Conventional Verification for Unconventional Computing: a Genetic XOR Gate Example

Savas Konur\textsuperscript{1 C}
Marian Gheorghe\textsuperscript{1}
Ciprian Dragomir\textsuperscript{1}
Florentin Ipate\textsuperscript{2}
Natalio Krasnogor\textsuperscript{3}

\textsuperscript{1}Department of Computer Science, University of Sheffield, Sheffield, UK
\{s.konur, m.gheorghe, c.dragomir\}@sheffield.ac.uk
\textsuperscript{2}Department of Computer Science, University of Bucharest, Bucharest, Romania
florentin.ipate@ifsoft.ro
\textsuperscript{3}School of Computing Science, Newcastle University, Newcastle, UK
natalio.krasnogor@newcastle.ac.uk

Abstract. As unconventional computation matures and non-standard programming frameworks are demonstrated, the need for formal verification will become more prevalent. This is so because “programming” in unconventional substrates is difficult. In this paper we show how conventional verification tools can be used to verify unconventional programs implementing a logical XOR gate.

1. Introduction

Unconventional computing, with many aspects including implementations in vivo, vitro and silico, models and methods, programming paradigms and tools, is a rapidly growing research area with results, promises and huge hope in building new computational devices and tools for solving better or/and faster
increasingly complex problems than current machines, models and tools, which either produce inefficient results or are just unable to solve them.

One specific class of models and experiments related to unconventional computing, often called natural computing, is inspired by natural processes occurring in biology or produces in vitro (DNA strands) or in vivo (bacteria) experiments simulating different computational devices. A thorough account of the developments in the area can be found in [25], but we also mention some specific demonstrations of unconventional computing using liposomes [27], programmable polymers [18] and photochromic molecules [8].

XOR gate is a classic computer science concept with various unconventional computing incarnations. More recently some implementations have been provided [28, 21] and solutions using synthetic biology computational tools have been proposed [4].

In this paper, we aim to reconsider this problem and to provide a set of unconventional computing models based on the P systems computational paradigm [19]. Here, we consider stochastic P systems [23] and kernel P systems [12] as representative classes of such models. These models are associated with verification methods using model checking approaches.

The key contributions of the paper are: the introduction of a set of unconventional models based on P systems, which naturally describe the genetic XOR gate problem, and the use of some model checkers for verifying properties of the models. This approach is complementary to the previous investigations and highlights new perspectives for investigating these systems.

2. Stochastic and Non-deterministic P Systems: Basic Concepts and Tools

Membrane computing [19] is a branch of natural computing inspired by the hierarchical structure of the living cell. The central model, called P system, consists of a membrane structure, the regions of which contain rewriting rules operating on multisets of objects [19]. The P system evolves by repeatedly applying rules, mimicking chemical reactions and transportation across membranes or cellular division or death processes, and halts when no more rules can be applied. The most recent developments in this field are reported in [20].

The closeness of this model to the biology makes it highly suited as a specification vehicle for representing biological systems, especially (multi-)cellular systems and molecular interactions taking place in different locations of living cells [11]. Different simple molecular interactions or more complex gene expressions, compartment translocation, as well as cell division and death are specified using multiset rewriting or communication rules, and compartment division or dissolution rules. In the case of stochastic P systems, constants are associated with rules in order to compute their probabilities to be applied, respectively, according to the Gillespie algorithm [24]. This approach is based on a Monte Carlo algorithm for stochastic simulation of molecular interactions taking place inside a single volume or across multiple compartments.

