideration is reduced to a formal model, which is then analyzed for emergent behavior. Traditionally, those models have been sets of ordinary differential equations (ODEs) used to describe the kinetics of the system’s chemical reaction network. Those ODEs have been used to retrieve continuous deterministic solutions, i.e. the concentrations of the involved chemical species over time.

But it has been shown recently, that for some classes of systems, a deterministic formalization is not appropriate. One example is the lambda phage decision circuit, where lambda phages, when infecting E.coli bacteria, either enter the lytic cycle (i.e. they directly force the host cell to produce replicas of the phage and then destroy the bacteria’s cell membrane, which is called lysis) or they enter the lysogenic cycle (i.e. they inject their genetic code into the bacteria’s DNA, such that phage replicas are produced in later generations of the host). The decision between lytic and lysogenic cycle has been shown to be probabilistic [ARM98], whereas a deterministic model would result in the phage always choosing one of the two pathways. Likewise, a stochastic model is needed in various cases, with another example being the circadian clocks as argued in [BL00].

Also the granularity of modeling can be an issue, where there are two extreme cases, i.e. modeling molecules individually or via concentrations. Concentrations quantify the amount of molecules per volume. Since concentrations themselves are real valued quantities and assumed to change continuously, their behavior is usually described using differential equations. This aggregation can only be justified, if the number of molecules is reasonably high. Whereas in the case of a low number of particles, molecules should be modeled as such, i.e. via population counters. Hence, the state space of possible configurations, a certain system is in at each point in time, is discrete. In [BP09], Bortolussi and Policriti point out that for some systems, to show oscillatory behavior on the model level, at least a certain subset of the involved species must be represented discretely.

Throughout this thesis we will therefore use a stochastic and discrete-state modeling approach via Continuous Time Markov Chains (CTMCs) as proposed by Gillespie [Gil77]. The usual way of analyzing those models in systems biology is via simulation, i.e. the generation of large numbers of sample trajectories. Those trajectories are evaluated using statistical methods in order to retrieve estimations of stochastic quantities (like the probability of certain events happening within certain time bounds)
together with a confidence interval, expressing the quality of that estimation via the
given set of sampled trajectories. A significant drawback of simulation is the high num-
ber of individual simulation runs that are often needed to retrieve a good estimation of
the quantity of interest [WC09].

A novel idea [BMM09] of exploring a system w.r.t. oscillatory behavior is to use model
checking. In model checking, the property to check is first translated into a (modal)
logical formula with a clearly defined semantics. In the case of CTMCs, a prominent
logic is Continuous Stochastic Logic (CSL). The underlying principle of CSL model
checking, as proposed in [BH03], involves iterative transient and steady-state analysis
methods in order to check the validity of real-time probabilistic properties. The seminal
work of Ballarini et al. [BMM09] features several logical characterizations of different
aspects of oscillation. Examples are the presence or absence of permanent oscillation as
well as whether the system everlastingly shows deviations by certain amplitude levels.

As opposed to their work, here we present a general framework to detect oscillatory
behavior that can be applied to arbitrary models as well as a formal derivation of that
framework from standard mathematical concepts.

1.1 Contributions

Hence, the main contributions of this master thesis are (i) a formal approach to defining
oscillatory and periodic behavior for a general class of systems, i.e. Markovian Popula-
tion Models, (ii) a procedure to retrieve the period length of oscillatory systems even in
the presence of stochastic noise, and (iii) an automated prototypical tool chain for char-
acterizing several aspects of oscillatory behavior, comprising Prism [KNP02], a standard
probabilistic model checker.

1.2 Organization

In Section 2, Markovian Population Models are introduced as a sub-class of continuous
time Markov chains. These models incorporate a precisely defined notion of observable
behavior, demanding for adapted versions of standard CSL and bisimulation, which is
the equivalence relation used to ensure that the overall behavior of a system will be
preserved during several model expansions presented later on in Section 6. Section 3 is
devoted to a detailed discussion on why the traditional approaches of systems biology,
mathematics and physics to define and detect oscillatory and periodic behavior fail in our
setting. We will finally revise the well-known mathematical concepts of oscillation and
periodicity in Section 4 and adapt these to our notion of observable behavior. We will
also combine both definitions in order to develop a noise resistant notion of periodicity.

Since we chose a model checking based approach to decide on the overall behavior of a
system, we will logically characterize those definitions in Section 5. In order to optimize
the model checking procedure for oscillatory behavior, we will present and discuss two
model expansion methods in Sections 6.1 and 6.2. Model checking the noise resistant
version of periodicity demands that the states of the system to analyze include additional
information about start and end of periods. For this, a product automata construction,
combining the original system and a period detector automaton is introduced in Section 6.3. That period detector is a special finite state automaton, recognizing different stages in the system’s cycles by keeping track of the progression in the observations that can be made. The tool-chain used for automating the task of model checking oscillatory and periodic behavior is introduced in Section 7. Two case studies are then used to evaluate our approach in Section 8. Finally, Section 9 concludes this thesis and briefly discusses future work.
2 Preliminaries

As already motivated before, we are using a purely discrete setting w.r.t. the states our systems will be in. The possible transitions, i.e. state changes, are stochastic and time passes continuously. We will also restrict to finite state spaces. The basic model structure that incorporates all of these properties are (finite) continuous time Markov chains.

2.1 Continuous Time Markov Chains

Definition 1 (Continuous Time Markov Chain). A (labeled) Continuous Time Markov Chain (CTMC) is a tuple $(S, R, AP, L)$, where $S$ is a finite set of states, $R : S \times S \to \mathbb{R}_{\geq 0}$ is a rate matrix, $AP$ is a set of atomic propositions, and $L : S \to 2^{AP}$ is a labeling function.

We further demand that the CTMC under consideration has no self-loops, i.e. the rate matrix $R$ possesses a zero-diagonal, that is for each state $s$ holds that the rate $R(s, s)$ back to $s$ is zero. This restriction is justified in our setting, since the more refined model structure presented later on only contains information about the rates of changes in state. E.g. any reaction type in a biological system will change the molecule level of at least one species.

Definition 2 (Initial Distribution). An initial distribution of a CTMC $C = (S, R, AP, L)$ is a function $\alpha : S \to [0, 1]$ such that $\Sigma_{s \in S} \alpha(s) = 1$.

For each state of a CTMC, the initial distribution contains information about the probability of the system starting in that specific state. E.g. in order to make a certain state $s_{\text{start}}$ the single start state, an initial distribution $\alpha$ with $\alpha(s_{\text{start}}) = 1$ and $\alpha(s) = 0$ for all $s \in S$ with $s \neq s_{\text{start}}$ is used.

Semantics: In the following, an intuitive description of the semantics of CTMCs and some basic analysis methods, needed for several example systems presented later on, is given. The reader is referred to [Nor97] for a formal introduction to CTMCs via stochastic processes. In principle, the semantics of CTMCs is governed by their rate matrix $R$. At each moment in time $t$, the system is in a unique state $s$. CTMCs are time-homogeneous, meaning that the given rate matrix is independent of time and hence does not change. Now, for any CTMC $C = (S, R, AP, L)$, two basic probability distributions for states can be calculated:

- the probability distribution of the successor state.

CTMCs possess the Markov property, i.e. under the condition the system is in some specific state $s$, the probability of the successor state only depends on the current state $s$ and is therefore independent of the rest of the system’s history.
If the current state \( s \) has several possible successor states, i.e. states \( s' \in S \) with \( R(s, s') > 0 \), there is a race going on in-between those states, where the probability \( P_{\text{succ}}(s, s') \) of winning, i.e. of becoming the actual successor, is proportional to the respective rate \( R(s, s') \). Formally,

\[
P_{\text{succ}}(s, s') = \frac{R(s, s')}{E(s)}
\]

where \( E(s) = \sum_{s' \in S} R(s, s') \) is the exit rate, i.e. the total rate of following any transition.

- the probability distribution \( P_s(t_{\text{res}} \geq t) \) of the residence time \( t_{\text{res}} \).

Under the condition, the system is in some state \( s \), the time \( t_{\text{res}} \) staying in that state \( s \) before leaving it via any transition behaves according to

\[
P_s(t_{\text{res}} > t) = e^{-E(s)\cdot t}
\]

Since \( P_s(t_{\text{res}} > t) \) is exponentially distributed and therefore is memoryless, we have that for any (positive) additional residence time \( t' \) holds \( P_s(t_{\text{res}} > t + t' \mid t_{\text{res}} > t) = P_s(t_{\text{res}} > t') \).

The expected time \( E_{\text{leave}}(s) \) of leaving a state \( s \in S \) is given by

\[
E_{\text{leave}}(s) = \frac{1}{E(s)}.
\]

**Example 1.** Figure 1 depicts the underlying graph of the CTMC

\[
C = (\{s_0, s_1, s_2\}, R, \{\text{start}, \text{accept}\}, L)
\]

with \( R(s_0, s_1) = \lambda_1, R(s_1, s_0) = \lambda_1, R(s_1, s_2) = \lambda_2, R(s_2, s_0) = \lambda_3, \) and \( 0 \) else. Further, let \( L(S_0) = \{\text{start}\}, L(S_1) = \emptyset \) and \( L(S_2) = \{\text{accept}\} \). Since the successor probability and the residence time are stochastically independent, we can calculate the probability

\[
P(s_1 \xrightarrow{\leq t} s_2)
\]
of doing a transition from \( s_1 \) to \( s_2 \) within \( t \) time units as

\[
P(s_1 \xrightarrow{\leq t} s_2) = \left(1 - e^{-(\lambda_2 + \lambda_4)t}\right) \cdot \frac{\lambda_2}{\lambda_2 + \lambda_4}.
\]

The expected time to leave state \( s_1 \) is

\[
E_{\text{leave}}(s_2) = \frac{1}{\lambda_2 + \lambda_4}.
\]

In order to get a greater variety of more powerful analysis methods for CTMCs exceeding the limited capabilities of the aforementioned basic probability calculations, we will later on switch to logical characterizations of properties that can be checked on CTMCs. The logic of choice will be Continuous Stochastic Logic (CSL) [BH03].

### 2.1.1 \( F \) Bisimulation

In Section 6 we will extend CTMCs to incorporate additional information. For this we need a correctness criterion in order to ensure that the overall behavior will be preserved by the construction. The criterion of choice is to demand that the original CTMC and the extended one should be equivalent w.r.t. an \( F \) bisimulation. This is an obvious choice, since CSL equivalence coincides with \( F \) Bisimilarity [BH03]. In the following, we define the projection \( X|_Y \) of a set \( X \) onto a set \( Y \) as \( X|_Y = \{ x \in X \mid x \in Y \} \) and for two functions \( f : X \rightarrow Z \) and \( f' : Y \rightarrow Z \) with \( X \cap Y = \emptyset \) we define their disjoint union \( f \cup f' : X \cup Y \rightarrow Z \) as

\[
f \cup f' : v \mapsto \begin{cases} f(v) & \text{if } v \in X, \\ f'(v) & \text{otherwise.} \end{cases}
\]

**Definition 3 (\( F \) Bisimulation).** An \( F \) bisimulation on a CTMC \( C = (S, R, AP, L) \) is an equivalence relation \( R \) on \( S \) such that, whenever \( (s, s') \in R \), then

\[
L(s)|_F = L(s')|_F \text{ and } R(s, C) = R(s', C') \text{ for all } C \in S/R,
\]

where \( S/R \) denotes the quotient space under \( R \) and \( R(s, C) = \sum_{s' \in C} R(s, s') \). States \( s \) and \( s' \) are \( F \) bisimilar if there exists an \( F \) bisimulation \( R \) that contains \( (s, s') \).

**Definition 4 (\( F \) Bisimilarity of CTMCs).** Two CTMCs \( C = (S, R, AP, L) \) and \( C' = (S', R', AP', L') \) with initial distributions \( \alpha \) and \( \alpha' \) are \( F \) bisimilar, written

\[
C \sim_F C'
\]

iff there exists an \( F \) bisimulation \( R \) on \( C \cup C' = (S \cup S', R \cup R', AP \cup AP', L \cup L') \) such that

\[
\forall C \in S \cup S'/R. \alpha(C|_S) = \alpha'(C|_{S'})
\]

with \( \alpha(X) = \sum_{x \in X} \alpha(x) \) and \( \alpha'(X') = \sum_{x' \in X'} \alpha'(x') \) for any subset \( X \subseteq S \) and \( X' \subseteq S' \).

Please note that the disjoint unions in the above definition can always be achieved via renaming of states.
2.1.2 Continuous Stochastic Logic

As already mentioned before, we use Continuous Stochastic Logic (CSL) as presented in [BH03] when analyzing the behavior of CTMCs. But we only need a certain subset of full CSL, i.e. we omit the next-operator.

Definition 5 (Continuous Stochastic Logic). Let \( p \in [0, 1] \) be a real number, \( \in \subseteq \{\leq, <, >, \geq\} \) a comparison operator, and \( I \subseteq \mathbb{R}_{\geq 0} \) a non-empty interval. The syntax of CSL formulas, ranged over by \( \Phi, \Psi \), over the set of atomic propositions \( AP \), is defined as

\[
\Phi, \Psi ::= \text{true} \mid a \in AP \mid \neg \Phi \mid \Phi \land \Psi \mid S_{\leq p}(\Phi) \mid P_{\leq p}(\Phi U t \Psi)
\]

CSL formulas are interpreted over the states of a CTMC \( C = (S, R, AP, L) \). The satisfaction relation \( \models \) relates a state \( s \in S \) with all CSL formulas that are satisfied by \( s \). Let \( \pi^C(s, S') \) with \( s \in S \) and \( S' \subseteq S \) denote the steady state probability, i.e. the probability in the long-run, of being in a state \( s' \in S' \) when starting in \( s \) as derived in [BH03]. We need this probability measure in order to define the semantics of the steady state operator \( S_{\leq p} \). Additionally, we require a notion of paths in CTMCs to be able to define the formal meaning of the path probability measure operator \( P_{\leq p} \).

Definition 6 (Paths in CTMCs). Given a CTMC \( C = (S, R, AP, L) \), an infinite path \( \sigma \) is a sequence \( s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} s_2 \xrightarrow{t_2} \ldots \) with \( s_i \in S \), \( t_i \in \mathbb{R}_{\geq 0} \) and \( R(s_i, s_{i+1}) > 0 \) for all \( i \in \mathbb{N}_0 \). A finite path \( \sigma \) is a sequence \( s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} \ldots \xrightarrow{t_{l-2}} s_{l-1} \xrightarrow{t_{l-1}} s_l \) such that \( s_l \) is absorbing, i.e. \( \forall s \in S. R(s_i, s) = 0 \), and \( R(s_i, s_{i+1}) > 0 \) for all \( i < l \).

Given an infinite path \( \sigma \), let \( \sigma[i] = s_i \) and \( \delta(\sigma, i) = t_i \) for \( i \in \mathbb{N} \). Further, let \( \sigma @ t = \sigma[i] \) for \( t \in \mathbb{R}_{\geq 0} \) and \( i = \min\{z \leq \sum_{k=0}^{l-1} t_k\} \). Now, if \( \sigma \) is a finite path, ending in state \( s_l \), \( i < l \), and \( t \leq \sum_{k=0}^{l-1} t_k \), the definitions are as above. For \( i = l \), \( \delta(\sigma, l) = \infty \) and undefined otherwise, and for \( t > \sum_{k=0}^{l-1} t_k \) we define \( \sigma @ t = s_l \). Moreover, let \( Path_s \) denote the set of paths starting in state \( s \) and let \( Pr_s Y \) denote the probability measure of the set of paths \( Y \) in \( C \) for starting state \( s \), as derived in [BH03].

Finally, we can define the formal semantics of CSL in terms of the satisfaction relation \( \models \in S \times \Phi \) that contains exactly those pairs \((s, \phi)\) of states \( s \) and CSL formulas \( \phi \), for which the formula is satisfied in the state.

Definition 7 (CSL Semantics). Let \( \text{Sat}(\Phi) = \{s \in S \mid s \models \Phi\} \). The relation \( \models \) is then defined as

\[
\begin{align*}
s &\models \text{true} \quad \text{for all } s \in S \\
s &\models a \quad \text{iff } a \in L(s) \\
s &\models \neg \Phi \quad \text{iff } s \not\models \Phi \\
s &\models \Phi \land \Psi \quad \text{iff } s \models \Phi \text{ and } s \models \Psi \\
s &\models S_{\leq p}(\Phi) \quad \text{iff } \pi^C(s, \text{Sat}(\Phi)) \leq p \\
s &\models P_{\leq p}(\Phi U t \Psi) \quad \text{iff } Pr_s\{\sigma \in Path_s \mid \exists t \in L. \sigma @ t \models \Psi \land (\forall t' \in [0, t), \sigma @ t' \models \Phi)\} \leq p
\end{align*}
\]
Other Boolean connectives can be constructed from the basic ones, e.g. \( \Phi \lor \Psi = \neg(\neg\Phi \land \neg\Psi) \), \( \Phi \Rightarrow \Psi = \neg\Phi \lor \Psi \), and also \( \text{false} = \neg\text{true} \). Other important temporal operators can be derived from the until operator. The \textit{eventually} operator is defined as
\[
P_{\leq p}(\Diamond_I \Phi) = \mathcal{P}_{\leq p}(\text{true} \cup I \Phi)
\]
and the \textit{always}-operator can finally be derived from the eventually operator via
\[
P_{\leq p}(\Box_I \Phi) = \mathcal{P}_{\leq p}(\Diamond_I \neg \Phi)
\]
where \( \neg < = \geq, \neg \leq = >, \neg > = \leq \) and \( \neg \geq = < \). We omit the time interval \( I \) for untimed versions of the temporal operators and implicitly assume \( I = [0, \infty) \).

\[2.2\text{ Markovian Population Models}\]

Although, our main interest will be the analysis of the behavior of biological systems we define our framework to be \textit{Markovian population models}. This class of systems is more general and also allows the treatment of differently motivated systems from areas like queuing and computer networks [HJW09].

The underlying structure of a Markovian Population Model is a CTMC in which each state uniquely encodes the current number of \textit{individuals} of certain \textit{population types}. In order to reason about the behavior, i.e. the changes in the number of individuals, an \textit{observable} is used to aggregate a state’s information to a single quantity. In the context of oscillatory behavior we will use \textit{linear functionals}.

**Definition 8** (Linear Functional). A function \( f : \mathbb{R}^N \rightarrow \mathbb{R}, x \mapsto f(x) \) is a \textit{linear functional} on \( \mathbb{R}^N \) with \( N \in \mathbb{N} \) iff \( \forall u, v \in \mathbb{R}^N, a \in \mathbb{R} \):

1. \( f(a \cdot v) = a \cdot f(v) \) (Homogeneity)
2. \( f(u + v) = f(u) + f(v) \) (Linearity)

Due to the homogeneity and linearity constraints on linear functionals, we are assured, that we will not artificially introduce oscillatory behavior, i.e. observe oscillations where there actually are none. But still, a linear functional as an observable has to be chosen with care since the other direction does not hold. For example, the function \( f \) with
\[
f : \mathbb{R}^N \rightarrow \mathbb{R}, v \mapsto 0
\]
is a linear functional for any \( N \in \mathbb{N} \), but no change in the population will ever be visible for any system. We can finally define Markovian population models as a sub-class of CTMCs incorporating linear functionals used as the notion of observation that can be made at the state level.

**Definition 9** (Markovian Population Model and Observable). A \textit{finite Markovian Population Model} (MPM) for \( N \in \mathbb{N} \) distinct population types is a CTMC \( \mathcal{M} = (S, R, AP, L) \), where \( S \subseteq \mathbb{N}_0^N \) with \( |S| \in \mathbb{N} \). An observable for \( \mathcal{M} \) is a linear functional \( \mu \) on \( \mathbb{R}^N \).
2.2.1 Continuous Stochastic Logic with Comparisons

States $s$ and $s'$ of a MPM can not only be distinguished by their atomic propositions like in ordinary CTMCs, but also by their observations. Thus, we also have to incorporate this increase in expressiveness within our logic. In detail, we extend CSL to also allow for the evaluation of boolean expressions, comparing the current observable $\mu$ or an atomic proposition $a$ on state level against a fixed value $x \in \mathbb{R}$. We call that extended logic CSL with comparisons ($CSL_c$). For comparisons against an atomic proposition $a$ we demand that each state $s$ in the MPM to be labeled with an unique assignment $a = x$ for some value $x \in \mathbb{R}$ and define $eval(s, a)$ to retrieve that value $x$, i.e. $eval(s, a) = x$. Formally, we add the production rules

$$\Phi, \Psi ::= \mu \leq x \mid a \leq x$$

with their semantics defined as

$s \models \mu \leq x$ if $\mu(s) \leq x$, and  
$s \models a \leq x$ if $eval(s, a) \leq x$.

Also when intermediately characterizing several properties in Linear Temporal Logic (LTL) [Pnu77] and Computation Tree Logic (CTL) [CE82] in Section 5, we assume these logics to be extended to also allow for those comparisons in a similar way.

Model Checking CSL c: Obviously, we can reduce the model checking problem for $CSL_c$ formulas to normal CSL model checking. The basic idea is to express each $CSL_c$ formula $\Phi$ containing comparisons on a MPM $\mathcal{M} = (S, R, AP, L)$ by an equivalent CSL formula $\Phi'$ on a slightly altered MPM $\mathcal{M}' = (S, R, AP', L')$. For this purpose we add for each sub-formula $\Phi_{sub}$ of $\Phi$, which is a comparison, a new atomic proposition $a_{sub}$ to $AP'$. Now, for each state $s$ in $\mathcal{M}$ we demand that $a_{sub} \in L'(s)$ iff $s \models \Phi_{sub}$. Checking $\Phi$ on $\mathcal{M}$ is then equivalent to checking formula $\Phi'$ on $\mathcal{M}'$, where each occurrence of a comparison $\Phi_{sub}$ is replaced by the atomic proposition $a_{sub}$ inside formula $\Phi$. We demand that for the sets of atomic propositions holds that $AP \subseteq AP'$ and that for each state $s \in S$ the labelings $L(s)$ and $L'(s)$ agree on $AP$. Obviously, the time needed to model check these extended formulas lies in $\mathcal{O}(|S|)$.

2.2.2 $F, \mu$ Bisimulation

Now, our logic CSL with comparisons allows to distinguish states not only on the level of atomic propositions but also on their observations $\mu$. Hence, we also have to extend the notion of bisimulation to cope with that increased expressiveness.

Definition 10 ($F, \mu$ Bisimulation). An $F, \mu$ bisimulation on a MPM $\mathcal{M} = (S, R, AP, L)$ with observable $\mu$ is an equivalence relation $\mathcal{R}$ on $S$ such that, whenever $(s, s') \in \mathcal{R}$, then

$L(s)|_F = L(s')|_F$, $\mu(s) = \mu(s')$ and $R(s, C) = R(s', C)$ for all $C \in S/\mathcal{R}$.
where $S/R$ denotes the quotient space under $R$ and $R(s,C) = \Sigma c \in C R(s,c)$. States $s$ and $s'$ are $F, \mu$ bisimilar iff there exists an $F, \mu$ bisimulation $R$ that contains $(s, s')$.

**Definition 11** ($F, \mu$ bisimilar MPMs). Two MPMs $M_1 = (S_1, R_1, AP_1, L_1)$ and $M_2 = (S_2, R_2, AP_2, L_2)$ with observables $\mu_1, \mu_2$, and initial distributions $\alpha_1, \alpha_2$ are $F, \mu$ bisimilar with $\mu = \mu_1 \cup \mu_2$, written

$$M_1 \sim_{F, \mu} M_2$$

iff there exists an $F, \mu$ bisimulation $R$ on $M_1 \cup M_2 = (S_1 \cup S_2, R_1 \cup R_2, AP_1 \cup AP_2, L_1 \cup L_2)$ with observable $\mu$ such that

$$\forall C \in S_1 \cup S_2 / R. \ \alpha_1(C | S_1) = \alpha_2(C | S_2)$$

with $\alpha(X) = \sum_{x \in X} \alpha(x)$.

Obviously, $F, \mu$ bisimulation equivalence and CSL$_c$ equivalence coincide for Markovian Population Models.