Definition 2.1. A stochastic P system (SP system) is a model consisting of a tissue P system with stochastic semantics [24]:

\[
SP = (O, L, \mu, M_1, \ldots, M_n, R_{1}, \ldots, R_{n})
\]

where \(O\) is a finite set of objects, called alphabet, denoting the entities involved in the system; \(L\) is a
finite set of *labels* naming compartments; $\mu$ is a membrane structure composed of $n \geq 1$ membranes defining the regions or compartments of the system and their connections, forming an arbitrary graph; $M_i = (l_i, w_i), 1 \leq i \leq n$, is the initial configuration of the compartment or region defined by the membrane $i$, where $l_i \in L$ is the label of the compartment and $w_i \in O^\ast$ is a finite initial multiset of objects; $R_i = \{r_{i1}^i, \ldots, r_{im_i}^i\}, 1 \leq i \leq n$, is a set of multiset rewriting rules, of the form: $r_{ik}^i : \begin{array}{c} \{ x \} \\ \xrightarrow{c_k} \{ y \} \end{array} l_i$, where $x$ and $y$ are multisets of objects ($y$ might be empty) over $O$, representing the molecular species consumed and produced in the corresponding molecular interaction occurring in the compartment labelled $l_i$. An application of a rule of this form changes the content of the membrane with label $l_i$ by replacing the multiset $x$ with $y$. The stochastic constant $c_k$ is used by the Gillespie algorithm [14] in order to compute the probabilities associated with the rules [24].

Stochastic P systems have been used as a basis for a specification language [24, 7] and applied, among others, in unconventional computing, e.g. liposomes [27] and the genetic AND/OR gates [26], as well as typical synthetic biology constructs, e.g. a pulse generator [6], and system biology constructs, e.g. quorum sensing.

**Note 1.** Stochastic P systems also include communication rules, but these are not discussed in this paper as the system that we deal with consists of one single compartment without communication rules. In this case, the label of the compartment will be also dropped.

Certain systems can be modelled with P systems which do not require probabilistic features. In [11] some types of P systems without probabilities are presented. These variants are utilised for specifying biological systems. **Kernel P systems (kP systems)** have been introduced as a unifying framework allowing to express within the same formalism many classes of non-deterministic P systems [12, 10, 2].

**Note 2.** In this paper, we use kernel P systems only for very limited purposes, obtaining them directly from the stochastic ones and making use of some tools deployed for them. The kP systems derived from SP systems use the same alphabet and rules without kinetic constants. In general, each kP system model has explicitly defined execution strategies for its components. For the system considered in this paper the execution strategy consists of executing one single rule per step, non-deterministically chosen from the set of rules that can be applied.

The **Infobiotics Workbench (IBW)** tool [6, 7] has been built for modelling and prototyping biological systems exhibiting molecular interactions. It allows to define such systems using the above mentioned formalism, SP systems, providing support for the simulation, verification, analysis and optimisation of these models. The experiments to be discussed later have been performed using IBW. The XOR gate will be modelled using SP systems and then simulated and formally verified. The formal verification is performed with third party tools, PRISM [16] and MC2 [9] (integrated in IBW). The corresponding non-deterministic model represented as a kP systems model will be verified using SPIN [17].

PRISM [16] is a very popular and widely used probabilistic model checker. It allows probabilistic properties, supporting PCTL [15] (a probabilistic extension of computation tree logic) and Continuous Stochastic Logic (CSL [1]). Both languages make use of special operators to express quantitative information which is useful for a precise, fine-grained analysis. The property languages also allow defining *reward*-based properties to describe quantitative expressions. PRISM suffers from the same problem exhibited by all model checkers, namely state space explosion and consequently cannot cope with very large state spaces. This is overcome by an alternative model checking approach, **statistical model checking**.

MC2 [9] is a statistical model checker, where properties are analysed against a finite set of simu-
lation traces using statistical methods, e.g. Monte Carlo. Unlike symbolic and numerical methods, e.g. those employed in PRISM, statistical model checkers do not analyse the system exhaustively, which increases the performance significantly. In MC2, properties are expressed using PLTLc [9], a probabilistic extension of linear temporal logic with constraints. PLTLc allows properties with some functions returning maximum/minimum values of a species and “derivative of the concentration of species at each time point” [9].

The kP Workbench (kPW) tool [10] has been built to support kP systems formalism, allowing simulation [3] and formal verification [2]. It uses a specific language, based on kP systems, kP-lingua, allowing to specify non-deterministic rule-based systems. The formal verification is performed using a model checking approach based on SPIN, which is incorporated into the framework. The models written in kP-lingua are automatically translated into SPIN. The non-deterministic version of the XOR gate model will be specified using kP systems and the formal verification will be provided in SPIN.

SPIN [17] is a widely used model checking tool with many applications in concurrent and distributed systems verification. A high level modelling language, PROMELA, suitable for describing concurrent processes and interprocess communication, is at the core of this tool. SPIN provides complete support for Linear-time Temporal Logic (LTL) and on the fly verification procedures which avoid the necessity to generate the global state space prior to performing a search.

3. XOR Gate and Unconventional Models

In this section we consider a genetic XOR logic gate. This gate was presented in various papers, including [28, 21]. The construction used in this paper is taken from [4] where it is defined in GEC, a language for synthetic biology. In this paper, we will use the IBW tool, which differs from the tools used in [4] in that it allows multicellular modelling and analysis, and integrates third-party model checking tools to support verification.

The XOR gate expresses the green fluorescent protein (GFP) if either of aTc or IPTG molecules are present, but not both. Figure 1 illustrates the genetic construction and the corresponding network.

The XOR device comprises two mechanisms. Each mechanism leads to the production of GFP, when it is activated; but two mechanisms cannot be activated at the same time. Namely, while one is active, the other one is inhibited by some protein.

In this system, the transcription factors LacI and TetR are expressed by a gene controlled by the same promoter. The LacI and TetR proteins work in the opposite way. LacI represses the first mechanism, but promotes the other one. On the other hand, TetR promotes the first mechanism, while inhibiting the second. In other words, both proteins serve as inhibitor and promoter in a complementary fashion. In each mechanism, while one protein is an inhibitor, the other one is promoter. When either of the proteins works as an inhibitor, it binds to the corresponding promoter which upregulates the expression of GFP. The XOR device receives two input signals: aTc and IPTG. The aTc and IPTG signals bind to TetR and LacI, respectively, to prevent them from interacting with the promoters producing GFP.

Two mechanisms together ensure that the production of GFP will be low when both input signals are set to very low or very high concentrations at the same time. In the former case, LacI and TetR will be produced in abundance, which will then repress the GFP expression. In the latter case, the LacI and TetR concentration will be very low, which is not sufficient to express GFP. On the other hand, if one signal is set to high and the other one is set to low, the device will produce high amount of GFP, hence will act as
3.1. Stochastic model

The stochastic model comprises a single compartment with initial concentrations of aTc and IPTG molecules and a set of SP system rules, which govern the kinetic and stochastic behaviour of the system. The initial values are those illustrated in Figure 3. The rewriting rules and the kinetic constants, provided in Table 1, describe the model shown in [4]. The label of the compartment is ignored as we have only one compartment – see Note 1.

A gene controlled by the same promoter expresses LacI and TetR (rules $r_1$ to $r_3$). Rules $r_{7a}$ and $r_{15a}$ describe the inhibition of the two mechanisms leading to the GFP production by binding to the promoters that upregulates the production process; $r_{7b}$ and $r_{15b}$ define the debinding process. The activation of the first mechanism by the transcription factor TetR binding to the promoter and the activation of the second mechanism by LacI, are modelled by rules $r_{9a}, r_{10}, r_{11}$ and $r_{13a}, r_{14}, r_{17}$, respectively. Rules $r_{9b}$ and $r_{13b}$ are debinding reactions. Rules $r_4$ and $r_5$ define the binding process involving LacI and IPTG and TetR and aTc, respectively. The degradation process of various molecular species is defined by rules $r_{18}$ to $r_{23}$.

Given certain initial values for aTc and IPTG, different output values are obtained for the GFP products, as shown in Table 1.