**Proposition 1** ($F, \mu$ Bisimulation Equivalence and CSL$_c$ equivalence coincide). For two MPMs $M_1 = (S_1, R_1, AP_1, L_1)$ and $M_2 = (S_2, R_2, AP_2, L_2)$ with observables $\mu_1, \mu_2$ and initial distributions $\alpha_1, \alpha_2$ holds

$$M_1 \sim_{F, \mu} M_2 \iff [\forall \Phi \in CSL_c. M_1, M_2 \models \Phi]$$

**Proof.** (Sketch) It suffices to show that for two states $s_1 \in S_1$ and $s_2 \in S_2$ holds

$$\mu_1(s_1) = \mu_2(s_2) \iff [\forall x \in R, \leq \in \{<, \leq, =, \geq, >\}, s_1 \models \mu \leq x \iff s_2 \models \mu \leq x] \quad (\ast)$$

since then, an analogous proof strategy for $F$ bisimulation and CSL equivalence as in [BH03] can be used. Proof of $(\ast)$:

"$\Rightarrow$": $s_1 \models \mu \leq x \iff s_2 \models \mu \leq x \iff \mu_1(s_1) \leq x \iff \mu_2(s_2) \leq x \iff \mu_1(s_1) \leq x \iff \mu_2(s_2) \leq x \iff$ true

"$\Leftarrow$": $\forall x \in R, \leq \in \{<, \leq, =, \geq, >\}, s_1 \models \mu \leq x \iff s_2 \models \mu \leq x \implies s_1 \models \mu = x \iff s_2 \models \mu = x \iff \mu_1(s_1) = \mu_2(s_2)$

Please note that since for $F, \mu$ bisimulation, the labelings of bisimilar states have to coincide anyways, nothing has to be shown for $a \leq x$ sub-expressions.
2.2.3 Markovian Population Models of Biological Reaction Networks

Now, we will introduce the type of systems we will mainly be interested in, which are biological reaction networks. Assume we want to model a biological system with \( N \) species \( S_1, \ldots, S_N \) and \( M \) distinct reaction types \( R_1, \ldots, R_M \) with reaction rate constants \( c_1, \ldots, c_M \in \mathbb{R}_{>0} \). Furthermore, each \( R_m \) with \( m \in \{1, \ldots, M\} \) has the form

\[
a_{m1} \cdot S_1 + \cdots + a_{mN} \cdot S_N \xrightarrow{c_m} b_{m1} \cdot S_1 + \cdots + b_{mN} \cdot S_N
\]

with \( a_{mi}, b_{mi} \in \mathbb{N}_0 \) for \( m \in \{1, \ldots, M\}, i \in \mathbb{N} \). If for a reactant or product species the corresponding coefficient inside a reaction type is zero, the species will be omitted. Also, coefficients equal to one will be omitted as well. For example, for a system involving two species \( A \) and \( B \), we write \( A \xrightarrow{c} B \) instead of \( 1 \cdot A + 0 \cdot B \xrightarrow{c} 0 \cdot A + 1 \cdot B \). Further assume that initially there exist exactly \( I_j \in \mathbb{N}_0 \) molecules of each species \( S_j \). For each reaction type \( R_m \) we can construct a change vector \( v_m = (v_{m1}, \ldots, v_{mN}) \in \mathbb{Z}^N \) with

\[
v_m = \sum_{i=1}^{N} b_{mi} \cdot \delta_i - \sum_{i=1}^{N} a_{mi} \cdot \delta_i
\]

where \( \delta_i \) denotes the \( i \)-th unit vector, i.e. \( \delta_i = (\delta_{i1}, \ldots, \delta_{iN})^T \) with \( \delta_{ii} = 1 \) and \( \delta_{ij} = 0 \) for any \( i \neq j \). For the reaction types to be well-formed we demand that for each change vector \( v_m \) holds \( v_m \neq (0, \ldots, 0)^T \). The corresponding propensity functions \( \gamma_m : \mathbb{N}_0^N \rightarrow \mathbb{R}_{\geq 0} \) are defined as :

\[
\gamma_m(x) = c_m \cdot \prod_{i=1}^{N} \left\lfloor \frac{x_i}{a_{mi}} \right\rfloor
\]

where for \( x = (x_1, \ldots, x_N)^T \), \([x]_i = x_i \) denotes the projection of the vector \( x \) onto its \( i \)-th component and

\[
\left[ \begin{array}{c}
\binom{n}{k} \\

t_k
\end{array} \right] = \begin{cases}
\binom{n}{k} & \text{if } n \geq k, \\
0 & \text{otherwise.}
\end{cases}
\]

We can now model such a system using a MPM \( \mathcal{M} = (S, R, AP, L) \) for \( N \) population types and an observable \( \mu \), where

\[
R(s, s') = \sum_{1 \leq m \leq M, s+v_m = s'} \gamma_m(s)
\]

and the corresponding initial distribution

\[
\alpha(s) = \begin{cases}
1 & \text{if } s = (I_1, \ldots, I_N), \\
0 & \text{otherwise.}
\end{cases}
\]

Of course, initial distributions for biologically motivated systems are not restricted to have a single initial state but may also distribute over several states. This construction is justified by the work of Gillespie [Gil77]. For the observable \( \mu \) we may use the convenient notion of coefficient vector induced linear functionals.
Proposition 2 (Coefficient Vector Induced Linear Functional). Given any vector \( v = (v_1, \ldots, v_N)^T \in \mathbb{R}^N \) (the coefficient vector), the function \( \mu_v : \mathbb{R}^N \to \mathbb{R}_{\geq 0}, x \mapsto \langle x, v \rangle \), with \( \langle x, v \rangle = \sum_{i=1}^{N} x_i \cdot v_i \), is a linear functional on \( \mathbb{R}^N \).

Proof. Let \( v = (v_1, \ldots, v_N)^T \in \mathbb{R}^N \), \( s \in \mathbb{R} \) and \( w, z \in \mathbb{R}^N \):

Homogeneity:
\[
\mu_v(s \cdot w) = \sum_{i=1}^{N} s \cdot w_i \cdot v_i = s \cdot \sum_{i=1}^{N} w_i \cdot v_i = s \cdot \sum_{i=1}^{N} v_i = s \cdot \mu_v(w)
\]

Linearity:
\[
\mu_v(w + z) = \sum_{i=1}^{N} (w_i + z_i) \cdot v_i = \sum_{i=1}^{N} (w_i \cdot v_i) + \sum_{i=1}^{N} (z_i \cdot v_i)
\]
\[
= \sum_{i=1}^{N} w_i \cdot v_i + \sum_{i=1}^{N} z_i \cdot v_i = \mu_v(w) + \mu_v(z)
\]

Consequently, we define the observable \( \mu \) of our MPM to be induced by some coefficient vector \( v \). This way it is able to represent all kinds of quantities expressible via weighted sums, like for example:

- the projection on the \( i \)-th population type (like stated above) via \( v = \delta_i \).
- the sum of two or more population types via \( v = \sum_{i \in I} \delta_i \), where \( I \) denotes the set of population type indices of interest.
- the average of two or more population types via \( v = \frac{1}{|I|} \cdot \sum_{i \in I} \delta_i \).
3 Defining and Detecting Oscillatory and Periodic Behavior

Now that we have introduced the type of models we will use throughout this thesis, we first have to define the general notion of observable behavior w.r.t. MPMs, in order to be able to formally define oscillatory and periodic behavior. For CTMCs in general, this would be the concept of paths like defined in Definition 6. In the case of MPMs, we are not interested in the timed sequence of atomic propositions of the underlying CTMC, but rather in the observable $\mu$ of the states visited over time. Hence, we project the observable’s value on time and retrieve the notion of $\mu$-trajectories.

Definition 12 ($\mu$-Trajectories of MPMs). Given a MPM $M = (S, R, AP, L)$ with observable $\mu$ and an infinite path $\sigma = s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} s_2 \xrightarrow{t_2} \ldots$ the corresponding $\mu$-trajectory $\tau_\sigma : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}$ is defined such that for all $i \in \mathbb{N}_0$ and any $t \in [0, t_i)$ we have that $\tau_\sigma(t + \sum_{j<i} t_j) = \mu(s_i)$. For a finite path $\sigma = s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} \ldots \xrightarrow{t_{l-2}} s_{l-1} \xrightarrow{t_{l-1}} s_l$ we define $\tau_\sigma$ such that for all $i \leq l - 1$ and $t \in [0, t_i)$, $\tau_\sigma(t + \sum_{j<i} t_j) = \mu(s_i)$, and for any $t \geq \sum_{i=0}^{l-1} t_i$, we have that $\tau_\sigma(t) = \mu(s_l)$.

Example 2. Consider a MPM $M = (S, R, AP, L)$ with observable $\mu$ and an infinite path

$$\sigma = s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} s_2 \xrightarrow{t_2} \ldots$$

in $M$. Assume that $s_l, s_m, s_h \in S$, $\mu(s_l) = 0$, $\mu(s_m) = 1$ and $\mu(s_h) = 2$, further assume that the sequence of intermediate states $s_0 s_1 s_2 s_3 s_4 s_5 \ldots$ of our path $\sigma$ is $(s_l s_m s_h s_m)^\omega$. Then, the corresponding $\mu$-trajectory of $\sigma$ is a function $\tau_\sigma : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}$ as depicted in Figure 2.

![Figure 2: Plot of a sample $\mu$-trajectory.](image)

Please note that usually when trajectories will be illustrated, for convenience, a hist-
rogram step style will be used, although the function is only piecewise defined between jump points, i.e. at points in time at which transitions have just been taken.

3.1 Continuous Deterministic Solutions of Biological Systems

At first we will illustrate why the traditional approach of systems biology to analyze biological systems can fail when trying to detect oscillatory and periodic behavior. We will do that by solving the Ordinary Differential Equation (ODE) system induced by the reaction types of an example system, the 3-way Oscillator as described in [BMM09] and first introduced in [Car06]. Please note that given a biological reaction network and the respective ODE solution and Markovian population model, the ODE solution approximates the expectation of the species’ molecule level in the MPM model interpreted per volume, i.e. as concentrations, if the molecule numbers and volume are considered in the limit [Kur72]. Consequently, the ODE solution is deterministic as it aggregates all possible progressions of the concentrations via their expectation, whereas in the stochastic model, i.e. the MPM, any possible evolution can be witnessed with a certain probability. For a MPM, such individual evolutions, i.e. \( \mu \)-trajectories (of finite length) can be retrieved via Gillespie simulation [Gil77], where each simulation run, i.e. random experiment respecting the MPM’s structure, corresponds to a unique (sample) \( \mu \)-trajectory.

The basic system consists of three species \( A, B \) and \( C \) that influence their production in a round robin fashion via positive feedback loops. More precisely, species \( A \) boosts the production of species \( B \), \( B \) boosts \( C \) and \( C \) boosts \( A \). Consequently, the basic 3-way Oscillator’s reaction network consists of three reaction types:

\[
A + B \xrightarrow{r_A} B + B \quad B + C \xrightarrow{r_B} C + C \quad C + A \xrightarrow{r_C} A + A
\]

Unfortunately, that basic system – although possibly oscillating in the beginning – stops reacting after one species gets extinct. E.g. if at some point in time all \( A \) molecules have been transformed to \( B \) molecules via the first equation, also all \( B \) molecules will inevitably be transformed to \( C \) molecules via the second equation and the cycle stops, since no reaction according to the third equation may happen due to the lack of \( A \)-molecules. In [Car06], Cardelli also shows how to make the 3-way Oscillator permanently oscillating. The trick is to introduce three additional species \( DA, DB \) and \( DC \) acting as doping substrates for the three original species. The behavior of these species is described by three additional equations

\[
DA + C \xrightarrow{r_C} A + DA \quad DB + A \xrightarrow{r_A} B + DB \quad DC + B \xrightarrow{r_B} C + DC
\]

i.e. each doping substance enables the transformation of a molecule of one species into another species, preventing the deadlock situation described above. The initial existence of one molecule per doping species is sufficient to make the whole system oscillating permanently, since the doping substances act like catalysts not disappearing after the catalysis. The usual way in systems biology to model a biological reaction network is via
a system of (coupled) ordinary differential equations (ODEs) derived via the law of mass action (as can be seen in [WC09]). For the 3-way Oscillator with doping we retrieve:

\[
\frac{d}{dt} A = r_C \cdot C \cdot (A + DA) - r_A \cdot A \cdot (B + DB)
\]

\[
\frac{d}{dt} B = r_A \cdot A \cdot (B + DB) - r_B \cdot B \cdot (C + DC)
\]

\[
\frac{d}{dt} C = r_B \cdot B \cdot (C + DC) - r_C \cdot C \cdot (A + DA)
\]

\[
\frac{d}{dt} DA = 0 \quad \frac{d}{dt} DB = 0 \quad \frac{d}{dt} DC = 0
\]

We are using Matlab\(^1\) routine twooode() (cf. Appendix A.1) to solve this equation system. Routine twooode first solves the ODE specified by sub-routine two_ode for the time interval \([0, 20]\) and initial concentrations \(15.0 \text{ mol}\) for species \(A\), \(0.0 \text{ mol}\) for species \(B\) and \(C\) and \(1.0 \text{ mol}\) for the doping species. The reaction rates \(r_A\), \(r_B\) and \(r_C\) are all set to 1.0. The resulting plot is illustrated in Figure 3. As can be seen, the deterministic solution of the system quickly converges to an equilibrium state after around three time units, in which species \(A\), \(B\) and \(C\) are present with the same concentration of \(5.0 \text{ mol}\) and hardly no oscillation of the species is visible afterwards. When initializing the system with that equilibrium state, i.e. a concentration of \(5.0 \text{ mol}\) for each of the species \(A\), \(B\) and \(C\), no single period is visible, not even in the beginning (cf. Fig. 4).

Now, we use the discrete stochastic approach described in Section 2.2.3 implemented within our tool, described later on in Section 7, and Prism in order to model the 3-way Oscillator and generate sample trajectories from it, mimicking Gillespie simulation. One of those sampled trajectories is illustrated in Figure 5. For this approach we assumed an amount of 5 molecules per species \(A\), \(B\) and \(C\) and one molecule per doping substance. As we will see in Section 8.1, indeed almost all \(\mu\)-trajectories oscillate. This illustrates the fact that for many systems showing oscillatory behavior, a treatment on the molecule level, i.e. a discrete and stochastic approach, should be followed (at least for certain species of a system) like already argued in detail by Bortolussi and Policriti in [BP09].

\(^1\)http://www.mathworks.de/
Figure 3: Plot of the solution to the (doped) 3-way Oscillator ODE with initial concentration $15.0 \text{ mol/l}$ of species $A$.

Figure 4: Plot of the solution to the (doped) 3-way Oscillator ODE with initial concentration $5.0 \text{ mol/l}$ of each species, $A$, $B$ and $C$. 

16
Figure 5: Plot of a sample trajectory projected onto the molecule level of species A of the (doped) 3-way Oscillator generated with Prism. The initial amount of molecules is 5 for each of the species A, B and C, and one for the doping substances.

3.2 Fourier Transform

The usual approach followed in fields like mathematics and physics, to detect oscillatory and periodic behavior within a function, is to use Fourier transform. A function’s behavior then is characterized by the strongest frequency component(s) that make up the function.

**Definition 13** (Fourier Transform). *Given an integrable function $f : \mathbb{R} \rightarrow \mathbb{R}$, the Fourier transform $\hat{f} : \mathbb{R} \rightarrow \mathbb{C}$ is defined via*

$$\hat{f}(a) := \int_{-\infty}^{\infty} f(x) e^{-2\pi iax} dx$$

*for any $a \in \mathbb{R}$.*

For a function $f$ and an ordinary frequency $a$ (in Hertz), the absolute value of the Fourier transform

$$|\hat{f}(a)| = \sqrt{\text{re}(\hat{f}(a))^2 + \text{im}(\hat{f}(a))^2}$$

can be used to reason about the presence of a periodic wave with period length $\frac{1}{a}$ inside the function (although interference effects like cancellation and amplification of different wave lengths have to be considered).
This approach works well for individual functions. Consequently, it could be used for single $\mu$-trajectories for example generated via Gillespie simulation. But there is a problem concerning the Fourier transformation of $\mu$-trajectories, which results from the trajectories’ discreteness in their co-domain. Because of the discontinuities in the observable at jump times of the model’s underlying CTMC, the $\mu$-trajectories are not continuous, hence it is not clear what sampling interval to take for the discrete Fourier transformation, needed for analyzing those trajectories. A solution could be the approach described in [HB86] that allows for the generation of a periodogram, i.e. a plot of possible period lengths against the their strength within the function, even for unevenly sampled time series.

But still, a crucial problem of simulation is the amount of simulation runs that have to be made to get a reliable result, which usually is exponential in the desired precision, even for simple quantities like the expectation of a probability of some event occurring. A more severe drawback of simulation is the inability to reason about infinite behavior, like ensuring that oscillations that might be visible in any time bounded simulation run do not stop some time later.

An alternative would be to not to generate large amounts of individual simulation runs, but to treat all or at least almost all $\mu$-trajectories at once and sample the transient probability distribution of the observations $\mu$ as a measure for the current amplitude. Many CSL model checkers not only allow the retrieval of boolean answers on whether a certain formula holds for a given probability bound but also, in the case of the probability or steady state operators, with which actual probability. Given a MPM $\mathcal{M} = (S, R, AP, L)$ with observable $\mu$, the transient probability distribution of the observable can be retrieved by querying a model checker like Prism for each possible observation $x \in \mu(S)$ via

$$\mathcal{P}_{=\tau}[\mathcal{O}_{(t,t)} \mu = x].$$

Indeed, Prism will be the model checker of choice when later on dealing with a practical implementation of detecting oscillatory behavior.

Unfortunately, another problem arises from that superimposed inspection of all possible $\mu$-trajectories, i.e. cancellation and averaging effects force the transient probability distribution of the observable to converge towards a steady state. Indeed, the transient probability distribution over states of every CTMC will converge towards a steady state distribution [Kul95]. Hence, the transient probability distribution is an unsuitable amplitude measure, needed for Fourier transform. Since, as time progresses, inevitably less and less deviations from the steady state distribution will be visible. This observation is illustrated by the following example.

**Example 3** (Convergence to Steady State). Consider the Prism model of a very simple system (cf. Fig. 6, left) consisting of two states encoding the presence ($c = 1$) and absence ($c = 0$) of a certain population. The corresponding observable $\mu$ simply projects on the current population level $c$. Using the analysis methods of Section 2.1 w.r.t. the underlying CTMC (cf. Fig. 6, right) one can reason that the expected residence time for both states $s_0$ and $s_1$ is $\frac{1}{1^2} = \frac{1}{1^2} = 1$ and with probability 1, leaving state $s_0$ results in entering state $s_1$ and vice versa. Using Prism, we can retrieve the probability that the
initial state $s_0$ is left for state $s_1$ within $t$ time units using the formula

$$P=?[\Diamond \leq t \ c=1]$$

which can be translated to the Prism query

$$P=? \ [true \ U\leq t \ c=1].$$

With a Prism experiment we can retrieve that probability for a varying value of $t$ with $t \in [0,5]$ in steps of 0.1 (cf. Fig. 7). For example, state $s_0$ is left with a probability of around 0.99 within 4.7 time units. The same holds for state $s_1$ leaving for state $s_0$ which can be checked by making state $s_1$ ($c = 1$) the initial state. Therefore, one can indeed expect the system to show an oscillatory behavior with a typical maximum period length of 4.7 time units in in 99% of all periods. Now, the transient probabilities of being in state $s_i$ ($c = i$), $i \in \{0,1\}$ at time $t$ can be retrieved via the formula

$$P=?[\Diamond_{[t,t]} c=i]$$

and the corresponding Prism query

$$P=? \ [true \ U[t,t] \ c=i].$$

The resulting probabilities for a varying value of $t \in [0,5]$ with step size 0.1 are illustrated in Figure 8. Obviously, the transient probabilities converge to the steady state probabilities (equal probability of 0.5 to be in either of the two states) which prohibits the use of the (transient) probability measure as an amplitude measure, since less and less information about the distribution of amplitudes within the single $\mu$-trajectories is preserved. More precisely, after approximately 2.5 time units hardly any deviation from the steady state distribution is visible although almost all trajectories are supposed to oscillate as argued before.

Also other mathematical methods of analyzing the periodic behavior of a system, like the Laplace Transform, need a reliable and informative measure of the amplitude at each moment in time, rendering their usage in combination with the transient probability distribution as difficult in our setting.
ctmc

const l1 = 1;
const l2 = 1;
const double t;

module simple
c: [0..1] init 0;

[] (c=0) -> 11: (c'=1);
[] (c=1) -> 12: (c'=0);
endmodule

Figure 6: PRISM Model of a simple oscillator between the two states $c = 0$ and $c = 1$

Figure 7: Prism experiment: Probability of leaving state $s_0$ within $t$ time units.
Figure 8: Transient probabilities of being in state $c = 0$ respectively $c = 1$ at time $t$. 
4 Mathematical Definitions of Oscillatory, Periodic and Noisy Periodic Behavior

In the previous section, we have shown that the ODE approach, i.e., the deterministic solution, may fail when analyzing oscillatory behavior and a discrete stochastic based procedure should be taken (using MPMs in our setting). Instead of generating and analyzing single trajectories of the MPM under consideration, we will reason about its structure, in order to decide whether it oscillates or not and with which period length. The basic idea to do that kind of reasoning is to use CSL model checking, which not only allows us to argue about almost all possible \( \mu \)-trajectories at once, but also about their infinite behavior.

However, for this purpose we need a formal definition of when a system oscillates or not. This definition should also be resistant to noise, which is common to stochastic systems, like MPMs. Therefore, we will formally approach notions of oscillatory and periodic behavior by starting from their mathematical definitions and refine them towards a noise robust definition, which we will then be able to model check against almost all \( \mu \)-trajectories in an efficient way.

4.1 Oscillatory Behavior

We will start with the mathematical definition of oscillation tailored towards \( \mu \)-trajectories. When considering the limiting behavior, every function either

- converges (cf. Fig. 9, left),
- diverges (cf. Fig. 9, right),
- or oscillates (cf. Fig. 10), then – by definition – it does not diverge nor converge.

**Definition 14 (Oscillatory \( \mu \)-Trajectory).** A \( \mu \)-trajectory \( \tau \) of a MPM \( \mathcal{M} = (S, R, AP, L) \) with observable \( \mu \) is said to oscillate iff \( \not\exists c \in \mu(S). \lim_{t \to \infty} \tau(t) = c \) (Non-Convergence) and \( \lim_{t \to \infty} \tau(t) \neq \pm \infty \) (Non-Divergence).

![Figure 9: Examples of a convergent \( \mu \)-trajectory (left) and a divergent \( \mu \)-trajectory (right).](image-url)
4.2 Periodic Behavior

Now, we will present the notion of mathematical periodicity. Intuitively, a function that is periodic with period $\lambda$ shows some sort of pattern of $\lambda$ time units, that is repeated in a concatenated fashion without any interruption. Obvious examples of periodic functions are the sine and cosine functions which are $2\pi$-periodic.

**Definition 15** (Periodic $\mu$-Trajectory). A $\mu$-trajectory $\tau$ of a MPM $\mathcal{M} = (S, R, AP, L)$ with observable $\mu$ is said to be periodic with period $\lambda \in \mathbb{R}_{>0}$ iff for all $x \in \mathbb{R}_{\geq 0}$

$$\tau(x + \lambda) = \tau(x).$$

4.3 Noisy Periodic Behavior

Usually we want a system to show both, periodic as well as oscillatory behavior. None of the two properties is implied by the other. Counterexamples for both directions are illustrated in Figure 12. E.g. any random walk (cf. Fig. 12, top) satisfies the oscillation
conditions since it does not converge and any constant function $f : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}$ with $\forall x, y \in \mathbb{R}_{\geq 0}. f(x) = f(y)$ (cf. Fig. 12, bottom) satisfies the conditions for periodicity for any period $\lambda \in \mathbb{R}_{>0}$.

Hence, the main issue of mathematical oscillation is that it is too weak for our purposes since the period lengths are not constrained in any way. On the other hand, mathematical periodicity is too strong for our purposes. The reason is the nature of stochastic systems to show noise, which results in deviations in time w.r.t. both, amplitude and period, from a strictly periodic system. This invalidates the periodicity conditions for any non-pathological system (cf. Fig. 13). Also, there is no constraint enforcing the function not to show convergent behavior (cf. Fig. 12, bottom).

Therefore, we will now suggest a different approach, i.e. instead of enforcing the system to conform to those general and strict notion of oscillation and periodicity we relax their conditions in order to be able to deal with the noisy setting. We call that newly defined property, \textit{noisy periodicity}.

For noisy periodicity, first we will ensure \textit{non-convergence} as in the case of classical oscillation. Therefore, we demand that a noisy periodic $\mu$-trajectory shows significant deviations in amplitude over time. We achieve that by assuming a lower boundary $b_{\text{low}} \in \mathbb{R}$ and a higher boundary $b_{\text{high}} \in \mathbb{R}$ with $b_{\text{low}} < b_{\text{high}}$ that have to be crossed infinitely often. This ensures, that the trajectory can not forever stay in one of the intervals $I_{\text{low}} = (-\infty, b_{\text{low}})$, $I_{\text{mid}} = [b_{\text{low}}, b_{\text{high}})$ and $I_{\text{high}} = [b_{\text{high}}, \infty)$ (which form a
Figure 13: Example of a perfectly oscillatory and periodic $\mu$-trajectory (blue) and a noisy version of it (red).

partition of $\mathbb{R}$). Still, a trajectory could oscillate between the intervals $I_{\text{low}}$ and $I_{\text{mid}}$ or $I_{\text{mid}}$ and $I_{\text{high}}$ only. Consequently, we also demand that all three intervals will be crossed infinitely often. A problem arises if $|b_{\text{low}} - b_{\text{high}}| \leq L$ with

$$L = \max_{s,s' \in S,R(s,s') > 0} |\mu(s) - \mu(s')|$$

since then, the interval $I_{\text{mid}}$ could possibly be skipped. Consequently, we assume

$$|b_{\text{low}} - b_{\text{high}}| > L.$$  

The existence of $L$ is trivially justified, since we only consider finite systems. But even for infinite systems that minimum exists, since changes in population are finite and with the observable being a linear functional the observation scales linearly with the populations. The amplitude $A = |b_{\text{low}} - b_{\text{high}}|$ can be regarded as the minimum amplitude the signal shall oscillate with. Please note that by now, we do not impose any time constraint on the period length, i.e. the time needed to pass the intervals $I_{\text{low}}, I_{\text{mid}}, I_{\text{high}}, I_{\text{mid}}$ and finally $I_{\text{low}}$. Consequently, a trajectory is also noisy periodic even if the period lengths differ greatly from period to period. We will later constrain that time directly on the level of logical characterization. Adding timing constraints right now would make that definition too complex while adding these on the logical level is an intuitive and minor extension.

**Definition 16.** Given a MPM $\mathcal{M} = (S, R, AP, L)$ with observable $\mu$ and given bounds $b_{\text{low}}$ and $b_{\text{high}}$ with $|b_{\text{low}} - b_{\text{high}}| > L$ and the respective intervals $I_{\text{low}} = (-\infty, b_{\text{low}})$, $I_{\text{mid}} = [b_{\text{low}}, b_{\text{high}})$ and $I_{\text{high}} = [b_{\text{high}}, \infty)$, a $\mu$-trajectory $\tau$ is said to be noisy periodic iff for

$$\Psi_{\text{enter}}(t) = \begin{cases} \text{true} & \text{if } \lim_{t' \to t, t' < t} \tau(t') \in I_{\text{low}} \land \lim_{t' \to t, t' > t} \tau(t') \in I_{\text{mid}} \\ \text{false} & \text{otherwise} \end{cases}$$
\[ \exists t. \Psi_{\text{enter}}(t) \land \forall t. \Psi_{\text{enter}}(t) \Rightarrow \exists t_h > t. \tau(t_h) \in I_{\text{high}} \land \exists t_l > t_h. \tau(t_l) \in I_{\text{low}}. \]

The predicate \( \Phi_{\text{enter}}(t) \) is true iff a period may just have started, i.e. a transition from an \( I_{\text{low}} \) to an \( I_{\text{mid}} \) level has just been made. The first part of the conjunction therefore states that this shall happen infinitely often. The second part demands that whenever a period may have started, a point in time \( t_h \) will be reached, entering the \( I_{\text{high}} \) level (having crossed the \( b_{\text{high}} \) boundary), and then getting back to the \( I_{\text{low}} \) level (by crossing the \( b_{\text{low}} \) boundary) at some future time point \( t_l \). Since \( |b_{\text{low}} - b_{\text{high}}| > L \), the \( I_{\text{mid}} \) interval will be crossed in between. A sample noisy periodic trajectory is depicted in Figure 14. Please note, that \( \Phi_{\text{enter}} \) only characterizes points in time when a period may have started. For example in Figure 14, time point \( t_3 \) satisfies \( \Phi_{\text{enter}} \) although the period has already started at time \( t_2 \). Since by now we have not incorporated timing constraints on the period length, those false alarms do not harm. The reason is that the second part of the conjunction will nevertheless hold, since it already has to hold for a point in time before, e.g. for \( t_2 \). When later on adding the ability to reason about period lengths, we will have to carefully deal with that artifact, since otherwise we would also recognize periods that are not actually present, like from \( t_3 \) to \( t_2 \) in our example.

![Figure 14: Example of a noisy periodic \( \mu \)-trajectory.](image-url)
5 Logical Characterizations

We will now show, how the mathematical definition of the last Section can be expressed in temporal logics in order to enable verifying discrete stochastic systems against them via model checking.

5.1 Logical Characterization of Oscillation

**Non-Divergence:** The non-divergence property of oscillation can be split into two parts:

1. \( \lim_{t \to \infty} \tau(t) \neq \infty \) (non-divergence to \( \infty \)), and
2. \( \lim_{t \to \infty} \tau(t) \neq -\infty \) (non-divergence to \( -\infty \)).

Non-divergence to \( \infty \) can be reformulated as:

\[
\lim_{t \to \infty} \tau(t) \neq \infty \\
\iff \forall c. \exists t. \forall t' > t. \tau(t') > c \\
\iff \exists c. \forall t. \exists t' > t. \tau(t') \leq c.
\]

The same can be done for non-divergence to \( -\infty \):

\[
\lim_{t \to \infty} \tau(t) \neq -\infty \\
\iff \forall c. \exists t. \forall t' > t. \tau(t') < c \\
\iff \exists c. \forall t. \exists t' > t. \tau(t') \geq c.
\]

Given a specific lower bound \( c_{low} \) and an upper bound \( c_{high} \) as well as interpreting time points \( t \) as jump times, i.e. points in time at which a transition from one state to another is made, we can directly express the non-divergence property to hold for all \( \mu \)-trajectories in Linear Temporal Logic (LTL) [Pnu77] by the formulas:

1. \( \Box \Diamond \mu \leq c_{high} \) (non-divergence to \( \infty \)), and
2. \( \Box \Diamond \mu \geq c_{low} \) (non-divergence to \( -\infty \)).

Both formulas can also be expressed in Computation Tree Logic (CTL) [CE82] via

1. \( \forall \Box \forall \Diamond \mu \leq c_{high} \) (non-divergence to \( \infty \)), and
2. \( \forall \Box \forall \Diamond \mu \geq c_{low} \) (non-divergence to \( -\infty \)).

When switching to our probabilistic and continuous time setting, we have to restrict the non-divergence property to hold for *almost all* \( \mu \)-trajectories since not all possible sets of trajectories are actually measurable. For this purpose, we replace the all-quantifications \( \forall \) by the probability operator \( \mathcal{P}_{\geq 1} \). Moreover, we use unbounded temporal operators \( \Box \) and \( \Diamond \) that range over all real time points and are therefore not restricted to jump time anymore. Hence, non-divergence can be expressed as

1. \( \mathcal{P}_{\geq 1}[\Box \mathcal{P}_{\geq 1}[\Diamond (\mu \leq c_{high})]] \) (non-divergence to \( \infty \)), and
2. \( \mathcal{P}_{\geq 1}[\Box \mathcal{P}_{\geq 1}[\Diamond (\mu \geq c_{low})]] \) (non-divergence to \( -\infty \)).
Non-Convergence: The definition of convergence of a $\mu$-trajectory $\tau$, of a MPM with state space $S$, to a value $c$ is
\[ \forall \epsilon > 0. \exists t. \forall t' > t. |\tau(t') - c| < \epsilon. \]
Since $\tau$ is not continuous, this definition can then be reformulated:
\[ \forall \epsilon > 0. \exists t. \forall t' > t. |\tau(t') - c| < \epsilon \iff \forall c \in \mu(S). \forall t. \exists t' > t. \tau(t') = c. \]
For general non-convergence we will exploit that the co-domain of $\tau$ is a finite set $\mu(S)$ (since $S$ is finite) in order to simplify its definition:
\[ \forall c \in \mu(S). \lim_{t \to \infty} \tau(t) = c \iff \forall c \in \mu(S). \forall t. \exists t' > t. \tau(t') = c \iff \forall c \in \mu(S). \forall t. \exists t' > t. \tau(t') = c. \]
This property then can again be expressed as a LTL formula to hold for all $\mu$-trajectories by demanding non-convergence for all possible observations $c \in \mu(S)$:
\[ \bigwedge_{c \in \mu(S)} \Box \Diamond (\mu \neq c). \]
Translating this formula to CTL, we retrieve:
\[ \bigwedge_{c \in \mu(S)} \forall \Box \forall \Diamond (\mu \neq c). \]
The CSL-formula for non-convergence finally becomes
\[ \bigwedge_{c \in \mu(S)} P_{\geq 1}[\Box P_{\geq 1}[\Diamond (\mu \neq c)]] \tag{1} \]
Since for finite MPMs, the set $\mu(S)$ is finite as well, $\min \mu(S)$ and $\max \mu(S)$ exist and any $\mu$-trajectory $\tau$ will never fall below $\min \mu(S)$ or exceed $\max \mu(S)$. Consequently, it suffices to check formula 1 (cf. page 28) for the oscillation property because then the non-divergence property is trivially satisfied.

With the stochastic approach being adequate for lower population counts, the set of possible observations $\mu(S)$ is bounded and therefore, model checking of formula 1 (cf. page 28) is feasible. But still, for bigger systems with a potentially high diversity of observations, it might be necessary to optimize the model checking procedure for that property. Indeed, we will later on present possible optimization strategies in Section 6. The reason is that although the mathematical oscillation property has been shown not to be the ideal notion of oscillation for our setting, it might in certain cases still be important to know whether a system converges or not.
5.2 Logical Characterization of Periodicity

Although, we have already argued in Section 4.3, that the mathematical notion of periodicity is too strict for our setting, we will present a logical characterization of it for the sake of completeness, especially since it served as an inspiration for the logical characterization of noisy periodicity.

A $\mu$-trajectory $\tau$ is periodic with period length $\lambda > 0$ iff

$$\forall t. \tau(t + \lambda) = \tau(t).$$

Again, we exploit that the co-domain of $\tau$ is a finite set $\mu(S)$ for a MPM with finite state space $S$, in order to reformulate the above formula via case distinction on those possible observation $c \in \mu(S)$ at any time point $t$ to:

$$\forall t. \tau(t) = \tau(t + \lambda) \iff \forall t. \bigwedge_{c \in \mu(S)} (\tau(t) = c) \Rightarrow (\tau(t + \lambda) = c).$$

A direct translation of that property into LTL or CTL is impossible since they can not deal (without extension) with continuous time. Hence, the direct encoding of the periodicity condition to hold for almost all trajectories into CSL is

$$\mathcal{P}_{\geq 1}[\square \bigwedge_{c \in \mu(S)} ((\mu = c) \Rightarrow \mathcal{P}_{\geq 1}[\diamond^{\lambda}[\mu = c]])].$$

5.3 Logical Characterization of Noisy Periodicity

The logical characterization of noisy periodicity (Definition 16) has to incorporate the information about when a period may just have started and when a period has been completed. Since, later on in Section 6.4 we want to quantify the time needed for a period, we will now restrict the predicate $\Phi_{\text{enter}}(t)$ to hold only for those moments in time, where the interval $I_{\text{low}}$ has just been left for the $I_{\text{mid}}$ interval and no period has begun after the last period completion, i.e. a new period has started. Actually, this definition of noisy periodicity is equivalent to the original one (cf. Section 4.3) as we will see in Section 6.3.3.

From now on, we will assume that the states of our MPM under consideration have been additionally marked by the atomic propositions:

- $\text{start}$: with the last transition, a new period has started, and
- $\text{accept}$: the system had started a period, had crossed the $I_{\text{high}}$ level and has just reached the $I_{\text{high}}$ interval again, i.e. a full period has been completed.

We postpone how the actual labeling is achieved to Section 6.3.1. Noisy periodicity, i.e.

$$\exists t. \Psi_{\text{enter}}(t) \land \forall t. \Psi_{\text{enter}}(t) \Rightarrow \exists t_h > t. \tau(t_h) \in I_{\text{high}} \land \exists t > t_h. \tau(t) \in I_{\text{low}}$$
characterized within LTL to hold for all $\mu$-trajectories of a MPM (when considering jump times only) becomes

$$\exists t . \Psi_{\text{enter}}(t) \land \forall t . \Psi_{\text{enter}}(t) \Rightarrow \exists t_h > t . \tau(t_h) \in I_{\text{high}} \land \exists t_l > t_h . \tau(t_l) \in I_{\text{low}} .$$

This LTL formula is equivalent to the CTL formula

$$\forall \Box \forall \Diamond start \land \forall \Box (start \Rightarrow \forall \Diamond accept)$$

and can finally be expressed in CSL, to hold for almost all $\mu$-trajectories and continuous time, as

$$\mathcal{P}_{\geq 1}[\Box [\mathcal{P}_{\geq 1} \Diamond start]] \land \mathcal{P}_{\geq 1}[\Box (start \Rightarrow \mathcal{P}_{\geq 1} [\Diamond accept])].$$

(2)
6 Model Checking and Optimizations

We have treated two concepts to characterize oscillatory and periodic behavior suitable for stochastic systems – mathematical oscillation (cf. Section 4.1) and noisy periodicity (cf. Section 4.3), illustrated in Figure 15. Although, a logical characterization of oscillation has already been presented in Section 5.1, model checking this formula will turn out to possess quadratic time complexity. Therefore, we will discuss several methods of improving the model checking procedure for oscillatory behavior. Moreover, we will present the product MPM construction needed for model checking noisy periodicity as well as ways to quantify the period length for noisy periodic systems.

6.1 Optimizing Model Checking of Oscillatory Behavior

For finite systems, only the non-convergence property

\[ \bigwedge_{c \in \mu(S)} P_{\geq 1}[\Box P_{\geq 1}[\diamond (\mu \neq c)]] \]
has to be checked explicitly in order to discover whether the system oscillates or not. Unfortunately, according to [BH03], model checking the above formula for a MPM $\mathcal{M} = (S, R, AP, L)$ with observable $\mu$ has time complexity $O(|\mu(S)| \cdot |S|)$, assuming the rate matrix $R$ is sparse. This assumption is justified for most MPMs since there are typically only a constant number of changes in the populations possible at any moment in time. E.g. for biological systems, the maximum fanout, i.e. the number of successor states, for each state is bounded by the number of reaction types. But nevertheless, since there usually is a high diversity in observations, i.e. $|\mu(S)| \in O(|S|)$, the overall model checking time complexity lies in $O(|S|^2)$.

For improving the model checking procedure for oscillatory behavior we will exploit the structure of $\mu$-trajectories since they are piecewise constant between two jump points of the underlying CTMC. Consequently, instead of directly model checking formula 1 (cf. page 28), we might want to check whether the system will infinitely often follow a transition that causes the observation to vary. For this, we have to ensure that:

1. The system can not reach a state $s$ with $R(s, s') = 0$ for all states $s'$, because otherwise the observation would definitely never change anymore and the oscillation property would be violated. Such a state $s$ with the aforementioned property is called absorbing.

2. Let $\Delta \mu$ denote the change in observation caused by the last transition in the MPM. Then we also have to ensure that whatever state the system reaches, there must eventually be change in observation, i.e. $\Delta \mu \neq 0$ (cf. Fig. 16).

Please note that although the absence of reachable absorbing states guarantees that infinitely many transitions are made, the second part of the property is still needed. The reason is that transitions might also change populations not represented within the observable. So, if all states are marked with the change in observation of the last transition ($\Delta \mu = x$) and absorbing states are additionally labeled with the atomic proposition absorbing, we can reformulate the non-convergence condition of oscillation via the CSL-formula

$$P_{\geq 1}[\Box(\neg \text{absorbing} \land P_{\geq 1}[\Diamond \neg(\Delta \mu = 0)])].$$

(3)

6.1.1 Delta Observation MPM Construction

Now, we will show how a MPM can be expanded to include the $\Delta \mu$ and absorbing labels while preserving the validity/invalidity of all CSL$_c$ formulae over the original set of atomic propositions. In the following, we will also write

$$s \xrightarrow{\lambda} R s'$$

instead of $R(s, s') = \lambda > 0$ for states $s$ and $s'$ and rate matrix $R$.

**Definition 17** (Delta Observation MPM). For a MPM $\mathcal{M} = (S, R, AP, L)$ with observable $\mu$ and initial distribution $\alpha$, the Delta Observation MPM is defined to be the MPM $\mathcal{M}^\Delta = (S', R', AP', L')$ with observable $\mu^\Delta$:
Figure 16: A $\mu$-trajectory with the difference in observation $\Delta \mu$ plotted at jump points.

- $S' = \{(s, \delta) \mid s \in S, \delta = 0 \lor \exists s' \in S.R(s', s) > 0 \land \mu(s') - \mu(s) = \delta\} \subset S \times \mathbb{R}$
- $AP' = AP \cup \{\mu'\delta' = \delta'\mid \delta' \in \mathbb{R}\} \cup \{\text{absorbing}\}$
- $\forall (s, \delta) \in S'$:
  $L'(s, \delta) = L(s) \cup \{\mu'\delta' = \delta'\} \cup \begin{cases} \emptyset & \text{if } \exists s' \in S'.R((s, \delta), s') \neq 0 \\ \{\text{absorbing}\} & \text{otherwise} \end{cases}$
- $R'$ as the minimal rate matrix satisfying the inference rule

$$s \xrightarrow{\lambda} s' \quad \mu(s') - \mu(s) = \delta'$$

\[\begin{array}{c}
(s, \delta) \xrightarrow{\lambda} (s', \delta')
\end{array}\]

i.e. all other rates are zero.

- $\mu^\Delta(s, \delta) = \mu(s)$

and the corresponding new initial distribution $\alpha'$ with

$$\alpha^\Delta(s, \delta) = \begin{cases} 
\alpha(s) & \text{if } \delta = 0 \\
0 & \text{otherwise}
\end{cases}$$

We assume $\{\mu'\delta' = x' \mid x' \in \mathbb{R}\} \cap AP = \emptyset$, which can be achieved via renaming.
Example 4 (Delta Observation MPM). Assume a MPM $\mathcal{M} = (S, R, AP, L)$ with observable $\mu$ like depicted in Figure 17 (left) for which the delta observation MPM $\mathcal{M}^\Delta = (S', R', AP', L')$ with observable $\mu^\Delta$ is built (cf. Fig. 17, right). The reachable states of $\mathcal{M}$ are

$$S_r = \{A, B, C, D, E\} \subset \mathbb{N}_0^N$$

for some $N \in \mathbb{N}$, respectively the reachable state space of $\mathcal{M}^\Delta$ is

$$S'_r = \{(A, -1), (A, 0), (B, 1), (C, 1), (C, 2), (D, 0), (D, 2), (E, 0)\} \subset S \times \mathbb{N}_0^N$$

The rates are depicted in Figure 17 and the labels and observations are:

<table>
<thead>
<tr>
<th>$s \in S_r$</th>
<th>$L(s)$</th>
<th>$\mu(s)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$\emptyset$</td>
<td>0</td>
</tr>
<tr>
<td>$B$</td>
<td>$\emptyset$</td>
<td>1</td>
</tr>
<tr>
<td>$C$</td>
<td>$\emptyset$</td>
<td>2</td>
</tr>
<tr>
<td>$D$</td>
<td>$\emptyset$</td>
<td>3</td>
</tr>
<tr>
<td>$E$</td>
<td>$\emptyset$</td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$s' \in S'_r$</th>
<th>$L'(s')$</th>
<th>$\mu^\Delta(s')$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(A, -1)$</td>
<td>${\Delta \mu = -1}$</td>
<td>0</td>
</tr>
<tr>
<td>$(A, 0)$</td>
<td>${\Delta \mu = 0}$</td>
<td>0</td>
</tr>
<tr>
<td>$(B, 1)$</td>
<td>${\Delta \mu = 1}$</td>
<td>1</td>
</tr>
<tr>
<td>$(C, 1)$</td>
<td>${\Delta \mu = 1, \text{absorbing}}$</td>
<td>2</td>
</tr>
<tr>
<td>$(C, 2)$</td>
<td>${\Delta \mu = 2, \text{absorbing}}$</td>
<td>2</td>
</tr>
<tr>
<td>$(D, 0)$</td>
<td>${\Delta \mu = 0}$</td>
<td>3</td>
</tr>
<tr>
<td>$(D, 2)$</td>
<td>${\Delta \mu = 2}$</td>
<td>3</td>
</tr>
<tr>
<td>$(E, 0)$</td>
<td>${\Delta \mu = 0}$</td>
<td>3</td>
</tr>
</tbody>
</table>

In this example it is possible to reach states $(C, 1)$ and $(C, 2)$ which are absorbing as well as a cycle $(E, 0) \leftrightarrow (D, 0)$ (blue dotted rectangle) that can not be left and leads to no change in observable, both with non-zero probability. Therefore, the oscillation property, i.e. Formula 3 (cf. page 32), is violated.

6.1.2 Model Checking Complexity

Assuming a MPM $\mathcal{M} = (S, R, AP, L)$ with observable $\mu$ and sparse rate matrix $R$ with maximum fanout

$$\max_{s \in S} \{|\{(s', \delta') \in S' \mid R(s, s') > 0\}| \in \mathcal{O}(1)$$

(which is a reasonable assumption like argued before). Consequently, the corresponding delta observation MPM $\mathcal{M}^\Delta = (S', R', AP')$ can be constructed with time complexity $\mathcal{O}(|S|)$. The reason is, that for the new set of states $S'$ holds

$$|S'| = \{|(s, \delta) \mid s \in S, \delta = 0 \lor \exists s' \in S. R(s', s) > 0 \land \mu(s') - \mu(s) = \delta\| \leq (\max_{s \in S} + 1) \cdot |S|.$$ 

Since $\max_{s \in S} \in \mathcal{O}(1)$, we retrieve $|S'| \in \mathcal{O}(|S|)$. Finally, with

$$\forall (s, \delta) \in S', |\{(s', \delta') \in S' \mid R'((s, \delta), (s', \delta')) > 0\}| = |\{s' \in S \mid R(s, s')\}|$$
and therefore

$$max_{FO}(S') = max_{FO}(S)$$

for each of the $|S'| \in \mathcal{O}(S)$ states in the delta observation MPM, only a maximum of $max_{FO}(S') \in \mathcal{O}(1)$ rates are non-zero inside the rate matrix $R'$. Moreover, model checking of CSL formula

$$P_{\geq 1}[\Box(\neg absorbing \land P_{\geq 1}[\Diamond(\neg (\Delta \mu = 0))])$$

is in $\mathcal{O}(S') = \mathcal{O}(S)$ according to the above reasoning and [BH03]. To sum up, the time complexity of model checking oscillation via formula 3 (cf. page 32) on MPMs with state space $S$ lies in $\mathcal{O}(|S|)$.

6.1.3 Correctness

Obviously, by construction, each state of the delta observation MPM is labeled by the change in observation that occurred during the last transition. In order to show that the presented approach to judge on the oscillatory behavior of a system is correct, we have to show that

- model checking Formula 1 (cf. page 28) on the original system is equivalent to model checking Formula 3 (cf. page 32) on the delta expanded MPM (cf. Fig. 18), as well as that
- the overall behavior is preserved by the construction.

Figure 17: Example of MPM delta observation expansion.
Oscillation for MPM $\mathcal{M}$

(を求め Definition 14)

\[
\downarrow
LTL \text{ formula} \quad \iff \quad LTL \text{ formula}
\]

\[
\downarrow
\text{CTL formula} \quad \iff \quad \text{CTL formula}
\]

\[
\downarrow
\text{CSL formula} \quad \iff \quad \text{CSL formula}
\]

Oscillation for delta observation MPM $\mathcal{M}^\Delta$

(を求め Definition 6.1.1)

\[
\downarrow
LTL \text{ formula}
\]

\[
\downarrow
\text{CTL formula}
\]

\[
\downarrow
\text{CSL formula}
\]

Figure 18: Correspondence of the presented logical characterizations of oscillation.

**Correspondence of the Logical Characterizations:** Since we are abstracting from time, we observe the system solely at jump times of the system and therefore only need to consider the sequence of visited states. Any trajectory is based on a path

\[
\sigma = s_0 t_0 s_1 t_1 \ldots
\]

of the underlying CTMC. We can project that path onto the sequence of visited states, i.e.

\[
\sigma_s = s_0 s_1 \ldots.
\]

Throughout this thesis, let $\sigma_s[i] = s_i$ denote the $(i+1)st$ state of a sequence of states. When deriving the CSL characterization of non-convergence from its mathematical definition in Section 5.1, we used an LTL characterization as an intermediate step. In order to show that the two logical characterizations of non-convergence coincide we have to show that any trajectory of the original system satisfies Formula 1 (を求め page 28) iff its corresponding trajectory in the expanded system satisfies Formula 3 (求め page 32). Proposition 3 states exactly that. Please note that any trajectory in the original system indeed maps to a uniquely defined trajectory in the expanded system and vice versa because the construction of Section 6.1.1 is deterministic. The reason is that the change in observation is uniquely defined for each combination of state and successor state. More precisely, for some sequence of states $\sigma_s$ of the original MPM as defined above we retrieve the sequence

\[
\sigma_s^\Delta = (s_0, 0) (s_1, \mu(s_1) - \mu(s_0)) (s_2, \mu(s_2) - \mu(s_1)) \ldots
\]

in the delta observation expanded MPM. In the following, we will denote the LTL satisfaction relation between sequences of states and LTL formulas as $\models_{LTL}$ [Pnu77].

**Proposition 3.** For any sequence $\sigma_s$ of states in a MPM $\mathcal{M} = (S, R, AP, L)$ with observable $\mu$,

\[
\sigma_s \models_{LTL} \bigwedge_{c \in \mu(S)} \Box \Diamond \neg (\mu = c)
\]

36
holds iff for the corresponding sequence of states $\sigma_s^\Delta$ in the delta observation MPM $M^\Delta$,

$$\sigma_s^\Delta \models_{\text{LTL}} \Box (\neg \text{absorbing} \land \Diamond \neg (\Delta \mu = 0))$$

holds.

\textbf{Proof.} Postponed to Appendix B.1. \hfill \square

From the LTL formula

$$\Box (\neg \text{absorbing} \land \Diamond \neg (\Delta \mu = 0))$$

we can derive the equivalent CTL formula

$$\forall \Box (\neg \text{absorbing} \land \forall \Diamond \neg (\Delta \mu = 0))$$

which then corresponds to (CSL) Formula 3 (cf. page 32), i.e.

$$\mathcal{P}_{\geq 1}[\Box (\neg \text{absorbing} \land \mathcal{P}_{\geq 1}[\Diamond \neg (\Delta \mu = 0)])].$$

\textbf{Preservation of the Overall Behavior:} We will now show, that the behavior of the original MPM is preserved.

\textbf{Lemma 1.} For a MPM $M = (S, R, AP, L)$ and its delta observation MPM $M^\Delta = (S', R', AP', L')$ with observations $\mu$, $\mu^\Delta$ and initial distributions $\alpha$ respectively $\alpha^\Delta$ holds

$$M \sim_{AP, \mu \cup \mu^\Delta} M^\Delta$$

i.e. the original system and its delta observation expansion are $AP, \mu \cup \mu^\Delta$- bisimilar (cf. Section 2.2.2).

\textbf{Proof.} Postponed to Appendix B.2 \hfill \square

\textbf{Proposition 4} (Delta Observation MPM CSLc Preservation). A MPM $M = (S, R, AP, L)$ and its delta observation MPM $M^\Delta$ are CSLc$|_{AP}$ equivalent, i.e. they satisfy exactly the same CSLc formulae restricted on the original set of atomic propositions $AP$.

\textbf{Proof.} The Proposition follows from Lemma 1 and Proposition 1. \hfill \square

\section{6.2 Further Improvements on Model Checking Oscillations}

Looking closer at the previous construction and the model checking procedure for oscillation, only the fact whether or not a change in observation happened during a transition really needs to be checked. We can therefore optimize model checking Formula 1 (cf. page 28) even further by only extending the given MPM by a Boolean predicate specifying exactly this. Consequently, we define the predicate $p_{\text{change}}$ that is used to detect whether a change in observation has been made as

$$p_{\text{change}}(s, s') = (\mu(s) \neq \mu(s')).$$
Now, for checking, whether the oscillation property holds for a system $M$, instead of verifying $P \geq 1$ on the delta observation expanded system, we will verify the CSL formula $P \geq 1[\lnot \text{absorbing} \land P \geq 1[\lnot (\Delta \mu = 0)]_{\text{p\_change}}]$ on the Predicate Expanded MPM $M_{\text{p\_change}}$.

### 6.2.1 MPM Predicate Expansion

First, we will describe the general methodology of extending a MPM by a predicate on the transition previously taken at each state. In the following, let $\mathbb{B} = \{\text{true, false}\}$ denote the set of boolean values.

**Definition 18 (MPM Predicate Expansion).** For a MPM $M = (S, R, AP, L)$ with observable $\mu$, initial distribution $\alpha$ and a predicate $p : S \times S \rightarrow \mathbb{B}$, we define the Predicate Expanded MPM as the MPM $M_p = (S', R', AP', L')$ with observable $\mu_p$, where:

- $S' = S \times \mathbb{B}$
- $AP' = AP \cup \{p, \lnot p\} \cup \{\text{absorbing}\}$
- $\forall s \in S', b \in \mathbb{B}:
  \begin{align*}
  L'(s, false) &= L(s) \cup \{p\} \cup \text{absorbing}(s) \\
  L'(s, true) &= L(s) \cup \{\lnot p\} \cup \text{absorbing}(s)
  \end{align*}$
  with $\text{absorbing}(s) = \begin{cases} \emptyset & \text{if } \exists s' \in S.R(s, s') > 0 \\ \{\text{absorbing}\} & \text{otherwise} \end{cases}$
- $R'$ as the minimal rate matrix satisfying the inference rules
  \begin{align*}
  s \xrightarrow{\lambda} R s' & \quad p(s, s') = \text{false} \quad b \in \mathbb{B} \\
  (s, b) \xrightarrow{\lambda_{R'}} (s', \text{false}) & \\
  s \xrightarrow{\lambda} R s' & \quad p(s, s') = \text{true} \quad b \in \mathbb{B} \\
  (s, b) \xrightarrow{\lambda_{R'}} (s', \text{true})
  \end{align*}
  i.e. other rates are zero.
- $\mu_p(s, false) = \mu_p(s, true) = \mu(s)$

and the corresponding new initial distribution $\alpha'$ is defined as

- $\alpha'(s, b) = \begin{cases} \alpha(s) & \text{if } b = \text{false} \\ 0 & \text{otherwise} \end{cases}$

We assume, $\{p, \lnot p\} \nsubseteq AP$, which can be enforced via renaming.
6.2.2 Model Checking Complexity

For a given MPM $M = (S, R, AP, L)$ and predicate $p$, let the corresponding predicate expanded MPM be $M^p = (S', R', AP', L')$. By definition, $S' = S \times \mathbb{B}$ and therefore $|S'| = |S| \cdot |\mathbb{B}| = 2 \cdot |S|$. The size of the reachable state space of $M^p$ might be even less. Since for each state $(s, b) \in S'$ and each rate $R(s, s')$ exactly one of the two inference rules of the construction in Section 6.2.1 is applicable, the number of non-zero entries in $R'$ only grows by the magnitude of the state space, i.e. by a factor of two and even less if only the reachable state space of $M^p$ is considered. Consequently, assuming a sparse rate matrix $R$, the time complexity of generating the predicate expanded MPM lies in $O(2 \cdot |S|) = O(|S|)$. Since the model checking time complexity of Formula 4 (cf. page 38) is linear, also the total complexity of model checking oscillation lies in $O(|S|)$.

The advantage w.r.t. the method proposed in Section 6.1.1 is, that the growth in size is not only bounded by the fanout, which varies from model to model, but even by a constant factor of two for all models.

6.2.3 Correctness

Via the construction described in Section 6.2.1 each state of the predicate expanded MPM is labeled with $p$ if the respective predicate holds due to the incoming transition and with $\neg p$ otherwise. Since this construction for the specific change predicate $p_{\text{change}}$ is similar to the delta observation expansion, an analog reasoning for showing the correspondence between formulas 1 (cf. page 28) and 4 (cf. page 38) can be used. For universal applicability, we will now explicitly show that the general predicate expansion preserves the behavior of the original system.

**Lemma 2.** For a MPM $M = (S, R, AP, L)$ and its Predicate Expanded MPM $M^p = (S', R', AP', L')$ with predicate $p$, observables $\mu, \mu^p$ and initial distributions $\alpha$ and $\alpha'$ holds

$$M \sim_{AP, \mu \cup \mu^p} M^p.$$ 

**Proof.** Postponed to Appendix B.3.

**Proposition 5** (MPM Predicate Expansion CSL Preservation). A MPM $M = (S, R, AP, L)$ and its Predicate Expanded MPM $M^p$ satisfy exactly the same CSL$_c$ formulae restricted to the original set of atomic propositions $AP$ for any predicate $p : S \times S \to \mathbb{B}$.

**Proof.** The proposition follows from Lemma 2 and Proposition 1.

6.3 Model Checking Noisy Periodicity

We will now show, how to allow for model checking noisy periodicity according to Formula 2 (cf. page 30). An important observation is, that the actual behavior, a noisy
periodic trajectory (Definition 16) must show (when starting in a state whose observation lies in the interval \( I_{low} \)), is an interval crossing corresponding to the \( \omega \)-regular expression \( E_{np} = E_{\omega_{\text{period}}} \) with

\[
E_{\text{period}} = I_{low}(I_{low}^*I_{mid}^*)^*I_{mid}(I_{mid}^*I_{high}^*)^*I_{high}(I_{high}^*I_{mid}^*)^*I_{mid}(I_{mid}^*I_{low}^*)^*
\]

where the unique period starting points (red) and completions of periods (blue) can be recognized.

### 6.3.1 Period Detector Expansion

This insight suggests the use of an automata theory based methodology in order to detect completions of full periods, i.e. of individual traversals according to the regular expression \( E_{\text{period}} \). Thus, we will compose the system with a finite state automaton \( D \) recognizing exactly the language \( L(E_{np}) \), called Period Detector, resulting in a product \( MPM \otimes D \). This procedure has been inspired by model checking algorithms for pathCSL [BCH+03] and asCSL [BCKS07], extensions to standard CSL, where the Until operator is replaced by a more general regular-expression-based path operator.

The logical characterization of noisy periodicity in Section 5.3 assumes the start and end states of periods to be marked as start and accept respectively. Consequently, the period detector \( D \) should incorporate those labelings for the appropriate states as well. Furthermore, the automaton \( D \) must be deterministic, since a product automaton of a non-deterministic automaton and a MPM does not necessarily preserve the probabilistic and timed behavior of the original MPM, i.e. outgoing transitions could be duplicated, leading to deviations in the total exit rates, altering the residence times and changing the successor state probability distributions.

**Definition 19** (Labeled Deterministic Finite State Automaton). A **labeled Deterministic Finite State Automaton (LDFA)** is a tuple \((Q, \Sigma, \rightarrow, \text{lab}, L)\), where

- \( Q \) is the finite set of states
- \( \Sigma \) is the alphabet
- \( \rightarrow \in Q \times \Sigma \times Q \) is the (deterministic) transition relation, i.e. whenever \( q \xrightarrow{\sigma} q' \) and \( q \xrightarrow{\sigma} q'' \) then \( q' = q'' \) for any combination of states \( q, q', q'' \in Q \) and label \( \sigma \in \Sigma \).
- \( \text{lab} \) is the set of labels
- \( L : S \rightarrow 2^{\text{lab}} \) is the labeling function

A LDFA is non-blocking iff for any state \( q \in Q \) and label \( \sigma \in \Sigma \) there is a state \( q' \in Q \) s.t. \( q \xrightarrow{\sigma} q' \).
Definition 20 (Period Detector). For given boundaries \( b_{\text{low}} \) and \( b_{\text{high}} \), inducing the intervals \( I_{\text{low}} = (-\infty, b_{\text{low}}) \), \( I_{\text{mid}} = [b_{\text{low}}, b_{\text{high}}) \) and \( I_{\text{high}} = [b_{\text{high}}, \infty) \), we define the Period Detector as the LDFA

\[
D = (\{q_0, q_0', q_0'', q_1, q_1', q_2, q_3, q_E\}, \{I_{\text{low}}, I_{\text{mid}}, I_{\text{high}}\}, \rightarrow, \{\text{start, accept, error}\}, L)
\]

where \( \rightarrow \) and \( L \) are given in Figure 19.

In order to understand the behavior of the period detector, we will first have a look at the supposedly simpler LDFA depicted in Figure 20. At first glance, the cycles in between the states \( q_0, q_1, q_2 \) and \( q_3 \) coincide with the desired behavior \( \mathcal{L}(E_{np}) \), i.e. any deviation from it will result in ending up in the absorbing error state \( q_E \), labeled with \text{error}. Please note that since no transition leaves \( q_E \), any run that infinitely often visits \text{start} and \text{accept} states can never enter the \( q_E \) state. Still, three problems exist within that approach to define the period detector with their solutions having been incorporated into the correct LDFA depicted in Figure 19:

- There is an \( I_{\text{low}} \) transition between \( q_1 \) and \( q_0 \) wrongly recognizing crossings of solely the \( b_{\text{low}} \) boundary as full periods. Therefore, in the correct period detector, state \( q_0'' \) is needed that is reached by an \( I_{\text{low}} \) transition, mimicking the behavior of \( q_0 \) but not being labeled with \text{accept}.

- The \( I_{\text{low}} \) self-loop of state \( q_0 \) makes accepting a period ambiguous. This is especially problematic when later on measuring the \text{inter period time}, i.e. the time between ending a period (accept) and starting a new one (start). Consequently, state \( q_0 \)
has to be split up further into $q_0$ and $q'_0$, where $q'_0$ mimics the behavior of $q_0$ but is not labeled with accept.

- Finally, also the $I_{mid}$ self-loop of state $q_1$ incorrectly classifies time points as starting times of a period. Again, a splitting of state $q_1$ into states $q_1$ and $q'_1$ makes $q_1$ the unique start state of a period.

![Figure 20: Incorrect smaller Period Detector LDFA (problems colored red).](image)

Now, we will show how to compose a MPM $\mathcal{M}$ under consideration with a period detector LDFA $\mathcal{D}$. For this we will use an adapted version of standard product automaton construction to retrieve a product automaton $\mathcal{M} \otimes \mathcal{D}$, in which each state will contain information about the current state of the original MPM and of the period detector, the system is in. Hence, the states of $\mathcal{M} \otimes \mathcal{D}$ will be labeled by both, the labels of the MPM and the period detector.

**Definition 21 (Product MPM).** Let $\mathcal{M} = (S, R, AP, L)$ be a MPM with observable $\mu$ and initial distribution $\alpha$. Further, let

$$\mathcal{D} = (\{q_0, q'_0, q''_0, q_1, q'_1, q_2, q_3, q_E\}, \{I_{low}, I_{mid}, I_{high}\}, \rightarrow, \{\text{start, accept, error}\}, L_{pd})$$

be a period detector LDFA. Then, the Product MPM $\mathcal{M} \otimes \mathcal{D}$ with observable $\mu^{\mathcal{M} \otimes \mathcal{D}}$ and initial distribution $\alpha^{\mathcal{M} \otimes \mathcal{D}}$ is defined as

$$\mathcal{M} \otimes \mathcal{D} = (S', R', AP', L')$$

where:
\[ S' = S \times Q \]
\[ AP' = AP \cup \text{lab} \]
\[ L : S \times Q \rightarrow 2^{AP'} \text{ is given by } L(s,q) = L(s) \cup L_{pd}(q) \]
\[ R' \text{ is the minimal rate matrix satisfying the inference rule } \]
\[
\begin{array}{c c c c}
  s & \xrightarrow{\lambda_R} & s' & q \xrightarrow{I} q' & \mu(s') \in I \\
  (s,q) & \xrightarrow{\lambda_{R'}} & (s',q')
\end{array}
\]
\[ \text{i.e. all other rates are zero.} \]

The initial distribution \( \alpha' \) is defined as:
\[
\alpha^{M \otimes D}(s,q) = \begin{cases} 
  \alpha(s) & \text{if } \mu(s) = I_{low} \land q = q_0 \\
  \alpha(s) & \text{if } \mu(s) = I_{mid} \land q = q'_1 \\
  \alpha(s) & \text{if } \mu(s) = I_{high} \land q = q_2 \\
  0 & \text{else}
\end{cases}
\]

### 6.3.2 Model Checking Complexity

Given a MPM \( \mathcal{M} = (S,R,AP,L) \) with sparse rate matrix \( R \) and its product MPM \( \mathcal{M} \otimes D = (S',R',AP',L') \) for some period detector LDFA \( \mathcal{D} = (Q,I,\rightarrow,\text{lab},L_{pd}) \), the state space grows from \( |S| \) to \( |S'| = |S| \cdot |Q| = 8 \cdot |S| \), i.e. by a constant factor of 8. Please note that the growth of the reachable state space will usually be much smaller for this construction as well. Since any period detector LDFA is non-blocking and deterministic, exactly one transition from \( (s,q) \) to \( (s',q') \) (for some \( q' \in Q \)) with \( R'((s,q),(s',q')) = r > 0 \) will be generated in the product MPM for each state \( (s,q) \in S' \) and each outgoing transition of the original MPM \( \mathcal{M} \) to some state \( s' \in S \) with \( R(s,s') = r > 0 \). Hence, the rate matrix also grows by the constant factor of \( |Q| = 8 \). Consequently, the product MPM can be built with time complexity \( O(|S| \cdot |Q|) = O(8 \cdot |S|) = O(|S|) \). Also the growth of the reachable part of the rate matrix is significantly smaller in most cases. Model checking CSL Formula 2 (cf. page 30) has time complexity \( O(|S'|) = O(|S|) \). To sum up, model checking noisy periodicity on MPM \( \mathcal{M} \) also has total time complexity \( O(|S|) \).

### 6.3.3 Correctness

In order to show that the period detector based approach of detecting noisy periodicity is correct we have to show that

- almost all \( \mu \)-trajectories of the original MPM are noisy periodic according to Definition 16 iff CSL Formula 2 (cf. page 30) is satisfied by the product MPM (cf. Fig. 21).
• during the product MPM construction, the overall behavior, including the state labelings of the original system, is preserved.

\[
\text{Noisy Periodicity for product MPM } \mathcal{M} \otimes \mathcal{D} \\
(\text{cf. Section 6.3.1})
\]

\[
\downarrow
\]

\[
\text{Noisy Periodicity for MPM } \mathcal{M} \iff \text{LTL formula}
\]

\[
\boxdot \text{start} \land \Box (\text{start} \Rightarrow \Diamond \text{accept}) \\
\downarrow
\]

\[
\text{CTL formula}
\]

\[
\forall \Box \forall \text{start} \land \forall (\text{start} \Rightarrow \forall \Diamond \text{accept}) \\
\downarrow
\]

\[
\text{CSL formula}
\]

\[
\forall \exists \geq 1 \left[\forall \exists \geq 1 [\text{start}] \right] \\
\land \forall \exists \geq 1 [\Box (\text{start} \Rightarrow \forall \exists \geq 1 [\Diamond \text{accept}])] \\
\]

Figure 21: Correspondence of the presented logical characterizations of oscillation.

**Correspondence between mathematical definition and CSL formula:** We are using a slightly modified version of product automata generation, where the states of the resulting MPM are additionally labeled with the period detector’s labels and therefore assume this construction to be correct. Now, we are left to show that for any µ-trajectory τ of a MPM M, the corresponding run r in the period detector D satisfies the LTL formula \(\Box \Diamond \text{start} \land \Box (\text{start} \Rightarrow \Diamond \text{accept})\) and vice versa, since this LTL formula was the starting point for deducing a logical characterization for noisy periodicity in Section 5.3.

We assume that we are given a MPM \(M = (S, R, AP, L)\) with observable µ. Furthermore, we demand that for the boundary values \(b_{low}\) and \(b_{high}\), we are using for the analysis of \(M\) w.r.t. noisy periodic behavior, the following holds

\[
|b_{low} - b_{high}| > L = \max_{s,s' \in S, R(s,s') \neq 0} |\mu(s) - \mu(s')|.
\]

Consequently, we retrieve the intervals \(I_{low} = (-\infty, b_{low})\), \(I_{mid} = [b_{low}, b_{high})\) and \(I_{high} = [b_{high}, \infty)\). Now, given a µ-trajectory of \(M\), we know that this trajectory has been generated by a path

\[
\sigma = s_0 \ t_0 \ s_1 \ t_1 \ \ldots
\]

on the underlying CTMC. When abstracting from time, i.e. considering jump times of the underlying CTMC only, we can project this path onto the sequence

\[
\sigma_s = s_0 \ s_1 \ \ldots
\]

of visited states entered at jump times of \(\tau\) for reasoning about the noisy periodic behavior. The reason is, that Definition 16 classifies µ-trajectories solely on the changes
in the observable $\mu$ which can only happen at transitions between states. Now, let

$$I(s) = \begin{cases} 
I_{\text{low}} & \text{if } \mu(s) \in I_{\text{low}}, \\
I_{\text{mid}} & \text{if } \mu(s) \in I_{\text{mid}}, \\
I_{\text{high}} & \text{otherwise}
\end{cases}$$

for any $s \in S$ denote the interval a state’s observation lies in. Then for any $\mu$-trajectory $\tau$ with underlying sequence of visited states $\sigma_I = s_0 s_1 \ldots$ we define the sequence of visited intervals $\sigma_I$ as

$$\sigma_I = I(s_0) I(s_1) \ldots$$

Consequently, for a $\mu$-trajectory to be noisy periodic according to Definition 16, it suffices to show that for $\sigma_I$ (its sequence of intervals crossed), Definition 22 holds.

**Definition 22** (Noisy Periodic Interval Sequence). A sequence of intervals $\sigma_I = I_0 I_1 \ldots$ is said to be noisy periodic for boundary values $b_{\text{low}}$ and $b_{\text{high}}$ iff

$$\exists t \in \mathbb{N}_0. \Phi_{\text{enter}}(t) \land \forall t' \in \mathbb{N}_0. \Phi_{\text{enter}}(t') \Rightarrow \exists t_h > t. \sigma_I[t_h] = I_{\text{high}} \land \exists t_l > t_h. \sigma_I[t_h] = I_{\text{low}}$$

with

$$I_{\text{low}} = (-\infty, b_{\text{low}}), I_{\text{mid}} = [b_{\text{low}}, b_{\text{high}}), I_{\text{high}} = [b_{\text{high}}, \infty)$$

and

$$\Phi_{\text{enter}}(t) = \sigma_I[t] = I_{\text{low}} \land \sigma_I[t + 1] = I_{\text{mid}}.$$

Since any period detector LDFA $D = \{Q, \{I_{\text{low}}, I_{\text{mid}}, I_{\text{high}}\}, \rightarrow, \text{lab}, L\}$ is defined to be deterministic and non-blocking, $\sigma_I$ interpreted as a word of $D$, corresponds to a unique run, i.e. a sequence $r = r_1 r_2 \ldots$ with $r_i \in Q$ of visited states of $D$.

Finally, Lemma 3 and Proposition 6 connect the mathematical definition of noisy periodicity with its logical characterization.

**Lemma 3.** A $\mu$-trajectory $\tau$ of a MPM $M = (S, R, AP, L)$ with a finite sequence of visited intervals $\sigma_I$ does not satisfy Definition 22 nor does for its run $r$ in the period detector $D = \{Q, \{I_{\text{low}}, I_{\text{mid}}, I_{\text{high}}\}, \rightarrow, \text{lab}, L\}$ with $I_{\text{low}} = (-\infty, b_{\text{low}}), I_{\text{mid}} = [b_{\text{low}}, b_{\text{high}}), I_{\text{high}} = [b_{\text{high}}, \infty)$ and $|b_{\text{low}} - b_{\text{high}}| > L$ hold

$$r \models_{\text{LTL}} \square \diamond \text{start} \land \square (\text{start} \Rightarrow \diamond \text{accept}).$$

**Proof.** Postponed to Appendix B.4.

**Proposition 6.** A $\mu$-trajectory $\tau$ of a MPM $M = (S, R, AP, L)$ with sequence of visited intervals $\sigma_I$ satisfies Definition 22 iff for its run $r$ in the period detector $D = \{Q, \{I_{\text{low}}, I_{\text{mid}}, I_{\text{high}}\}, \rightarrow, \text{lab}, L\}$ with $I_{\text{low}} = (-\infty, b_{\text{low}}), I_{\text{mid}} = [b_{\text{low}}, b_{\text{high}}), I_{\text{high}} = [b_{\text{high}}, \infty)$ and $|b_{\text{low}} - b_{\text{high}}| > L$ holds

$$r \models_{\text{LTL}} \square \diamond \text{start} \land \square (\text{start} \Rightarrow \diamond \text{accept}).$$

**Proof.** Postponed to Appendix B.5.
Preservation of the overall behavior: We will now show that the product MPM construction preserves the overall behavior of the original MPM.

Lemma 4. For a MPM $M = (S, R, AP, L)$ and its product MPM $M \otimes D = (S', R', AP', L')$ with a period detector LDFA $D = (Q, I, \to, lab, L_{pd})$, observables $\mu, \mu_{M \otimes D}$ and initial distributions $\alpha$ and $\alpha_{M \otimes D}$ holds

$$M \sim_{AP, \mu, \mu_{M \otimes D}} M \otimes D.$$


Proposition 7 (Product MPM Expansion CSL Preservation). A MPM $M = (S, R, AP, L)$ and its Product MPM $M \otimes D$ with period detector $D$ satisfy exactly the same CSL formulas restricted on the original set of atomic propositions $AP$ for any period detector $D$.

Proof. The proposition follows from Lemma 4 and Proposition 1.

6.4 Quantifying the Period Length in Noisy Periodic Systems

With the Product MPM construction of the previous Section and CSL Formula 2 (cf. page 30):

$$P \geq 1 [\Box (P \geq 1 \diamond \text{start})] \land P \geq 1 [\Box (\text{start} \Rightarrow P \geq 1 \diamond \text{accept})]$$

we were able to model check whether almost all $\mu$-trajectories of a MPM show noisy periodic behavior, i.e. there are infinitely many periods of minimal amplitude $|b_{low} - b_{high}|$. But by now, there was no constraint on the period length of the individual periods.

For the quantification of period length within this thesis, we will distinguish two kinds of period lengths (cf. Fig. 22). On the one hand, we are interested in the time needed to complete a period when it has just started. We will call that period length, the intra period length. On the other hand, we also would like to characterize the time needed in between two periods, i.e. the time between the completion of a period and the beginning of the next. We will refer to that period length as the inter period length.

The rationale behind splitting the period length into these two parts is that for many systems, the relevant properties are contained in only one of the two. For example in the case of neural networks it is assumed that information is encoded within in the interevent intervals (cf. Fig. 23) between successive spikes [AT02], which is measured exactly by the inter period length in our approach.

We can derive CSL formulas, allowing the quantification of these period lengths, directly from Formula 2 (cf. page 30). The first part of the formula, i.e.

$$P \geq 1 [\Box (P \geq 1 \diamond \text{start})]$$

46
will not be altered, since it demands that the system under consideration shall infinitely often start a period. The second part however intuitively encodes that whenever a period is started it shall somewhere complete and hence has to be adapted.

**Quantifying the Intra Period Length:** Now, instead of checking whether each period is completed somewhere, we would like to retrieve the time needed, i.e. the intra period length. Since our setting is stochastic, the period lengths will vary from period to period. Therefore, we will characterize that time via an interval \([T_{\text{intra,min}}, T_{\text{intra,max}}]\) in which a period is completed with a probability greater or equal to \(p\). The respective formula, derived from the second part of Formula 2 (cf. page 30) by constraining that whenever a period has started it will not be completed (\(\neg\text{accept}\)) until at least \(t \in [T_{\text{intra,min}}, T_{\text{intra,max}}]\) time units have passed, is

\[
P_{\geq 1} [\square (\text{start} \Rightarrow P_{\geq p_{\text{intra}}} [\neg\text{accept} U_{[T_{\text{intra,min}}, T_{\text{intra,max}}]} \text{accept}])]. \tag{6}
\]

Please note, that we had to introduce a probability bound \(p_{\text{intra}}\) in the inner probability operator, since apart from pathological cases (like already starting in the goal set), the probability of eventually reaching some set of goal states at the best converges against 1.0. The reason is that the exit time of states of CTMCs is exponentially distributed. Consequently, any finite time constraint on the eventually operator in combination with a probability bound \(\geq 1\) invalidates the whole formula in most of the cases. Therefore,
instead of enforcing that almost all trajectories when starting a period also complete it within the given time interval, we only demand that for certain fraction $p_{\text{intra}}$.

Unfortunately, this characterization is a multivariate problem in three variables $p_{\text{intra}}$, $T_{\text{intra,min}}$ and $T_{\text{intra,max}}$ usually with infinitely many solutions even for a fixed value of $p_{\text{intra}}$. Therefore, we will estimate $T_{\text{intra,min}}$ and $T_{\text{intra,max}}$ independently of each other and then retrieve the maximal probability bound $p_{\text{intra}}$ s.t. Formula 6 (cf. page 47) still holds.

Given a noisy periodic system, we can constrain the maximum intra period length, parameterized by $T_{\text{intra,max}}$, for probability bound $p_{\text{intra,max}}$ resembling the second part of Formula 2 (cf. page 30), via

$$P \geq 1 - 2 \left( \text{start} \Rightarrow P \geq p_{\text{intra,max}} \left[ \Diamond [0, T_{\text{intra,max}}] \text{accept} \right] \right). \quad (7)$$

For the minimum intra period length we have to ensure that any period that has started will not be completed within $T_{\text{intra,min}}$ time units at least with probability $p_{\text{intra,min}}$, i.e.

$$P \geq 1 - 2 \left( \text{start} \Rightarrow P \geq p_{\text{intra,min}} \left[ \neg \text{accept} \cup [T_{\text{intra,min}}, \infty) \text{ accept} \right] \right). \quad (8)$$

Here we had to use the until operator in order to express that with probability $p_{\text{intra,min}}$, a just started period shall be completed after at least $T_{\text{intra,min}}$ time units. To sum up, we suggest the following procedure to retrieve appropriate values for $T_{\text{intra,max}}$, $T_{\text{intra,min}}$ and $p_{\text{intra}}$:

- choose a probability bound $p$,
- retrieve $T_{\text{intra,max}}$ via Formula 7 for probability bound $p$,
- retrieve $T_{\text{intra,min}}$ via Formula 8 for probability bound $p$, and finally
Quantifying the Inter Period Length: The inter period time, i.e. the time between
ending a period and starting a new one, can then be quantified by swapping the start
and accept atomic propositions in Formula 6, i.e.

$$\mathcal{P}_{\geq 1}[^{\Box}(\text{accept} \Rightarrow \mathcal{P}_{\geq p_{\text{inter}}}[\neg \text{start} \cup [T_{\text{inter},\text{min}}, T_{\text{inter},\text{max}}] \text{start}])].$$

(9)

Consequently, the characterization of the maximum inter period length becomes

$$\mathcal{P}_{\geq 1}[^{\Box}(\text{accept} \Rightarrow \mathcal{P}_{\geq p_{\text{inter,max}}}[\Diamond [0, T_{\text{inter},\text{max}}] \text{start}])].$$

(10)

and for the minimum inter period length we get

$$\mathcal{P}_{\geq 1}[^{\Box}(\text{accept} \Rightarrow \mathcal{P}_{\geq p_{\text{inter,min}}}[\neg \text{start} \cup [T_{\text{inter},\text{min}}, \infty] \text{start}])].$$

(11)

Finally, in order to retrieve appropriate values for $T_{\text{inter},\text{max}}$, $T_{\text{inter},\text{min}}$ and $p_{\text{inter}}$ we suggest the following procedure:

- choose a probability bound $p$,
- retrieve $T_{\text{inter},\text{max}}$ via Formula 10 for probability bound $p$,
- retrieve $T_{\text{inter},\text{min}}$ via Formula 11 for probability bound $p$, and finally
- retrieve $p_{\text{inter}}$ via Formula 9 for time bounds $T_{\text{inter},\text{min}}, T_{\text{inter},\text{max}}$.

Figure 24 illustrates how these quantities characterize the periodic behavior of a system.

Algorithmic Considerations: However, model checking is a method that takes a system
to analyze together with a logical formula to check on that system and then states
whether the formula is satisfied or not. That is, there is no direct way of querying
the probability and time bounds. Thus, we propose binary search based approaches to
retrieve the time bounds for a given probability bound, up to a certain precision, i.e.
Algorithm 1 with formulas 7, 10 for upper time bounds $T_{*,\text{max}}$ and Algorithm 2 with
formulas 8, 11 for lower time bounds $T_{*,\text{min}}$. Given such time bounds $T_{*,\text{min}}, T_{*,\text{max}}$, the
maximum probability bound for formulas 6, 9 to hold can be estimated using Algorithm
3.
Determining the complete Period Length: The complete period length can be determined with our approach as well, but with the period detector automaton as presented in Section 6.3.1, this is not trivially possible. The reason is that we can not directly measure the time between two consecutive passings of the unique start state. Hence, two period detector automata could be connected in series, i.e. the $I_{low}$ transition of one automata connects to the $q_0$ state of the other one. Please note that the overall behavior of the period detector is not changed by this extension. Now, the total period length corresponds to the time between the start state of one automaton and the start state of the other. First experiments show the applicability of this construction.
Algorithm 1: EstimateTmax(Φ, p, ε)

**Input:** Φ: param. CSL formula, p: prob. bound, ε: precision of result

**Output:** maximal time bound to satisfy Φ for probability bound p

**begin**

\[ T_{\text{min}} \leftarrow 0.0; T_{\text{max}} \leftarrow 1.0 \]

// quickly get a rough value for T where formula holds

while ModelCheck(Φ, p, T_{\text{max}}) = false do

\[ T_{\text{min}} \leftarrow T_{\text{max}}; T_{\text{max}} \leftarrow 2 \cdot T_{\text{max}} \]

// search for T by shrinking \([T_{\text{min}}, T_{\text{max}}]\)

repeat

\[ T_{\text{mid}} \leftarrow (T_{\text{max}} - T_{\text{min}})/2 \]

if ModelCheck(Φ, p, T_{\text{mid}}) = true then

\[ T_{\text{max}} \leftarrow T_{\text{mid}} \]

else

\[ T_{\text{min}} \leftarrow T_{\text{mid}} \]

until \(|T_{\text{min}} - T_{\text{max}}| < \epsilon\)

return \(T_{\text{max}}\)

**end**

Algorithm 2: EstimateTmin(Φ, p, ε)

**Input:** Φ: parameterized CSL formula, p: prob. bound, ε: precision of result

**Output:** minimal time bound to satisfy Φ for probability bound p

**begin**

\[ T_{\text{min}} \leftarrow 0.0; T_{\text{max}} \leftarrow 1.0 \]

// quickly get a rough value for T where formula does not hold

while ModelCheck(Φ, p, T_{\text{max}}) = true do

\[ T_{\text{min}} \leftarrow T_{\text{max}}; T_{\text{max}} \leftarrow 2 \cdot T_{\text{max}} \]

// search for T by shrinking \([T_{\text{min}}, T_{\text{max}}]\)

repeat

\[ T_{\text{mid}} \leftarrow (T_{\text{max}} - T_{\text{min}})/2 \]

if ModelCheck(Φ, p, T_{\text{mid}}) = false then

\[ T_{\text{max}} \leftarrow T_{\text{mid}} \]

else

\[ T_{\text{min}} \leftarrow T_{\text{mid}} \]

until \(|T_{\text{min}} - T_{\text{max}}| < \epsilon\)

return \(T_{\text{min}}\)

**end**
Algorithm 3: EstimateP(Φ, T_{min}, T_{max}, \epsilon)

**Input**: Φ: parameterized CSL formula, T_{min}: lower time bound, T_{max}: upper time bound, \epsilon: precision of result

**Output**: maximal probability bound to satisfy Φ for time interval [T_{min}, T_{max}]

begin

\[ p_{min} \leftarrow 0; p_{max} \leftarrow 1 \]

// search for \( p \) by shrinking \([p_{min}, p_{max}]\)

repeat

\[ p_{mid} \leftarrow \frac{(p_{max} - p_{min})}{2} \]

if ModelCheck(Φ, p, [T_{min}, T_{max}]) = false then

\[ p_{max} \leftarrow p_{mid} \]

else

\[ p_{min} \leftarrow p_{mid} \]

until \(|p_{min} - p_{max}| < \epsilon\)

return \( p_{min} \)

end
7 Tool: BioToPrism

In order to automate the task of checking whether a biological system oscillates or is noisy periodic and in the latter case for which intra and inter period length bounds, a prototypical tool – BioToPrism – has been developed in Java, forming a fully working tool chain (cf. Fig. 25) together with the Prism model checker. Currently, the only supported observable for the resulting Markovian Population Models is the projection onto the molecule level of a certain species of interest, $S$.

![Diagram]

Figure 25: An overview of the tool chain.

Depending on whether oscillatory or noisy periodic behavior shall be investigated, an according pathway in the tool chain has to be taken.

**Checking Oscillatory Behavior:** The XML-based description of the biological system is processed by the tool which directly produces an output Prism model. This model contains one module encoding the Markovian Population Model generated via the reaction types and another module keeping track of the changes in molecule level. Both modules are composed parallelly, together resembling the change predicate expanded MPM like introduced in Section 6.2.1. The oscillation property can then be checked within Prism by model checking Formula 4 (cf. page 38).

**Checking Noisy Periodic Behavior:** The input system is first encoded as a single module inside a Prism model. After that, Prism can be used to retrieve the steady-state distribution of the molecule level of the species of interest. The tool then calculates the
expected number of molecules in the steady state and the respective standard deviation. These quantities can then be used to retrieve the boundary values \( b_{\text{low}} \) and \( b_{\text{high}} \) for certain kinds of noisy periodic models. Usually, a significant fluctuation of twice the standard deviation from the mean number of molecules is taken for the minimal amplitude of oscillation. Of course, both boundaries can also be specified manually. Finally, a period detector module is constructed, which is then parallelly composed with the MPM encoding the system of interest – analogous to the period detector expansion of Section 6.3.1. Consequently, in order to check whether the system is noisy periodic, Prism can be used to model check Formula 2 (cf. page 30). In order to retrieve the inter and intra period length bounds, formulas 5, 6, 7, 8, 9, 10, and 11 (cf. pages 47f.) can be model checked using Prism.

### 7.1 Input File Format

The basic structure of the XML-based input format used to describe biological systems is:

```xml
<system name="system_name">
    <species> ... </species>
    <rates> ... </rates>
    <reactions> ... </reactions>
</system>
```

The system has to be given a name specified inside the `name` attribute of the `<system>` tag. The Prism module encoding the system will then be named that way.

**Species:** Inside the `<species>` tag all species that occur inside the chemical reactions governing the biological system have to be listed. By now, only discrete molecule levels are supported. A discrete species can be declared with

```xml
<discrete name="species_name" max="vmax" init="vinit" interest="yes_no"/>
```

where the `name` tag specifies the species’ name which then can be referenced inside the description of the reaction types later on. Since our framework and Prism can only handle finite MPMs, the maximum number of molecules has to be constrained by the `max` attribute. Currently, only a single initial configuration is supported, i.e. the `init` attribute states the initial molecule level of the respective species. The species of interest has to be marked by the attribute `interest="yes"`, all others by `interest="no"`.

**Rate Constants:** Each chemical reaction type has a certain reaction rate. Inside the input format of the tool all reaction rates are treated as constants, and hence have to be defined before usage within the description of the reaction types. The advantage is that those rates are then defined as constants in the resulting Prism model as well,
allowing for quick changes directly within Prism without having to re-translate the XML description file. A single rate constant is declared within the `<rates>` tag via

```xml
<rate name="rate_name" value="vrate"/>
```

where each rate has to be assigned a name specified by the `name` attribute and a real value by the `value` attribute.

**Reaction Types:** Reaction types are declared within the `<reactions>` tag on the system level. The syntax of a single reaction type is:

```xml
<reaction>
  <reactants> ... </reactants>
  <rate name="rate_name"/>
  <products> ... </products>
</reaction>
```

Within the `<reactants>` tag, all reactant species of the current reaction type have to be declared via

```xml
<reactant name="reactant_name" count="rcount"/>
```

where the `reactant_name` attribute refers to the name of a species declared inside the `<species>` tag before. The `count` attribute contains the number of molecules of that kind of reactant species needed. The product species are declared analogously via

```xml
<product name="product_name" count="pcount"/>
```

The reaction rate constant is declared via the `<rate>` tag where the attribute `rate_name` refers to a previously defined constant.

### 7.2 Prism Model Generation and Model Checking

The command line parameters of the tool are

```
BioToPrism in.bio out.pm [-cd | -pd [low high | ssdist.file]]
```

When only specifying the input biological system XML-description `in.bio`, solely the Prism model `out.pm` of the system is generated. The parameter `-cd` (change detector) is used to generate the change predicate expanded system, while `-pd` (period detector) triggers the period detector construction with the option to use the specified boundary values $b_{low} = \text{low}$ and $b_{high} = \text{high}$ or letting the tool calculate appropriate values from a steady state distribution file `ssdist.file`. 
ctmc

const maxS_i = max_i;  // species S_i
c const double c_r = v_r;  // rate constant c_r

module sname
  // species declarations
  nS_i : [0..maxS_i] init init_i;  // species S_i
  ...
  // reaction types
  ...
endmodule

// optional :
module [ChangeDetector|PeriodDetector]
  ...
endmodule

Figure 26: Basic structure of Prism models generated by the tool.

7.2.1 Basic Model Generation

From the XML-description of the biological system the following information is extracted:

- **sname**, the name of the system
- \( N \) species \( S_i \in \{S_1,\ldots,S_N\} \), each with a maximum number of molecules \( max_i \) and an initial count \( init_i \). Exactly one species \( S_k \) with \( k \in \{1,\ldots,N\} \) is marked as the species of interest.
- \( R \) reaction rate constants \( c_r \in \{c_1,\ldots,c_R\} \) with values \( v_1,\ldots,v_R \)
- \( M \) reaction types of the form
  \[ a_{m1} \cdot S_1 + \ldots a_{mN} \cdot S_N \xrightarrow{c_m} b_{m1} \cdot S_1 \ldots b_{mN} \cdot S_N \]
  with \( m \in \{1,\ldots,M\} \). Coefficients \( a_{mi} \) for \( i \in \{1,\ldots,N\} \) that are not explicitly defined as a reactant species w.r.t. reaction type \( m \) are assumed to be zero. The same holds for the coefficients of the product species.

From this information, a Prism model is generated. The basic structure of that Prism model is depicted in Figure 26. The constant section of the Prism model contains a maximum constant \( maxS_i \) for each species \( S_i \). Also the rate constants \( c_r \) are encoded as constants \( c_r \). The system itself is translated into a Prism module named **sname**.
Inside that module, each reaction type \( m \in \{1, \ldots, M\} \) is translated to a Prism guarded command

\[
[X] \ (nS_1 >= a_{m1}) \ & \ \ldots \ & \ (nS_N >= a_{mN}) \ & \\
(nS_1 + b_{m1} - a_{m1} <= maxS_1) \ & \ \ldots \ & \ (nS_N + b_{mN} - a_{mN} <= maxS_N) \ \\
\rightarrow \ c_m \ * \ #((nS_1, a_{m1}) \ * \ \ldots \ * \ #((nS_N, a_{mN}) : \\
(nS_1' = nS_1 + b_{m1} - a_{m1}) \ & \ \ldots \ & \ (nS_N' = nS_N + b_{mN} - a_{mN});
\]

according to Section 2.2.3, where

\[
#(nS_i, v) = \frac{nE \cdot (nE - 1) \cdot \ldots \cdot (nE - v + 1)}{v \cdot (v - 1) \cdot \ldots \cdot 1}
\]

is a symbolic expression stating the number of possibilities to choose \( v \) elements out of \( nS_i \), e.g.

\[
#(nE, 3) = \frac{nE \cdot (nE - 1) \cdot (nE - 2)}{3 \cdot 2 \cdot 1}.
\]

The guard of the reaction type’s translation evaluates to true iff the reaction type currently is applicable, i.e. there are enough reactant molecules available \( (nS_i >= a_{mi}) \) and after the reaction, the maximum bound of each species is not exceeded \( (nS_i + b_{mi} - a_{mi} <= maxS_i) \). The synchronization label \([X]\) depends on the context of model generation:

- If the tool shall only generate the Prism model without expansion, this label is defined as \([\]\).
- When generating the change predicate expanded version:

\[
[X] = \begin{cases} 
\text{[change]} & \text{if } a_{mk} \neq b_{mk}, \\
\text{[stay]} & \text{otherwise}
\end{cases}
\]

Consequently, the synchronization label encodes whether the reaction type leads to a change in the molecule level of the species of interest, or not.

- When generating the period detector expanded version, the label is defined as

\[
[X] = \begin{cases} 
\text{[decY]} & \text{if } Y = (a_{mk} \neq b_{mk}) > 0, \\
\text{[incY]} & \text{if } Y = (a_{mk} \neq b_{mk}) < 0, \\
\text{[stay]} & \text{otherwise.}
\end{cases}
\]

Thus, the label now also encodes the quantity \( Y \) of change in molecule level (if any).
module ChangeDetector
    changed : bool init false;

    [change] true -> 1 : (changed’ = true);
    [stay] true -> 1 : (changed’ = false);
endmodule

Figure 27: Change Detector Prism module.

7.2.2 Change Detector Expansion

When the tool has been called with the command line parameter `-cd`, additionally to the module encoding the biological system, the change detector module is built and parallelly composed. This module (cf. Fig. 27) keeps track of whether a change in molecule level has happened during the last transition via the boolean variable `changed` which is set at each transition via synchronization with the system’s module on the labels `change` and `stay`, resembling the construction of Section 6.2.1. Please note that inside the change detector, rates of 1 have been chosen to preserve the rates of the original system. Now, in order to check for oscillatory behavior, Formula 4 (cf. page 38) can be expressed in Prism CSL syntax utilizing the `changed` predicate as

\[ P \leq 0 \ [true U P < 1 [true U changed]] \]

where reachable absorbing states are detected automatically by Prism.

7.2.3 Period Detector Expansion

When checking for noisy periodic behavior, the interval boundaries `b_{low}` and `b_{high}` encoding the minimal amplitude of oscillation have to be specified in beforehand.

Automated Boundary Value Estimation: This can be done either manually via the command line parameter `-pd b_{low} b_{high}` or by specifying a file `ssdist.file` containing the steady state distribution of the molecule level of the species of interest via the parameter `-pd ssdist.file`. Such a file can be created from a system description file `system.bio` by the following steps:

1. Generation of the plain system via `BioToPrism system.bio system.pm`.
2. Starting a Prism experiment on the generated model `system.pm`, querying the steady state distribution via the formula

\[ S =? [nS_k = number] \]

for each possible molecule level, \( number \in \{0, \ldots maxS_k\} \), where \( S_k \) is the species of interest.
3. Exporting the experiment results to a file `ssdist.file`.

The steady state distribution $\pi_{S_k}$ is then used to calculate the expectation

$$E[S_k] = \sum_{n \in \{1, \ldots, \max S_k\}} \pi_{S_k}(n) \cdot n,$$

variance

$$V[S_k] = \sum_{n \in \{1, \ldots, \max S_k\}} (E[S_k] - n)^2 \cdot \pi_{S_k}(n),$$

and standard deviation

$$\sigma_{S_k} = \sqrt{V[S_k]}.$$

The two boundary values are then calculated via

$$b_{low} = \max\{E[S_k] - \sigma_{S_k}, 1.0\} \quad \text{and} \quad b_{high} = E[S_k] + \sigma_{S_k}$$

and stored as constants `low` and `high` inside the Prism model. The minimal value of $b_{low}$ is set to 1.0, because our observable projects a state on the number of molecules of the species of interest and therefore is discrete. More precisely, $\mu(s) \in \mathbb{N}$ for each state $s$. Now, if $E[S_k] < \sigma_{S_k} + 1.0$, the interval $I_{low} = (-\infty, b_{low})$ would only contain negative discrete molecule levels and therefore the interval would never be crossed. Thus, we assume the $I_{low}$ interval to include at least the zero molecule level. Another problem arises if $|b_{low} - b_{high}|$ is smaller than the maximal change in molecule level of the species of interest, i.e. if its standard deviation is too small. Then, the $I_{mid} = (b_{low}, b_{high}]$ interval can possibly be skipped. Those cases result in the noisy periodicity formula to be violated since the absorbing, error-labeled $q_E$ state of the period detector will be reached. The tool is able to detect situations like this and outputs a warning.

This approach to automatically estimate the boundary values $b_{low}$ and $b_{high}$ works well in cases where the center of oscillation roughly coincides with the expectation of the observation, like e.g. in systems where the $\mu$-trajectories show a sinusoidal oscillation pattern (cf. Fig. 28, left). But for certain patterns like periodic peaks interleaved with high noise, this may lead to problems, like wrong characterizations of the period lengths. For instance, the period length will be underestimated for the trajectory depicted on the right side of Figure 28, since also noise will be regarded as the starts and ends of periods. The suggested solution for situations like this is to generate sample traces of the system under consideration in order to estimate the boundary values and then to manually adjust them via the command line parameters `-pd b_low b_high`. For the periodic peak example, a more appropriate choice of the values could be $b_{low} = E[\mu] + 3 \cdot \sigma[\mu]$ and $b_{low} = E[\mu] + 8 \cdot \sigma[\mu]$. The values of $E[\mu]$ and $\sigma[\mu]$ for the steady state are printed to the standard output when calling the tool with the `-pd ssdist.file` parameter.

**Period Detector Module Generation:** In order to keep track of the interval, the current molecule level of the species of interest lies in, the period detector module (cf. Fig. 30) is built and composed parallely to the module encoding the biological system.
Figure 28: Sinusoidal $\mu$-trajectory (left); $\mu$-trajectory with periodic peaks (right).

Figure 29: $\mu$-trajectory with periodic peaks and corrected boundary values.

The module encodes exactly the period detector LDFA presented in Section 6.3.1. The inner state variable keeps track of the period detector’s current state. Transitions of the detector are synchronized with those of the system’s module via the incY, decY and stay labels encoding the actual change $Y$ in molecule level. Formulas incY.inZ, decY.inZ and stay.inZ evaluate to true iff the molecule level of the species of interest lies in interval $I_Z$ with $Z \in \{\text{low, mid, high}\}$ after the respective transition. Again a rate of 1 is used to preserve the original system’s rates.

**Applied Noisy Periodicity Model Checking:** The boolean formulas state, accept and error can be used to reason about the current label of the period detector component inside Prism CSL formulae. Consequently, model checking the first condition of noisy periodicity (cf. Formula 5, page 46) becomes

$$P_{<=0} \left[ \text{true U P<1 [true U start]} \right].$$

The above mentioned problem of the $I_{\text{mid}}$ interval being too small can be detected within Prism via the formula

$$P_{<=0} \left[ \text{true U P<1 [true U start]} \right].$$
\[ P > 0 \text{ [true U error]} \].

The various period length bounds can be quantified using the Prism CSL formulas (cf. pages 47f.)

- \( P \leq 0 \text{ [true U (start & P<p [true U<=T accept])]} \)
  (maximum intra period length – Formula 7),

- \( P \leq 0 \text{ [true U (start & P<p ![accept U>=T accept])]} \)
  (minimum intra period length – Formula 8),

- \( P \leq 0 \text{ [true U (accept & P<p [true U<=T start])]} \)
  (maximum inter period length – Formula 10), and

- \( P \leq 0 \text{ [true U (accept & P<p ![start U[T_{min}, T_{max}] start])]} \)
  (minimum inter period length – Formula 11).

Finally, the probability bound for the combined interval boundaries (formulas 6 and 9) can be quantified using

- \( P \leq 0 \text{ [true U (start & P<p ![accept U[T_{min}, T_{max}] accept])]} \)
  for the intra period length, and

- \( P \leq 0 \text{ [true U (accept & P<p ![start U[T_{min}, T_{max}] start])]} \)
  for the inter period length.

Repeated Prism experiments with varying values for \( T \) and \( p \) can be used to mimic the binary search algorithms presented in Section 6.4.
formula incl_inLow = ((nS_k+1) < low);
formula incl_inMid = (((nS_k+1) >= low) & ((nS_k+1) < high));
formula incl_inHigh = ((nS_k+1) >= high);
...
formula start = (state = 2);
formula accept = (state = 0);
formula error = (state = 7);

module PeriodDetector
  state: [0..7] init 3;

  [incl] (state = 0) & incl_inLow -> 1 : (state' = 1);
  [incl] (state = 0) & incl_inMid -> 1 : (state' = 2);
  [incl] (state = 0) & incl_inHigh -> 1 : (state' = 7);
  [incl] (state = 1) & incl_inLow -> 1 : (state' = 1);
  [incl] (state = 1) & incl_inMid -> 1 : (state' = 2);
  [incl] (state = 1) & incl_inHigh -> 1 : (state' = 7);
  ...
  [dec] (state = 0) & dec_inLow -> 1 : (state' = 1);
  [dec] (state = 0) & dec_inMid -> 1 : (state' = 2);
  [dec] (state = 0) & dec_inHigh -> 1 : (state' = 7);
  ...
  [stay] (state = 0) & stay_inLow -> 1 : (state' = 1);
  ...
endmodule

Figure 30: Period Detector Prism module.
8 Case Studies

Finally, we evaluated the presented approaches via two case studies, the 3-way Oscillator (doped and un-doped) [BMM09] as well as the Repressilator [EL00]. The hardware and software used was a Mac Mini, 2.0GHz Intel Core 2 Duo, 1 GB 1067MHz DDR3 running Mac OS X 10.5.8 and Prism 3.3.beta2 (standard options).

8.1 3-way Oscillator

As a first case study we analyzed whether the 3-way Oscillator presented in Section 3.1 oscillates and shows a noisy periodic behavior. We will used the tool BioToPrism for the model generation from the description of the system’s reaction types and Prism for the model checking part. Please note that the original system as well as its doped version are closed, i.e. no molecules enter the system from outside. Consequently, the generated MPM is finite.

8.1.1 3-way Oscillator without doping

First, we investigated the original (un-doped) 3-way Oscillator with its reaction types

\[ A + B \xrightarrow{r_A} B + B \quad B + C \xrightarrow{r_B} C + C \quad C + A \xrightarrow{r_C} A + A. \]

As already argued in Section 3.1, the system might reach a deadlock state in which no reaction type is applicable any more. Hence, the system does not oscillate nor does it show sustaining noisy periodic behavior. These two observations are reflected in our approach as well. An XML description of the above reaction network is depicted in Appendix C.1. The rates \( r_A \), \( r_B \), and \( r_C \) were equally set to 1.0. Since the system behaves symmetrically in all three species \( A \), \( B \) and \( C \), it suffices to analyze the behavior of species \( A \).

Model Growth: First, we will studied the growth in the number of states and transitions, caused by the presented expansions, i.e. we generated the basic Prism models and their change detector and period detector expanded versions for varying initial numbers \( n \) of molecules for each species (table 1). Obviously, as argued in Section 6.2.2, the growth in state space and transitions for the change detector expanded model w.r.t. to the basic model is bounded by a constant factor of 2. But also for the period detector expanded model, the number of states and transitions is only roughly doubled, although the hard limit is 8 (see Section 6.3.2).

Model Checking Results: During Prism’s model building phase, three reachable absorbing states \((nA, nB, nC) \in \{(n, 0, 0), (0, n, 0), (0, 0, n)\} \) have been detected for each \( n \in \{5, 10, 20, 30\} \). Thus, the system does not oscillate according to Formula 4 (cf. page 38). Likewise, the first part of the noisy periodicity property, stating that almost all trajectories consist of an infinite sequence of periods, is violated for any non-pathological set of boundary values \( b_{\text{low}} \) and \( b_{\text{high}} \).
Table 1: 3-way Oscillator (no doping) Prism models for initial molecule level $n$ of each species $A$, $B$, and $C$, where * means that absorbing states have been detected during Prism’s model building phase.

<table>
<thead>
<tr>
<th>$n$</th>
<th>basic model (states/transition)</th>
<th>change detector exp. model (states/transition)</th>
<th>oscillatory</th>
<th>period detector exp. model (states/transition)</th>
<th>noisy periodic</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>136 / 318 *</td>
<td>237 / 573 *</td>
<td>no</td>
<td>278 / 640 *</td>
<td>no</td>
</tr>
<tr>
<td>10</td>
<td>496 / 1308 *</td>
<td>927 / 2493 *</td>
<td>no</td>
<td>998 / 2602 *</td>
<td>no</td>
</tr>
<tr>
<td>20</td>
<td>1891 / 5313 *</td>
<td>3657 / 10383 *</td>
<td>no</td>
<td>3761 / 10495 *</td>
<td>no</td>
</tr>
<tr>
<td>30</td>
<td>4186 / 12018 *</td>
<td>8187 / 23673 *</td>
<td>no</td>
<td>8288 / 23680 *</td>
<td>no</td>
</tr>
</tbody>
</table>

8.1.2 3-way Oscillator with doping

When adding doping substances for each species $A$, $B$ and $C$, the 3-way Oscillator’s chemical reaction network becomes

\[
A + B \xrightarrow{r_A} B + B \quad B + C \xrightarrow{r_B} C + C \quad C + A \xrightarrow{r_C} A + A
\]

\[
DA + C \xrightarrow{r_C} A + DA \quad DB + A \xrightarrow{r_A} B + DB \quad DC + B \xrightarrow{r_B} C + DC.
\]

Again, we set the reaction rates $r_A$, $r_B$, and $r_C$ to 1.0 and the species of interest to be $A$. The corresponding XML description is depicted in Appendix C.2.

**Model Growth:** The growth in state space and number of transitions (table 2) again is roughly bounded by 2, like in the un-doped version. For the boundary values of the period detector the standard deviations from the expected molecule level in the steady state are taken (table 2). Period detector expanded models for $n = 50$ and $n = 100$ were not generated since the steady state distribution calculation could not be completed within reasonable time.

Table 2: 3-way Oscillator (with doping) Prism models for initial molecule level $n$ of each species $A$, $B$, and $C$.

<table>
<thead>
<tr>
<th>$n$</th>
<th>basic model (states/transition)</th>
<th>change detector exp. model (states/transition)</th>
<th>period detector exp.model (states/transition)</th>
<th>$E[\mu]$</th>
<th>$\sigma[\mu]$</th>
<th>$I_{mid}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>136 / 360</td>
<td>255 / 689</td>
<td>272 / 731</td>
<td>5.0</td>
<td>3.874</td>
<td>[1.126, 8.874]</td>
</tr>
<tr>
<td>10</td>
<td>496 / 1395</td>
<td>960 / 2729</td>
<td>957 / 2708</td>
<td>10.0</td>
<td>7.416</td>
<td>[2.584, 17.416]</td>
</tr>
<tr>
<td>20</td>
<td>1891 / 5490</td>
<td>3720 / 10859</td>
<td>3541 / 10306</td>
<td>20.0</td>
<td>14.49</td>
<td>[5.51, 34.49]</td>
</tr>
<tr>
<td>30</td>
<td>4186 / 12285</td>
<td>8280 / 24389</td>
<td>7756 / 22797</td>
<td>30.0</td>
<td>21.56</td>
<td>[8.44, 51.56]</td>
</tr>
<tr>
<td>50</td>
<td>11476 / 33975</td>
<td>22800 / 67649</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>100</td>
<td>45451 / 135450</td>
<td>90600 / 270299</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Model Checking Results: The model checking results and times of the oscillation condition as well as the noisy periodicity property are depicted in table 3. Since the doping substances allow the system to recover from the deadlock situations \( (n_A, n_B, n_C) \in \{(n, 0, 0), (0, n, 0), (0, 0, n)\} \) of the un-doped version, the 3-way Oscillator becomes oscillatory as well as noisy periodic – as expected.

<table>
<thead>
<tr>
<th>n</th>
<th>oscillatory</th>
<th>time needed (model building + checking)</th>
<th>noisy periodic</th>
<th>time needed (model building + checking)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>yes</td>
<td>0.078s + 0.00505s</td>
<td>yes</td>
<td>0.15s + 0.039s</td>
</tr>
<tr>
<td>10</td>
<td>yes</td>
<td>0.178s + 0.011s</td>
<td>yes</td>
<td>0.667s + 0.122s</td>
</tr>
<tr>
<td>20</td>
<td>yes</td>
<td>0.842s + 0.026s</td>
<td>yes</td>
<td>1.367s + 0.55s</td>
</tr>
<tr>
<td>30</td>
<td>yes</td>
<td>2.162s + 0.036s</td>
<td>yes</td>
<td>3.674s + 1.377s</td>
</tr>
<tr>
<td>50</td>
<td>yes</td>
<td>7.468s + 0.088s</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>100</td>
<td>yes</td>
<td>40.2s + 0.172s</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 3: 3-way Oscillator (with doping) model checking times and results.

Quantification of the Period Length: Having ensured that the system indeed behaves noisy periodic, we quantified the minimum/maximum intra and inter period length boundaries according to Section 6.4. We did that for a fixed initial molecule level \((n_A, n_B, n_C) = (5, 5, 5)\). At first a basic model was generated, which was then used to calculate the steady state distribution of the molecule level of species \(A\) (table 2) in order to estimate the boundary values \(b_{\text{low}} = 1.126\) and \(b_{\text{high}} = 8.874\).

The results for a varying initial probability bound \(p\) for time boundaries \(T_{*, \text{min}}, T_{*, \text{max}}\) are depicted in Figures 31 and 32. First, we used Prism experiments and the formulas from Section 7.2.3 to retrieve the minimum upper boundaries \(T_{\text{intra,max}}\) (ordinate of the plot in Fig. 31) for the intra period length, s.t. whenever the system has just started a period, that period will be completed within \(T_{\text{intra,max}}\) time units, with a probability of at least \(p\) (abscissa of the plot in Fig. 31). Next, we did the same for the maximum lower boundaries \(T_{\text{inter,min}}, \text{i.e. whenever a period has started, with probability } p \text{ it will only complete after } T_{\text{inter,min}}\) time units. Now, we combined both boundaries by calculating the maximum probability bound \(p_{\text{intra}}, \text{s.t. whenever a period has begun, it will be finished within } t \in [T_{\text{intra,min}}, T_{\text{intra,max}}] \text{ with at least that probability.} \)

As can be seen in Figure 31, up to a certain probability bound \(p\) of around 0.25, the lower time bound \(T_{\text{intra,min}}\) is greater than the upper time bound \(T_{\text{inter,max}}\). The reason is that simply too few of the trajectories are constrained on and therefore a very low upper bound and a very high lower bound suffice to capture that probability mass \(p\). Furthermore, the minimum respectively the maximum time points were taken for the bounds. The same procedure was repeated for the inter period length (cf. Fig. 32). Finally, a \(\mu\)-trajectory of the doped 3-way Oscillator can be characterized e.g. as:

Each period very likely \((p_{\text{intra}} = 0.866)\) completes within \(t \in [0.415, 1.822]\) and a
new period most likely \((p_{\text{inter}} = 0.898)\) starts after \(t \in [0.0041, 0.513]\) time units.

A sample \(\mu\)-trajectory of the doped 3-way Oscillator is depicted in Figure 33, which illustrates that the intra and inter period lengths indeed behave like predicted.

Figure 31: Estimating the intra period length boundary window \([T_{\text{intra,min}}, T_{\text{intra,max}}]\) for probability bound \(p_{\text{intra}}\) via initial constraint \(p\) on individual boundaries \(T_{\text{intra,min}}\) and \(T_{\text{intra,max}}\).

Figure 32: Estimating the inter period length boundary window \([T_{\text{inter,min}}, T_{\text{inter,max}}]\) for probability bound \(p_{\text{inter}}\) via initial constraint \(p\) on individual boundaries \(T_{\text{inter,min}}\) and \(T_{\text{inter,max}}\).
8.2 Repressilator

Just like the 3-way Oscillator, the Repressilator is a synthetic, self-regulating gene network which is supposed to oscillate [EL00]. Also for this system an at least partial discrete and stochastic way of modeling and analysis is suggested [BP09], since the classical ODE approach does not reveal information about the oscillatory character of the system. The system consists of three proteins $A$, $B$, and $C$ which are expressed by three genes. In contrast to the 3-way Oscillator, the three species do not boost their expression in a cyclic fashion, but repress it.

Each of the proteins is supposed to be produced with the same reaction rate constant $k_p$ but only if its gene is not repressed. We model the un-repressed genes as species $G_A$, $G_B$ and $G_C$. Thus, the chemical reaction types for protein production become

$$G_A \xrightleftharpoons[k_p]{\quad} G_A + A$$
$$G_B \xrightarrow[k_p]{\quad} G_B + B$$
$$G_C \xrightarrow[k_p]{\quad} G_C + C.$$
Each of the species $A$, $B$, and $C$ also degrades with the same reaction rate $k_d$:

$$A \xrightarrow{k_d} \emptyset$$

$$B \xrightarrow{k_d} \emptyset$$

$$C \xrightarrow{k_d} \emptyset.$$ 

Now, $C$ molecules restrict the production of $A$ molecules by repressing the $G_A$ gene. In the same way $A$ molecules repress the $G_B$ gene, and $B$ molecules repress the $G_C$ gene, forming a cycle. Again, all those reactions happen with the same rate $k_b$:

$$C + G_A \xrightarrow{k_b} C$$

$$A + G_B \xrightarrow{k_b} A$$

$$B + G_C \xrightarrow{k_b} B.$$ 

Moreover, the genes can recover with rate $k_u$:

$$\emptyset \xrightarrow{k_u} G_A$$

$$\emptyset \xrightarrow{k_u} G_B$$

$$\emptyset \xrightarrow{k_u} G_C.$$ 

Furthermore, a maximum of only one un-repressed gene per type may exist. The corresponding XML description is depicted in Appendix C.3. For the case study we used the rates

$$k_b = 5.0, k_p = 5.0, k_d = 1.2,$$ and $$k_u = 1.0.$$ 

The protein degradation rate $k_d$ was chosen to be slightly larger as the gene un-binding rate $k_u$ in order to retrieve a higher frequency of periods, as suggested in [BP09]. We took an initial molecule level of zero for all species $A, B, C$. Also all three gene species $G_A, G_B$, and $G_C$ are repressed initially, i.e. their initial molecule level is zero, too. The constraint to have a maximum of one gene molecule per type was realized in the XML-description by setting the max attribute of the three species $G_A, G_B$ and $G_C$ to 1.

Obviously, the Repressilator system is not a closed system, since the reaction rules abstract from details governing protein production. More precisely, proteins are translated directly from the respective genes without taking into account the intermediate steps that require the presence of certain other species like amino acids. The effect of this abstraction is that the amount of proteins in our model – in principle – might grow unboundedly, although this is very unlikely to observe in real world. Consequently, when naively generating the state space from this description and setting no constraint on the maximum number of proteins, the state space would be infinite.

So, in order to retrieve an upper bound for the number of protein molecules (max attribute in the XML description), the steady state distribution of their molecule level was analyzed for varying boundary values. Since the Repressilator behaves symmetrically
in all three protein species (just like the 3-way Oscillator), the same bound was taken for all species and it sufficed to retrieve the steady state molecule level distribution of species A (cf. Fig. 34). The actual bound of 10 molecules was taken, since more than 99.8% of the steady state’s probability mass is located within a molecule level below 10 and model checking times were still feasible. Consequently, the model size could not be varied because of the upper bound and increasing noise level for lower molecule numbers. Please note that this is only an approximation of the real steady state of the infinite sized system, because the upper bounds already alter the system. Nevertheless, this approach has been taken in order to be able to analyze this system at all. Unfortunately, it is out of the scope of this master thesis to come up with a formal approach for retrieving a finite subset of the infinite state space that should be analyzed instead. This problem is therefore considered as future work.

<table>
<thead>
<tr>
<th></th>
<th>basic model</th>
<th>change detector exp. model</th>
<th>period detector exp. model</th>
</tr>
</thead>
<tbody>
<tr>
<td>states</td>
<td>10648</td>
<td>20812</td>
<td>15972</td>
</tr>
<tr>
<td>transitions</td>
<td>74052</td>
<td>144870</td>
<td>109362</td>
</tr>
</tbody>
</table>

Table 4: Repressilator Model Sizes

Table 4 shows the sizes of the state spaces of the basic model as well the change and period detector expanded models. As in the 3-way Oscillator case study, the change
detector expanded model is roughly twice the size of the original model, i.e. very close to the hard bound of 2. Now, the size of the period detector expanded model is only about 1.5 the size of the original model, i.e. far below the hard limit of 8.

In the following analysis, the projection from states onto the respective A molecule level was taken as the observable, since – as already argued – the system behaves symmetrically in the protein species. Now, model checking the oscillation condition revealed that the Repressilator system as represented within this case study indeed oscillates. In order to analyze noisy periodicity, the minimal amplitude of oscillation has been set to twice the standard deviation from the expected molecule, i.e.

\[ E[\mu] = 1.758, \sigma[\mu] = 2.0059 \]
\[ b_{\text{low}} = \max\{1.0, E[\mu] - \sigma[\mu]\} = 1.0 \]
\[ b_{\text{high}} = E[\mu] + \sigma[\mu] = 3.7639 \]

just like proposed in Section 7.2.3. As expected, the first condition of noisy periodicity, i.e. infinitely many period starts, is satisfied. The results of the quantification of the period length boundaries are depicted in Figures 35 for the intra period length and Figure 36 for the inter period length. A sample \( \mu \)-trajectory is illustrated in Figure 37. Apparently, the Repressilator’s behavior with the reaction rates chosen as above, is quite noisy. That means both, the amplitude as well as the period length of the individual periods, differ significantly from period to period. Nevertheless, our approach estimated the minimal amplitude of oscillation reasonably, as can be seen in Figure 37. Also, the intra and intra period length boundaries that are retrieved, still classify the magnitude of the period length in a sensible way, i.e. nearly all periods last for less than around 17 or more than roughly 2 time units as quantified by a 0.84 and higher probability level for each period.
Figure 36: Repressilator inter period length quantification.

Figure 37: Sample trajectory of the Repressilator.
9 Conclusions

To sum up, in this master thesis we propose a model checking based approach to reason about the oscillatory and periodic character of Markovian population models (MPMs), based on continuous time Markov chains (CTMCs), that allow for the encoding of systems from various areas like queuing and computer networks as well as biological reaction networks. We address the problem of defining oscillatory and periodic character in noisy settings by deriving an appropriate notion, *noisy periodicity*, from standard mathematical concepts. This property is then formally translated into Continuous Stochastic Logic (CSL), a prominent specification formalism for CTMCs. For this purpose, the MPM of interest is extended via a product automaton construction to incorporate additional information about the evolution of the supposedly oscillatory quantity. This expansion also allows to reason about the present period lengths. When only interested in sustaining change of some quantity, the standard mathematical notion of oscillation suffices. Therefore, also for this definition, a formally derived CSL translation together with several methods to optimize the corresponding model checking procedure are presented. Finally, a prototypically implemented tool embedded in a fully working tool chain (including the standard probabilistic model checker Prism) is introduced, to automate the task of model checking noisy periodicity as well as oscillatory behavior. This tool chain is then evaluated via two popular case studies, the 3-way Oscillator and the Repressilator.

9.1 Future Work

The presented approach to quantify the period length of noisy periodicity splits that quantity into two parts, the *intra* and *inter* period length, i.e. the time needed for a period and the time between two periods. Since for both, individual upper and lower bounds are calculated, the overall time needed for a cycle is not directly represented. This splitting is an artifact resulting from the period detector expansion. If the total period length would have to be measured at once, the accept state of the period detector automaton would coincide with the start state. Now, the total period length, i.e. the time needed from the start to accept state would be recognized as zero. Although a possible solution has been presented, i.e. connecting two period detectors in sequence and measuring the time between the two start states, an evaluation via case studies is left as future work.

Another issue is to improve the tool’s estimation of reasonable boundary values $b_{low}$ and $b_{high}$ for systems not showing a sinusoidal behavior. A suggestion would be to sample trajectories in order to categorize the behavior and then depending on the category to use different presets, parameterized by statistical quantities derived from the generated trajectories or by model checking (like the expectation and standard deviation of the observation’s steady state distribution).

Moreover, only closed biological systems could be analyzed by now. The reason is that any chemical reaction type involving an inflow of molecules from outside the system, e.g. reaction types like

$$\emptyset \xrightarrow{c} X$$
result in infinite state spaces, since they allow for an unbounded production of molecules (in this example of $X$ molecules). Traditional probabilistic model checking techniques are not applicable to infinite state systems. One way to solve that problem is to restrict model checking oscillatory behavior and noisy periodicity to a certain finite subset of states, that accounts for most of the steady state distribution’s probability mass. For that, the presented CSL formulas have to be carefully translated to hold for the steady state. An advantage of this approach would be that those formulas could also be used to skip the transient phase of the system, which might contain a warm-up phase where steady oscillations of significant amplitude first have to stabilize.

Finally, modeling systems with high molecule levels clearly suffers from state space explosion. A solution could be to model those systems using a hybrid approach, where larger quantities of molecules are treated deterministically and continuously, and lower molecule levels are still treated discretely and stochastic.
A Matlab Routines

A.1 3-way Oscillator ODE

% Three-Way Oscillator ODE solution
function twoode()

global k

k = ones(3,1); % reaction rates, all 1.0

[T,Y] = ode23(@two_ode,[0,20],[15;0;0;1;1;1]);
plot(T,Y);
legend('A','B','C','DA','DB','DC');
end

function dy = two_ode(t,y)

global k

% y = (A; B; C; DA; DB; DC)
dy = zeros(6,1);

dy = [
    -k(1)*y(1)*y(2) + k(3)*y(1)*y(3) - k(1)*y(1)*y(5) + k(3)*y(3)*y(4); % A
    -k(2)*y(2)*y(3) + k(1)*y(1)*y(2) - k(2)*y(2)*y(6) + k(1)*y(1)*y(5); % B
    -k(3)*y(1)*y(3) + k(2)*y(2)*y(3) - k(3)*y(3)*y(4) + k(2)*y(2)*y(6); % C
    0; % DA
    0; % DB
    0]; % DC
end
B Proofs

B.1 Proof of Proposition 3

Proof. In the following proof we assume the (standard) construction presented in Section 6.1.1 to be correct, i.e. given a MPM \( M = (S, R, AP, L) \) with observable \( \mu \) and its delta observation MPM \( M^\Delta = (S', AP', R', L') \) with observable \( \mu^\Delta \), any state \((s, \delta) \in S'\) is labeled absorbing iff \( \exists s' \in S.R(s, s') > 0 \) (equivalently \( \exists (s', \delta') \in S'.R'((s, \delta), (s', \delta')) > 0 \)) due to the construction of \( R' \). Moreover, for any state \((s, \delta) \in S'\) exists a unique predecessor state \((s', \delta') \in S'\) and \( \delta = \mu(s) - \mu(s') \). In order to show

\[ \sigma_s \models_{LTL} \bigwedge_{c \in \mu(S)} \Box \Diamond \neg (\mu = c) \iff \sigma_s^\Delta \models_{LTL} \Box (\neg \text{absorbing} \land \Diamond (\Delta \mu = 0)) \]

we distinguish between finite and infinite sequences. But first we will transform the two formulas, i.e.

\[ \sigma_s \models_{LTL} \bigwedge_{c \in \mu(S)} \Box \Diamond \neg (\mu = c) \iff \forall c \in \mu(S). \forall t. \exists t' \geq t. \sigma_s[t'] \not\models_{LTL} \mu = c \]

\[ \iff \forall t. \forall c \in \mu(S). \exists t' \geq t. \mu(\sigma_s[t']) \neq c \]

and

\[ \sigma_s^\Delta \models_{LTL} \Box (\neg \text{absorbing} \land \Diamond (\Delta \mu = 0)) \]

\[ \iff \forall t. \sigma_s^\Delta[t] \models_{LTL} \neg \text{absorbing} \land \exists t' \geq t. \sigma_s^\Delta[t'] \not\models_{LTL} \Delta \mu = 0 \]

\[ \iff \forall t. \exists s \in S.R(\sigma_s[t], s) > 0 \land \exists t' \geq t. \mu(\sigma_s[t' - 1]) - \mu(\sigma_s[t']) \neq 0 \]

\[ \iff \forall t. \exists s \in S.R(\sigma_s[t], s) > 0 \land \exists t' \geq t. \mu(\sigma_s[t' - 1]) - \mu(\sigma_s[t']) \neq 0 \]

Finite Sequences: We first treat finite sequences, i.e.

\[ \sigma_s = s_0 s_1 \ldots s_n \]

and

\[ \sigma_s^\Delta = (s_0, 0) (s_1, \mu(s_1) - \mu(s_0)) \ldots (s_n, \mu(s_n) - \mu(s_{n-1})) \]

with \( \exists s \in S.R(s_n, s) > 0 \) and \( \exists s' \in S'.R'(s_n, \mu(s_n) - \mu(s_{n-1})), s') > 0 \). In this case, neither \( \Phi \) nor \( \Phi^\Delta \) holds. \( \Phi \) is violated since there is no \( \sigma_s[t'] \) with \( t' \geq t = n \) for any \( c \neq \mu(\sigma_s[t]) \). For that \( t = n \), also \( \Phi^\Delta \) is violated, since \( \exists s \in S.R(\sigma_s[t], s) > 0 \).
**Infinite Sequences:** For infinite sequences $\sigma_s$ and $\sigma_s^\Delta$ we can simplify $\Phi^\Delta$ to $\Phi'^\Delta$, i.e.

$$\forall t. \exists t' \geq t. \mu(\sigma_s[t' - 1]) \neq \mu(\sigma_s[t'])$$

since $\forall t. \exists s \in S.R(\sigma_s[t], s) > 0$ is trivially satisfied.

- Proof $\Phi \Rightarrow \Phi'^\Delta$: Let time step $t_p$ be chosen freely but fixed and let $c = \mu(\sigma_s[t_p])$. Then with $\Phi$ there exists a $t'_p \geq t_p$ s.t. $\mu(\sigma_s[t'_p]) \neq c$. Let $t_m$ be the minimal such $t'_p$, obviously $t'_p > t_p$ for $\mu(\sigma_s[t'_p]) \neq c = \mu(\sigma_s[t_p])$ to hold. Consequently, $\mu(\sigma_s[t_m - 1]) = c \neq \mu(\sigma_s[t_m]) = c'$ and therefore also $\Phi'^\Delta$ holds with $t = t_p$ (which was chosen freely) and choosing $t' = t_m$ like defined above.

- Proof $\Phi'^\Delta \Rightarrow \Phi$: Let time step $t_p$ and observation $c$ be chosen freely but fixed. From $\Phi'^\Delta$ we know that $\exists t'_p \geq t_p + 1. \mu(\sigma_s[t'_p - 1]) \neq \mu(\sigma_s[t'_p])$. Now for $\Phi$ to hold for $t = t_p$ and the chosen $c$, we pick
  - $t' = t'_p - 1$ if $\mu(\sigma_s[t'_p - 1]) \neq c$, and
  - $t' = t'_p$ otherwise.
B.2 Proof of Lemma 1

Proof. Let $\mathcal{R}' = \{(s, s') \in S \times S' | \exists \delta \in \mathbb{R}. s' = (s, \delta)\}$ then

$$\mathcal{R} = \mathcal{R}' \cup \mathcal{R}'^{-1} \cup id(S) \cup \{(s, \delta), (s, \delta')\} | (s, \delta), (s, \delta') \in S'\}$$

where $id(S)$ denotes the identity on $S$, is an $AP, \mu \cup \mu^\Delta$ bisimulation on $\mathcal{M} \cup \mathcal{M}^\Delta$. Let $L^\Delta = L \cup L'$. For checking that all pairs $(s, s')$ of states identified by $\mathcal{R}$, i.e. $(s, s') \in \mathcal{R}$, have the same labeling w.r.t. to $AP$ and coincide on their observation we have to distinguish the cases:

- $(s, s') \in id(S)$: with $s = s'$ we have $L^\Delta(s) = L^\Delta(s)$ and therefore $L^\Delta(s)|_{AP} = L^\Delta(s)|_{AP}$. Obviously, $\mu \cup \mu^\Delta(s) = \mu(s) = \mu(s') = \mu \cup \mu^\Delta(s')$.

- $(s, s') \in \mathcal{R}'$: $s' = (s, \delta)$ for some $\delta \in \mathbb{R}$ and $L^\Delta(s') = L^\Delta(s) \cup A$ with $A = \emptyset$ or $A = \{\text{absorbing}\} \notin AP$. Consequently, we have $L^\Delta(s)|_{AP} = L^\Delta(s) = (L^\Delta(s) \cup \cup A)|_{AP} = L^\Delta(s')|_{AP}$. For the observation we have $\mu \cup \mu^\Delta(s') = \mu^\Delta(s, \delta) = \mu(s) = \mu \cup \mu^\Delta(s')$.

- $(s, s') \in \mathcal{R}'^{-1}$: symmetric to the case above.

- $(s, s') \in \{(s, \delta), (s, \delta')\} \cap (S, \delta), (s, \delta') \in S'$: $s = (s'', \delta)$ and $s' = (s'', \delta')$ for some $s'' \in S$ and $\delta, \delta' \in \mathbb{R}$. Therefore, $L^\Delta(s)|_{AP} = (L^\Delta(s'') \cup A)|_{AP} = L^\Delta(s'') = (L^\Delta(s'') \cup A')|_{AP} = L^\Delta(s'')|_{AP}$ for $A, A' \subseteq \{\text{absorbing}\} \notin AP$. Also, the observations coincide with $\mu \cup \mu^\Delta(s) = \mu \cup \mu^\Delta(s'', \delta) = \mu(s'') = \mu \cup \mu^\Delta(s'', \delta') = \mu \cup \mu^\Delta(s')$.

Moreover, we have to show:

$$\forall (s, s') \in \mathcal{R}. \forall C_t \in S \cup S' \setminus \mathcal{R}. \mathcal{R} \cup \mathcal{R}'(s, C_t) = \mathcal{R} \cup \mathcal{R}'(s', C_t).$$

The respective equivalence classes on $S \cup S'$ w.r.t. $\mathcal{R}$ are

$$C_s = \{s\} \cup \{(s', \delta') \in S' | s = s'\} \text{ for any } s \in S.$$  

With $S \cap S' = \emptyset$ and therefore $\forall s \in S, s' \in S'. \mathcal{R} \cup \mathcal{R}'(s, s') = 0$ we are left to show that

$$\forall (s, s') \in \mathcal{R}. \forall C_t \in S \cup S' \setminus \mathcal{R}. \mathcal{U}(s, C_t|_{\mathcal{T}}) = \mathcal{U}'(s', C_t|_{\mathcal{T}})$$

with

$$T(\cdot) = \begin{cases} S & \text{if } s(\cdot) \in S, \\ S' & \text{otherwise}, \end{cases}$$

and

$$U(\cdot) = \begin{cases} R & \text{if } s(\cdot) \in S, \\ R' & \text{otherwise}. \end{cases}$$

Again, we will use case differentiation on the entries $(s, s') \in \mathcal{R}$. Then, for any $C_t \in S \cup S' \setminus \mathcal{R}$:
• \((s, s') \in \text{id}(S): s \in S \text{ and } s = s', \text{ consequently } s' \in S \text{ and } R(s, C_t|_S) = R(s', C_t|_S).\)
• \((s, s') \in \mathcal{R}': s \in S \text{ and } s' \in S' \text{ with } s' = (s, \delta) \text{ for some } \delta \in \mathbb{R}. \text{ Therefore,}\)
  \[
  R(s, C_t|_S) = R(s, t) = R'((s, \delta), (t, \mu(t) - \mu(s))) = \sum_{\{t, \delta'\in S' | \mu(t) - \mu(s) \neq \delta'\}} R'((s, \delta), (t, \delta')) = R'((s, \delta), \{(t, \delta') | (t, \delta') \in S'\}) = R'((s, \delta), C_t|_{S'}) = R'((s', C_t|_{S'}).\)
• \((s, s') \in \mathcal{R}^{-1}: \text{ symmetric to case above.}\)
• \((s, s') \in \{(s, \delta), (s, \delta') | (s, \delta), (s, \delta') \in S'\}: s = (s'', \delta) \text{ and } s' = (s'', \delta') \text{ for some } s'' \in S \text{ and } \delta, \delta' \in \mathbb{R}. \text{ Then,}\)
  \[
  R'(s, C_t|_{S'}) = R'((s'', \delta), \{(t, \delta'') | (t, \delta'') \in S'\}) = \sum_{(t, \delta'') \in S'} R'((s'', \delta), (t, \delta'')) = R'((s'', \delta), (t, \mu(t)) + \sum_{\{t, \delta'') \in S' | \mu(t) - \mu(s'') \neq \delta''\}} R'((s'', \delta), (t, \delta'')) = R(s'', t).\]
Analogously,
  \[
  R'(s', C_t|_{S'}) = R'((s'', \delta'), C_t|_{S'}) = R(s'', t).\]
For the initial distributions we have that
  \[
  \forall C \in S \cup S' \setminus \mathcal{R}. \alpha(C|_S) = \alpha^\Delta(C|_{S'})
  \]
  \[
  \iff \forall s \in S. \alpha(C_s|_S) = \alpha^\Delta(C_s|_{S'})
  \]
  \[
  \iff \forall s \in S. \alpha(\{s\}) = \alpha^\Delta(\{(s, \delta) | (s, \delta) \in S'\})
  \]
  \[
  \iff \forall s \in S. \alpha(s) = \sum_{(s, \delta) \in S'} \alpha^\Delta(s, \delta)
  \]
  \[
  \text{Def. } \alpha^\Delta \forall s \in S. \alpha(s) = \alpha^\Delta(s, 0) + \sum_{(s, \delta) \in S' \setminus \{(s, 0)\}} \alpha^\Delta(s, \delta) = 0
  \]
  \[
  \iff \text{true.}\]
B.3 Proof of Lemma 2

Proof. Let \( \mathcal{R}' = \{(s, \text{false}), (s, \text{true}) | s \in S\} \), then

\[
\mathcal{R}' \cup \mathcal{R}'^{-1} \cup id(S) \cup \{(s, p), (s, p') | b, b' \in \mathbb{B}, s \in S\}
\]

is an \( AP, \mu \cup \mu^p \) bisimulation on \( M \cup M^p \). Let \( L^p = L \cup L' \), then for equal labeling and coinciding observations of states \((s, s') \in \mathcal{R}\) we have to distinguish the following cases:

- \((s, s') \in id(S)\): \( s = s' \) and \( L^p(s) = L^p(s') \) and also \( L^p(s)|_{AP} = L^p(s')|_{AP} \). Obviously, also \( \mu \cup \mu^p(s) = \mu(s) = \mu(s') = \mu \cup \mu^p(s') \).

- \((s, s') \in \mathcal{R}'\): \( s' = (s, b) \) for \( b \in \mathbb{B} \) and \( L^p(s) = L^p(s) \cup P \cup A \) with \( P \in \{p, -p\} \) depending on \( b \) and \( A = \emptyset \) or \( A = \{\text{absorbing}\} \). In any case \( A, B \not\subseteq AP \) and therefore: \( L^p(s)|_{AP} = L^p(s) = (L^p(s) \cup P \cup A)|_{AP} = L^p(s')|_{AP} \). For the observations we have \( \mu \cup \mu^p(s) = \mu(s) = \mu^p(s, b) = \mu^p(s') = \mu \cup \mu^p(s') \).

- \((s, s') \in \mathcal{R}'^{-1}\): symmetric to case above.

- \((s, s') \in \{(s, b), (s, b') | b, b' \in \mathbb{B}, s \in S\}\): \( s = (s'', b) \) and \( s' = (s'', b') \) for some \( s'' \in S \) and \( b, b' \in \mathbb{B} \). Therefore, \( L^p(s)|_{AP} = (L^p(s'') \cup P \cup A)|_{AP} = L^p(s'') = (L^p(s'') \cup P' \cup A')|_{AP} = L^p(s')|_{AP} \) for some \( A, A' \not\subseteq \{\text{absorbing}\} \not\subseteq AP \) and \( P, P' \in \{p, -p\} \not\subseteq AP \), depending on \( b, b' \). Moreover, the observations coincide with \( \mu \cup \mu^p(s) = \mu^p(s'', b) = \mu(s'') = \mu^p(s'', b') = \mu^p(s') = \mu \cup \mu^p(s') \).

Moreover, we have to show that

\[
\forall (s, s') \in \mathcal{R}, \forall C_t \in S \cup S' \setminus \mathcal{R}. R \cup R'(s, C_t) = R \cup R'(s', C_t).
\]

The respective equivalence classes on \( S \cup S' \) w.r.t. \( \mathcal{R} \)

\[
C_s = \{(s, \text{true}), (s, \text{false})\} \quad \text{for any } s \in S.
\]

With \( S \cap S' = \emptyset \) and therefore \( \forall s \in S, s' \in S'. R \cup R'(s, s') = 0 \) we are left to show that

\[
\forall (s, s') \in \mathcal{R}, \forall C_t \in S \cup S' \setminus \mathcal{R}. U(s, C_t|_T) = U'(s', C_t|_{T'})
\]

with

\[
T(\cdot) = \begin{cases} S & \text{if } s(\cdot) \in S, \\ S' & \text{otherwise,} \end{cases}
\]

and

\[
U(\cdot) = \begin{cases} R & \text{if } s(\cdot) \in S, \\ R' & \text{otherwise.} \end{cases}
\]

Again, we will use case differentiation on the entries \((s, s') \in \mathcal{R}\). Then, for any \( C_t \in S \cup S' \setminus \mathcal{R} \):

- \((s, s') \in id(S)\): \( s = s' \in S \Rightarrow R(s, C_t|_S) = R(s', C_t|_{S'}) \).

79
• \((s, s') \in \mathcal{R}': s \in S\) and \(s' = (s, b)\) for \(b \in \mathbb{B}\). Then,

\[
R(s, C|_S) = R(s, t) = \begin{cases} R(s, t) + 0 & \text{if } p(s, t) \\ 0 + R(s, t) & \text{if } \neg p(s, t) \end{cases}
\]

\[
R'((s, b), (t, \text{true})) + R'((s, b), (t, \text{false})) = R'((s, b), \{(t, \text{true}), (t, \text{false})\}) = R'(s', C|_{S'})
\]

\[
R(s, C|_{S'}) = R(s, t) = \begin{cases} R(s, t) + 0 & \text{if } p(s, t) \\ 0 + R(s, t) & \text{if } \neg p(s, t) \end{cases}
\]

• \((s, s') \in \mathcal{R}': \) symmetric to case above.

• \((s, s') \in \{((s, p), (s, p')) \mid p, p' \in \mathbb{B}, s \in S\}: s = (s'', b)\) and \(s' = (s'', b')\) for some \(b, b' \in \mathbb{B}\). Then,

\[
R'(s, C|_{S'}) = R'((s'', b), \{(t, \text{true}), (t, \text{false})\}) = R'((s'', b), (t, \text{true})) + R'((s'', b), (t, \text{false})) = \begin{cases} R(s, t) + 0 & \text{if } p(s'', t) \\ 0 + R(s, t) & \text{if } \neg p(s'', t) \end{cases}
\]

Analogously, \(R'(s', C|_{S'}) = R(s'', t)\).

For the initial distributions we have that

\[
\forall C \in S \cup S' \setminus \mathcal{R}. \alpha(C|_S) = \alpha(C|_{S'})
\]

\[
\iff \forall s \in S. \alpha(C_s|_S) = \alpha'(C_s|_{S'})
\]

\[
\iff \forall s \in S. \alpha(\{s\}) = \alpha'(\{(s, \text{false}), (s, \text{true})\})
\]

\[
\text{Def} \quad \alpha' \forall s \in S. \alpha(s) = \alpha'(s, \text{false}) + \underbrace{\alpha'(s, \text{true})}_0 = 0.
\]

\(\Box\)
B.4 Proof of Lemma 3

Proof. Let
\[ \sigma_I = I_0 \ I_1 \ldots I_m \]
be the finite sequence of visited intervals of $\mu$-trajectory $\tau$ and
\[ r = r_0 \ r_1 \ldots r_n \]
the corresponding run in the period detector LDFA.

- In order to prove $r \not\models_{LTL} \Box \Diamond start \land \Box (start \Rightarrow \Diamond accept)$ we will distinguish two cases:
  - if $r_n = q_1$, i.e. $r_n \models_{LTL} start$: since $\not\exists r_{i>n}$ and $r_n = q_1 \not\models_{LTL} accept$ we have that $r \not\models_{LTL} \Box (start \Rightarrow \Diamond accept)$, and
  - otherwise if $r_n \neq q_1$, i.e. $r_n \not\models_{LTL} start$: since $\not\exists r_{i>n}$ we have that $r \not\models_{LTL} \Box \Diamond start$.

- Definition 22 is not satisfied for $\sigma_I$, since $\not\exists t \in N_0.\Phi_{enter}(t) \Leftrightarrow \forall t \in N_0.\exists t' > t. (\sigma_I[t'] = I_{low} \land \sigma[t'+1] = I_{mid})$ would also have to hold for $t = m$ but $\not\exists \sigma_I[t']$ with $t' > m$.

\[ \square \]
B.5 Proof of Proposition 6

Proof. We need to show that for any $\mu$-trajectory $\tau$ of $M$ with infinite sequence of visited intervals $\sigma_I$ and and its corresponding (unique) infinite run $r$ in the period detector $\mathcal{D}$, holds

$$\exists t \in \mathbb{N}_0, \Phi_{\text{enter}}(t) \land \forall t \in \mathbb{N}_0, \Phi_{\text{enter}}(t) \Rightarrow \exists t_h > t, \sigma_I[t_h] = I_{\text{high}} \land \exists t_l > t_h, \sigma_I[t_h] = I_{\text{low}}$$

$$\iff r \models \Box \Diamond \text{start} \land \Box (\text{start} \Rightarrow \Diamond \text{accept})$$

with

$$\Phi_{\text{enter}}(t) = \sigma_I[t] = I_{\text{low}} \land \sigma_I[t+1] = I_{\text{mid}}.$$  

We have already shown this equality for finite $\sigma_I$ and $r$ in Lemma 3, since in all those cases, neither the condition of Definition 22 holds for $\sigma_I$ nor is the LTL formula satisfied for the run $r$. Therefore, the reasoning below refers to infinite behavior only. W.l.o.g. let the period detector $\mathcal{D}$ be defined as

$$\mathcal{D} = (\{q_0, q_0', q_b, q_1, q_2, q_3, q_E\}, \{I_{\text{low}}, I_{\text{mid}}, I_{\text{high}}\}, \rightarrow, \text{lab}, L)$$

with the transitions and labelings as depicted in the following graph.

![Period Detector $\mathcal{D}$](image)

First we will show, that

$$(r \models \Box \Diamond \text{start}) \Rightarrow (r \models \Box (\text{start} \Rightarrow \Diamond \text{accept})).$$

Proof:

$$r \models \Box \Diamond \text{start}$$
\[\iff \forall t \exists t' \geq t. r[t'] = \text{start}.\]

With \(q_1\) being the unique state of \(D\) labeled with \text{start} we get
\[\iff \forall t \exists t' \geq t. r[t'] = q_1.\]

Since \(q_1 \not\rightarrow q_1\), we retrieve
\[\iff \forall t \exists t' > t. r[t'] = q_1\]

and especially
\[\Rightarrow \forall t. r[t] = q_1 \Rightarrow \exists t' > t. r[t'] = q_1.\]

Since any path in \(D\) from state \(q_1\) back to \(q_1\) has to cross state \(q_0\), we can conclude
\[\Rightarrow \forall t. r[t] = q_1 \Rightarrow \exists t'' > t. r[t''] = q_0\]

and since \(q_1\) is the unique state labeled with \text{start} and \(q_0\) is uniquely labeled with \text{accept}, we finally retrieve
\[\iff \forall t. r[t] \models \text{start} \Rightarrow \exists t'' > t. r[t''] \models \text{accept}\]

\[\Rightarrow r \models \Box (\text{start} \Rightarrow \Diamond \text{accept}).\]

Please note that this only holds since we have no constraint on the time needed to reach an \text{accept} state from a \text{start} state in LTL, i.e. on the eventually operator. When we quantify that time in Section 6.4 within CSL, this implication does not hold in general.

We will now prove that the error state \(q_E\) of the period detector will never be visited due to our assumptions. Proof: State \(q_E\) can only be visited by

- \(I_{\text{low}} \xrightarrow{\text{high}} \text{high} \xrightarrow{\text{low}}\) transitions, i.e.
  \[\exists t \in \mathbb{N}_0. \sigma_I[t] = I_{\text{low}} \land \sigma_I[t+1] = I_{\text{high}}\]
  \[\Rightarrow \exists s, s' \in S. R(s, s') > 0 \land \mu(s) < b_{\text{low}} \land b_{\text{high}} \geq \mu(s')\]
  Contradiction to \(|b_{\text{low}} - b_{\text{high}}| > \max_{s, s' \in S, R(s, s') > 0} |\mu(s) - \mu(s')|\)

- a single \(\text{high} \xrightarrow{\text{high}}\) transition, when starting in \(q_0\), i.e.
  \[r = s_0 \ x_1 \ldots \text{ with } \mu(s_0) < b_{\text{low}} \land \mu(s_1) \geq b_{\text{high}}\]
  Contradiction to \(|b_{\text{low}} - b_{\text{high}}| > \max_{s, s' \in S, R(s, s') > 0} |\mu(s) - \mu(s')|\)
Proof of the proposition:

• "⇐":

\[ r \models \Box \Diamond \text{start} \land \Box (\text{start} \Rightarrow \Diamond \text{accept}) \]
\[ \Rightarrow r \models \Box \Diamond \text{start} \]
\[ \iff \forall t . \exists t' \geq t . r[t'] \models \text{start} \]

Since the only state in \(D\) labeled with \text{start} is \(q_1\), we may conclude
\[ \iff \forall t . \exists t' \geq t . r[t'] = q_1 \]
and since \(r\) is an infinite sequence and \(q_1 \not\rightarrow q_1\), we get
\[ \Rightarrow \forall t . \exists t' > t . r[t'] = q_1 \quad (\ast). \]

The only predecessors of \(q_1\) are \(q_0\) and \(q'_0\), and both reach \(q_1\) via an \(I_{\text{mid}}\) transitions. Also according to the construction in Section 6.3.1, no state in the initial distribution of the product automaton will directly start in \(q_1\), consequently
\[ \Rightarrow \forall t . \exists t' > t . (r[t' - 1] = q_0 \lor r[t' - 1] = q'_0) \land \sigma_I[t'] = I_{\text{mid}}. \]

The predecessor of \(q_0\) is \(q_3\) and the predecessors of \(q'_0\) are \(q_0\) and \(q'_0\). In any case, state \(q_0\) respectively \(q'_0\) are reached via an \(I_{\text{low}}\) transition, i.e.
\[ \Rightarrow \forall t . \exists t' > t . (r[t' - 2] = q_0 \lor r[t' - 2] = q'_0) \land \sigma_I[t' - 1] = I_{\text{low}} \land \sigma_I[t'] = I_{\text{mid}} \]
\[ \Rightarrow \forall t . \exists t' > t . \sigma_I[t' - 1] = I_{\text{low}} \land \sigma_I[t'] = I_{\text{mid}} \]
\[ \Rightarrow \forall t . \exists t' > t . \sigma_I[t'] = I_{\text{low}} \land \sigma_I[t' + 1] = I_{\text{mid}} \]
\[ \Rightarrow \exists t \in \mathbb{N}_0 . \Phi_{\text{enter}}(t). \]

Moreover, let \(t \in \mathbb{N}_0\) with \(\Phi_{\text{enter}}(t)\) be chosen freely but fixed, we have to show that \(\exists h > t . \sigma_I[t_h] = I_{\text{high}} \land \exists t > t_h . \sigma_I[t_h] = I_{\text{low}}, \) i.e.
\[ \Phi_{\text{enter}}(t) \]
\[ \iff \sigma_I[t] = I_{\text{low}} \land \sigma_I[t + 1] = I_{\text{mid}} \]

The only states that are reachable via \(I_{\text{low}}I_{\text{mid}}\) transitions in \(D\) are \(q_1, q'_1\), and \(q_E\). As already argued, the error state will not be reached, consequently
\[ \Rightarrow r[t + 1] \in \{q_1, q'_1\} \]
and with \((\ast)\) we get
\[ \Rightarrow r[t + 1] \in \{q_1, q'_1\} \land \exists t' > t + 1 . r[t'] = q_1. \]
Let $t' > t + 1$ be the first point in time satisfying $r[t'] = q_1$. Now, any path of $\mathcal{D}$, starting in $q_0$ or $q_1'$, and ending in $q_1$, first has to cross state $q_2$ via an $I_{\text{high}}$ transition and enter state $q_1$ via an $I_{\text{low}}$ transition, i.e.

$$\Rightarrow \exists t, \ t' > t \land \sigma_I[t] = I_{\text{high}} \land \exists t' > t, \sigma_I[t] = I_{\text{low}}$$

$$\Rightarrow \exists t, \ t > t + 1 \land \alpha_I[t] = I_{\text{low}} \land \exists t > t, \alpha_I[t] = I_{\text{low}}.$$ 

Since $t$ with $\Phi_{\text{enter}}(t)$ was chosen freely but fixed, we finally retrieve

$$\Rightarrow \forall t \in \mathbb{N}_0, \Phi_{\text{enter}}(t) \Rightarrow \exists t, \ t > t \land \alpha_I[t] = I_{\text{high}} \land \exists t > t, \alpha_I[t] = I_{\text{low}}.$$ 

\[ \Rightarrow \forall t \in \mathbb{N}_0, \exists t' > t, \ r[t'] \in \{q_1, q_1'\} \quad (1) \]

$$\land \forall t \in \mathbb{N}_0, r[t] \in \{q_1, q_1'\} \Rightarrow \exists h > t, \sigma_I[h] = I_{\text{high}} \land \exists t > t, \sigma_I[t] = I_{\text{low}}. \quad (2)$$

Now, part (2) of that statement can be instantiated for each time point $t'$ inside part (1), i.e.

$$\Rightarrow \forall t \in \mathbb{N}_0, \exists t' > t, \ r[t'] \in \{q_1, q_1'\} \land \exists h > t', \alpha_I[h] = I_{\text{high}} \land \exists t > t, \alpha_I[t] = I_{\text{low}}.$$ 

Obviously, every path of $\mathcal{D}$ that starts in state $q_1$ or $q_1'$ and includes first an $I_{\text{high}}$ transitions and afterwards an $I_{\text{low}}$-transition must cross either state $q_0$ or $q_E$. As already argued, the error state will not be reached, consequently

$$\Rightarrow \forall t \in \mathbb{N}_0, \exists t' > t, \ r[t'] \in \{q_1, q_1'\} \land \exists t > t, \alpha_I[t] = I_{\text{low}}.$$ 

since w.l.o.g. we may assume that this $t_i$ the minimal one. Part (1) also holds for any such $t_i$ and therefore

$$\Rightarrow \forall t \in \mathbb{N}_0, \exists t' > t, \ r[t'] \in \{q_1, q_1'\} \land \exists t > t, \alpha_I[t] = I_{\text{low}}$$

\[ \land \exists t > t, r[t] \in \{q_1, q_1'\}. \]
Due to the structure of the period detector LDFA, in any case, $q_1$ will be visited, consequently

$$\Rightarrow \forall t \in \mathbb{N}_0. \exists t' > t, r[t'] \in \{q_1, q'_1\} \land \exists t_h > t', \sigma_I[t_h] = I_{high} \land \exists t_l > t, \sigma_I[t_l] = I_{low}$$

$$\land r[t_l] = q_0 \land \exists t'_s > t_l, r[t'_s] = q_1.$$

In particular

$$\Rightarrow \forall t \in \mathbb{N}_0. \exists t'' > t, r[t''] = q_1.$$

Again, $q_1$ is the unique state labeled with $start$ and therefore

$$\Leftrightarrow \forall t \in \mathbb{N}_0. \exists t'' > t, r[t''] \models start$$

$$\Leftrightarrow r \models \Box\Diamond start.$$ 

Since $r \models \Box\Diamond start \Rightarrow \Box(start \Rightarrow \Diamond accept)$ as shown before, we finally retrieve

$$\Leftrightarrow r \models \Box\Diamond start \land \Box(start \Rightarrow \Diamond accept).$$

□
B.6 Proof of Lemma 4

Proof. Let \( \mathcal{R}' = \{(s, (s, q)) \mid s \in S, q \in Q\} \), then

\[
\mathcal{R} = \mathcal{R}' \cup \mathcal{R}'^{-1} \cup id(S) \cup \{(s, q), (s, q') \mid q, q' \in Q, s \in S\}
\]

is an \( AP, \mu \cup \mu M^{\otimes D} \) bisimulation on \( \mathcal{M} \cup (\mathcal{M} \otimes \mathcal{D}) \). Let \( L M^{\otimes D} = L \cup L' \), then in order to check for equal labeling and observations of states \((s, s') \in \mathcal{R}\) we have to distinguish the following cases:

- \( (s, s') \in id(S) : s = s' \) and \( L M^{\otimes D}(s) = L M^{\otimes D}(s') \) and also \( L M^{\otimes D}|_{AP}(s) = L M^{\otimes D}|_{AP}(s') \). Trivially, also the observations match, i.e. \( \mu \cup \mu M^{\otimes D}(s) = \mu(s) = \mu(s') = \mu \cup \mu M^{\otimes D}(s') \).

- \( (s, s') \in \mathcal{R}' : s' = (s, q) \) for some \( q \in Q \) and \( L M^{\otimes D}(s')|_{AP} = L'(s)|_{AP} = (L(s) \cup L_{sae})|_{AP} = L(s) = L(s) = L M^{\otimes D}(s)|_{AP} \) for some \( L_{sae} \subseteq \{start, accept, error\} \not\subseteq AP \). Also the observations coincide with \( \mu \cup \mu M^{\otimes D}(s) = \mu(s) = \mu M^{\otimes D}(s, q) = \mu \cup \mu M^{\otimes D}(s') \).

- \( (s, s') \in \mathcal{R}'^{-1} \): symmetric to case above.

- \( (s, s') \in \{(s, q), (s, q') \mid q, q' \in Q, s \in S\} : s = (s'', q) \) and \( s' = (s'', q') \) for some \( q, q' \in Q, s'' \in S \). Consequently, \( L M^{\otimes D}(s)|_{AP} = L'(s'', q)|_{AP} = (L(s'') \cup L_1)|_{AP} = L(s'') = (L(s'') \cup L_2)|_{AP} = L(s'', q') = L M^{\otimes D}(s')|_{AP} \) for \( L_1, L_2 \subseteq \{start, accept, error\} \not\subseteq AP \) depending on \( q \) and \( q' \). Moreover, the observations match with \( \mu \cup \mu M^{\otimes D}(s) = \mu M^{\otimes D}(s'', q) = \mu(s'') = \mu M^{\otimes D}(s'', q') = \mu \cup \mu M^{\otimes D}(s') \).

Moreover, we have to show that

\[
\forall(s, s') \in \mathcal{R}, \forall C_t \in S' \cup S' \setminus \mathcal{R}. R \cup R'(s, C_t) = R \cup R'(s', C_t).
\]

The respective equivalence class on \( S' \cup S' \) w.r.t. \( \mathcal{R} \) are

\[
C_s = \{s\} \cup \{(s, q) \mid q \in Q\} \text{ for any } s \in S.
\]

With \( S \cap S' = \emptyset \) and therefore \( \forall s \in S, s' \in S'. R \cup R'(s, s') = 0 \) we are left to show:

\[
\forall(s, s') \in \mathcal{R}, \forall C_t \in S' \cup S' \setminus \mathcal{R}. U(s, C_t|_{T'}) = U(s', C_t|_{T'})
\]

with

\[
T'(s) = \begin{cases} S & \text{if } s(\cdot) \in S, \text{ and} \\ S' & \text{otherwise,} 
\end{cases}
\]

and

\[
U'(s) = \begin{cases} R & \text{if } s(\cdot) \in S, \text{ and} \\ R' & \text{otherwise.} 
\end{cases}
\]

Again, we will use case differentiation on the entries \( (s, s') \in \mathcal{R} \). Then, for any \( C_t \in S' \cup S' \setminus \mathcal{R} \):
\[
\begin{align*}
\bullet (s, s') &\in id(S): s, s' \in S \text{ with } s = s' \text{ and trivially } R(s, C_t|s) = R(s', C_t|s). \\
\bullet (s, s') &\in R': s \in S \text{ and } s' \in S' \text{ with } s' = (s, q) \text{ for some } q \in Q. \text{ Then, } \\
R(s, C_t|s) &= R(s, t)
\end{align*}
\]

and also
\[
R'(s', C_t|s') = R'((s, q), C_t|s') = R'((s, q), \{t\} \times Q) = \sum_{(t, q') \in \{t\} \times Q} R'((s, q), (t, q')) = *
\]

and since \(D\) is deterministic and non-blocking, there exists a unique \(q_{next} \in Q\) with \(q \xrightarrow{I_n} q_{next}\) for \(\mu(t) \in I_n \in I\), consequently
\[
* = R'((s, q), (t, q_{next})) + \underbrace{\sum_{(t, q') \in \{t\} \times (Q \setminus \{q_{next}\})} R'((s, q), (t, q'))}_{=0} = R(s, t).
\]

\[
\begin{align*}
\bullet (s, s') &\in R'^\prime_{-1}: \text{ symmetric to case above.} \\
\bullet (s, s') &\in \{(s, q), (s, q') \mid q, q' \in Q, s \in S\}: s = (s'', q) \text{ and } s' = (s'', q') \text{ for some } s'' \in S \text{ and } q, q' \in Q. \text{ Then, } \\
R'(s, C_t|s') &= R'((s'', q), \{t\} \times Q) = \sum_{(t, q'') \in \{t\} \times Q} R'((s'', q), (t, q'')) = *
\end{align*}
\]

and since \(D\) is deterministic and non-blocking, there exists a unique \(q_{next} \in Q\) with \(q \xrightarrow{I_n} q_{next}\) for \(\mu(t) \in I_n \in I\). Consequently,
\[
* = R'((s'', q), (t, q_{next})) + \underbrace{\sum_{(t, q'') \in \{t\} \times (Q \setminus \{q_{next}\})} R'((s'', q), (t, q''))}_{=0} = R(s'', t).
\]

Analogously,
\[
R'(s', C_t|s') = R'((s'', q'), (C_t|s')) = R(s'', t).
\]

For the initial distributions we have that
\[
\forall C \in S \cup S' \setminus R. \alpha(C|s) = \alpha^{M \otimes D}(C|s') \\
\iff \forall s \in S. \alpha(C_s|s) = \alpha^{M \otimes D}(C_s|s') \\
\iff \forall s \in S. \alpha(\{s\}) = \alpha^{M \otimes D}(\{s\} \times Q) \\
\iff \forall s \in S. \alpha(s) = \sum_{(s, q) \in s \times Q} \alpha^{M \otimes D}(s, q) \iff \ast \ast.
\]

88
Let $q_{\text{start}} \in Q$ be the period detector state $q_0$ if $\mu(s) \in I_{\text{low}} \in I$, $q'_1$ if $\mu(s) \in I_{\text{mid}} \in I$ and $q_2$ otherwise. Then,

\[
\text{Def. } Q^\otimes D \implies \forall s \in S, \alpha(s) = \alpha^M \otimes D(s, q_{\text{start}}) + \sum_{(s, q) \in \{s\} \times (Q \setminus \{q_{\text{start}}\})} \alpha^M \otimes D(s, q)
\]

\[= \alpha(s) + \sum_{(s, q) \in \{s\} \times (Q \setminus \{q_{\text{start}}\})} \alpha^M \otimes D(s, q) = 0 \iff \text{true.}
\]
C XML descriptions

C.1 3-way Oscillator without doping

```xml
<?xml version="1.0"?>
<system name="ThreeWayOscillator">
  <species>
    <discrete name="A" max="3n" init="n" interest="yes"/>
    <discrete name="B" max="3n" init="n"/>
    <discrete name="C" max="3n" init="n"/>
  </species>

  <rates>
    <rate name="rA" value="1.0"/>
    <rate name="rB" value="1.0"/>
    <rate name="rC" value="1.0"/>
  </rates>

  <reactions>
    <!-- NORMAL REACTION TYPES -->
    <!--    A+B -rA-> 2B  -->
    <reaction>
      <reactants>
        <reactant name="A" count="1"/>
        <reactant name="B" count="1"/>
      </reactants>
      <rate name="rA"/>
      <products>
        <product name="B" count="2"/>
      </products>
    </reaction>

    <!--    B+C -c2-> 2C  -->
    <reaction>
      <reactants>
        <reactant name="B" count="1"/>
        <reactant name="C" count="1"/>
      </reactants>
      <rate name="rB"/>
      <products>
        <product name="C" count="2"/>
      </products>
    </reaction>

    <!--    C+A -rC-> 2A  -->
    <reaction>
      <reactants>
        <reactant name="C" count="1"/>
        <reactant name="A" count="1"/>
      </reactants>
      <rate name="rC"/>
      <products>
        <product name="A" count="2"/>
      </products>
    </reaction>
  </reactions>
</system>
```
C.2 3-way Oscillator with doping

```xml
<?xml version="1.0"?>
<system name="ThreeWayOscillator">
    <species>
        <discrete name="A" max="3n" init="n" interest="yes"/>
        <discrete name="B" max="3n" init="n"/>
        <discrete name="C" max="3n" init="n"/>
        <discrete name="DA" max="1" init="1"/>
        <discrete name="DB" max="1" init="1"/>
        <discrete name="DC" max="1" init="1"/>
    </species>

    <rates>
        <rate name="rA" value="1.0"/>
        <rate name="rB" value="1.0"/>
        <rate name="rC" value="1.0"/>
    </rates>

    <reactions>
        <!-- NORMAL REACTION TYPES -->
        <!--    A+B -rA-> 2B    -->
        <reaction>
            <reactants>
                <reactant name="A" count="1"/>
                <reactant name="B" count="1"/>
            </reactants>
            <rate name="rA"/>
            <products>
                <product name="B" count="2"/>
            </products>
        </reaction>
        <!--    B+C -c2-> 2C    -->
        <reaction>
            <reactants>
                <reactant name="B" count="1"/>
                <reactant name="C" count="1"/>
            </reactants>
            <rate name="rB"/>
            <products>
                <product name="C" count="2"/>
            </products>
        </reaction>
        <!--    C+A -rC-> 2A    -->
        <reaction>
            <reactants>
                <reactant name="C" count="1"/>
                <reactant name="A" count="1"/>
            </reactants>
            <rate name="rC"/>
            <products>
                <product name="A" count="2"/>
            </products>
        </reaction>
    </reactions>
</system>
```
...<![endif]-- DOPING -->
<![endif]-- DA+C -rC-> A+DA -->
<reaction>
  <reactants>
    <reactant name="DA" count="1" />
    <reactant name="C" count="1" />
  </reactants>
  <rate name="rC" />
  <products>
    <product name="A" count="1" />
    <product name="DA" count="1" />
  </products>
</reaction>
<![endif]-- DB+A -rA-> B+DB -->
<reaction>
  <reactants>
    <reactant name="DB" count="1" />
    <reactant name="A" count="1" />
  </reactants>
  <rate name="rA" />
  <products>
    <product name="B" count="1" />
    <product name="DB" count="1" />
  </products>
</reaction>
<![endif]-- DC+B -rB-> C+DC -->
<reaction>
  <reactants>
    <reactant name="DC" count="1" />
    <reactant name="B" count="1" />
  </reactants>
  <rate name="rB" />
  <products>
    <product name="C" count="1" />
    <product name="DC" count="1" />
  </products>
</reaction>
</reactions>
</system>
C.3 Repressilator

<?xml version="1.0"?>
<system name="Repressilator">
  <species>
    <discrete name="A" max="10" init="0" interest="yes"/>
    <discrete name="B" max="10" init="0"/>
    <discrete name="C" max="10" init="0"/>
    <discrete name="GA" max="1" init="0"/>
    <discrete name="GB" max="1" init="0"/>
    <discrete name="GC" max="1" init="0"/>
  </species>
  <rates>
    <rate name="kp" value="5.0"/>
    <rate name="kb" value="5.0"/>
    <rate name="ku" value="1.0"/>
    <rate name="kd" value="1.2"/>
  </rates>
  <reactions>
    <!-- PRODUCTION -->
    <!--    GA -kp-> GA + A   -->
    <reaction>
      <reactants>
        <reactant name="GA" count="1"/>
      </reactants>
      <rate name="kp"/>
      <products>
        <product name="GA" count="1"/>
        <product name="A" count="1"/>
      </products>
    </reaction>
    <!--    GB -kp-> GB + B   -->
    <reaction>
      <reactants>
        <reactant name="GB" count="1"/>
      </reactants>
      <rate name="kp"/>
      <products>
        <product name="GB" count="1"/>
        <product name="B" count="1"/>
      </products>
    </reaction>
    <!--    GC -kp-> GC + C   -->
    <reaction>
      <reactants>
        <reactant name="GC" count="1"/>
      </reactants>
      <rate name="kp"/>
      <products>
        <product name="GC" count="1"/>
        <product name="C" count="1"/>
      </products>
    </reaction>
    <!-- DEGRADATION -->
    <!--    A -kd-> 0   -->
    <reaction>
      <reactants>
        <reactant name="A" count="1"/>
      </reactants>
      <rate name="kd"/>
    </reaction>
    <!--    B -kd-> 0   -->
    <reaction>
      <reactants>
        <reactant name="B" count="1"/>
      </reactants>
      <rate name="kd"/>
    </reaction>
  </reactions>
</system>
<reaction>
  <reactants>
    <reactant name="A" count="1" />
  </reactants>
  <rate name="kd" />
</reaction>

<reaction>
  <reactants>
    <reactant name="B" count="1" />
  </reactants>
  <rate name="kd" />
</reaction>

<reaction>
  <reactants>
    <reactant name="C" count="1" />
  </reactants>
  <rate name="kd" />
</reaction>

<reaction>
  <reactants>
    <reactant name="C" count="1" />
    <reactant name="GA" count="1" />
  </reactants>
  <rate name="kb" />
  <products>
    <product name="C" count="1" />
  </products>
</reaction>

<reaction>
  <reactants>
    <reactant name="A" count="1" />
    <reactant name="GB" count="1" />
  </reactants>
  <rate name="kb" />
  <products>
    <product name="A" count="1" />
  </products>
</reaction>

<reaction>
  <reactants>
    <reactant name="B" count="1" />
    <reactant name="GC" count="1" />
  </reactants>
  <rate name="kb" />
  <products>
    <product name="B" count="1" />
  </products>
</reaction>

<reaction>
  <rate name="ku" />
  <products>
    <product name="GA" count="1" />
  </products>
</reaction>
List of Formulas

Formula 1 (Non-Convergence) ................................................. 28
Formula 2 (Noisy Periodicity) ................................................. 30
Formula 3 (Non-Convergence for Delta Observation MPMs) ........... 32
Formula 4 (Non-Convergence for Change Predicate expanded MPMs) ... 38
Formula 5 (Infinitely many period starts for Noisy Periodicity) .......... 46
Formula 6 (Quantification of overall intra period length) ................. 47
Formula 7 (Quantification of maximal intra period length) ............... 48
Formula 8 (Quantification of minimum intra period length) .............. 48
Formula 9 (Quantification of overall inter period length) ............... 49
Formula 10 (Quantification of maximum inter period length) .......... 49
Formula 11 (Quantification of minimum inter period length) ........... 49
List of Tables

1  3-way Oscillator (no doping) Prism models for initial molecule level $n$ of each species $A$, $B$, and $C$, where $\ast$ means that absorbing states have been detected during Prism’s model building phase. . . . . . . . . . . . . . . . . 64
2  3-way Oscillator (with doping) Prism models for initial molecule level $n$ of each species $A$, $B$, and $C$ . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 64
3  3-way Oscillator (with doping) model checking times and results. . . . . . 65
4  Repressilator Model Sizes . . . . . . . . . . . . . . . . . . . . . . . . . . . 70
List of Figures

1. A simple CTMC. .................................................. 5
2. Plot of a sample $\mu$-trajectory. .......................... 13
3. Plot of the solution to the (doped) 3-way Oscillator ODE with initial concentration $15.0 \text{ mol l}^{-1}$ of species $A$. .................. 16
4. Plot of the solution to the (doped) 3-way Oscillator ODE with initial concentration $5.0 \text{ mol l}^{-1}$ of each species, $A$, $B$ and $C$. ........... 16
5. Plot of a sample trajectory projected onto the molecule level of species $A$ of the (doped) 3-way Oscillator generated with Prism. The initial amount of molecules is 5 for each of the species $A$, $B$ and $C$, and one for the doping substances. ................................................................. 17
6. PRISM Model of a simple oscillator between the two states $c = 0$ and $c = 1$ ........................................ 20
7. Prism experiment: Probability of leaving state $s_0$ within $t$ time units. ........................................... 20
8. Transient probabilities of being in state $c = 0$ respectively $c = 1$ at time $t$. .................................... 21
9. Examples of a convergent $\mu$-trajectory (left) and a divergent $\mu$-trajectory (right). ................................. 22
10. Example of an oscillatory $\mu$-trajectory ....................... 23
11. Example of a periodic $\mu$-trajectory with period $\lambda$. ...... 23
12. Example of an oscillatory but non periodic $\mu$-trajectory (top) and a periodic but non oscillatory $\mu$-trajectory (bottom). .......... 24
13. Example of a perfectly oscillatory and periodic $\mu$-trajectory (blue) and a noisy version of it (red). .......................... 25
14. Example of a noisy periodic $\mu$-trajectory. ..................... 26
15. Oscillatory behavior (left) and noisy periodicity (right). ........ 31
16. A $\mu$-trajectory with the difference in observation $\Delta \mu$ plotted at jump points. ................................. 33
17. Example of MPM delta observation expansion. ............ 35
18. Correspondence of the presented logical characterizations of oscillation. ........................................ 36
19. Period Detector LDFA .................................................. 41
20. Incorrect smaller Period Detector LDFA (problems colored red). .................................................. 42
21. Correspondence of the presented logical characterizations of oscillation. ........................................ 44
22. A sample $\mu$-trajectory period with its intra and inter period length. .................................................. 47
23. Several neuronal spikes with their interevent times $t_i$. ........ 48
24. Quantification of the intra and inter period lengths. .......... 50
25. An overview of the tool chain. .................................... 53
26. Basic structure of Prism models generated by the tool. ........ 56
27. Change Detector Prism module. .................................... 58
28. Sinusoidal $\mu$-trajectory (left); $\mu$-trajectory with periodic peaks (right). ........................................ 60
29. $\mu$-trajectory with periodic peaks and corrected boundary values. ............................................. 60
30. Period Detector Prism module. ..................................... 62
31. Estimating the $\text{intra}$ period length boundary window $[T_{\text{intra,min}}, T_{\text{intra,max}}]$ for probability bound $p_{\text{intra}}$ via initial constraint $p$ on individual boundaries $T_{\text{intra,min}}$ and $T_{\text{intra,max}}$. .......... 66
<table>
<thead>
<tr>
<th>Page</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>Estimating the <em>inter</em> period length boundary window ([T_{\text{inter,min}}, T_{\text{inter,max}}]) for probability bound (p_{\text{inter}}) via initial constraint (p) on individual boundaries (T_{\text{inter,min}}) and (T_{\text{inter,max}}).</td>
</tr>
<tr>
<td>33</td>
<td>3-way Oscillator sample (\mu)-trajectory showing the (A) molecule level over time.</td>
</tr>
<tr>
<td>34</td>
<td>Repressilator steady state distribution of species (A)'s molecule level.</td>
</tr>
<tr>
<td>35</td>
<td>Repressilator intra period length quantification.</td>
</tr>
<tr>
<td>36</td>
<td>Repressilator inter period length quantification.</td>
</tr>
<tr>
<td>37</td>
<td>Sample trajectory of the Repressilator.</td>
</tr>
</tbody>
</table>
References