3.2. Non-deterministic model

The rules of the non-deterministic model are obtained directly from the set of SP system rules given in Table 1, by removing the kinetic constants; the alphabets of these two systems are the same, in accordance with Note 2. Some of the rules that do not contribute to the model, with kinetic constants equal to 0.
<table>
<thead>
<tr>
<th>Rule</th>
<th>Reaction</th>
<th>Stochastic constant</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_1$</td>
<td>$\text{gene}<em>{\text{LacI}} \text{TetR} \xrightarrow{k_1} \text{gene}</em>{\text{LacI}} \text{TetR} + \text{protein}_{\text{LacI}} \text{TetR}$</td>
<td>$k_1 = 0.12$</td>
</tr>
<tr>
<td>$r_2$</td>
<td>$\text{protein}<em>{\text{LacI}} \text{TetR} \xrightarrow{k_2} \text{protein}</em>{\text{LacI}} \text{TetR} + \text{LacI}$</td>
<td>$k_2 = 0.1$</td>
</tr>
<tr>
<td>$r_3$</td>
<td>$\text{protein}<em>{\text{LacI}} \text{TetR} \xrightarrow{k_3} \text{protein}</em>{\text{LacI}} \text{TetR} + \text{TetR}$</td>
<td>$k_3 = 0.1$</td>
</tr>
<tr>
<td>$r_4$</td>
<td>$\text{LacI} + \text{IPTG} \xrightarrow{k_4} \text{LacI-IPTG}$</td>
<td>$k_4 = 1.0$</td>
</tr>
<tr>
<td>$r_5$</td>
<td>$\text{TetR} + \text{aTc} \xrightarrow{k_5} \text{TetR-aTc}$</td>
<td>$k_5 = 1.0$</td>
</tr>
<tr>
<td>$r_6$</td>
<td>$\text{gene}<em>{\text{GFP1}} \xrightarrow{k_6} \text{gene}</em>{\text{GFP1}} + \text{protein}_{\text{GFP1}}$</td>
<td>$k_6 = 0$</td>
</tr>
<tr>
<td>$r_7a$</td>
<td>$\text{gene}<em>{\text{GFP1}} + \text{LacI} \xrightarrow{k_7a} \text{gene}</em>{\text{GFP1-LacI}}$</td>
<td>$k_7a = 1.0$</td>
</tr>
<tr>
<td>$r_7b$</td>
<td>$\text{gene}<em>{\text{GFP1-LacI}} \xrightarrow{k_7b} \text{gene}</em>{\text{GFP1-LacI}} + \text{protein}_{\text{GFP1}}$</td>
<td></td>
</tr>
<tr>
<td>$r_8$</td>
<td>$\text{gene}<em>{\text{GFP1}} + \text{TetR} \xrightarrow{k_8a} \text{gene}</em>{\text{GFP1-TetR}}$</td>
<td>$k_8a = 0.5$</td>
</tr>
<tr>
<td>$r_9a$</td>
<td>$\text{gene}<em>{\text{GFP1}} + \text{TetR} \xrightarrow{k_9a} \text{gene}</em>{\text{GFP1-TetR}}$</td>
<td>$k_9a = 0.5$</td>
</tr>
<tr>
<td>$r_9b$</td>
<td>$\text{gene}<em>{\text{GFP1}} - \text{TetR} \xrightarrow{k_9b} \text{gene}</em>{\text{GFP1}} + \text{TetR}$</td>
<td>$k_9b = 1.0$</td>
</tr>
<tr>
<td>$r_{10}$</td>
<td>$\text{gene}<em>{\text{GFP1}} - \text{TetR} \xrightarrow{k</em>{10}} \text{gene}<em>{\text{GFP1}} - \text{TetR} + \text{protein}</em>{\text{GFP1}}$</td>
<td>$k_{10} = 0.1$</td>
</tr>
<tr>
<td>$r_{11}$</td>
<td>$\text{protein}<em>{\text{GFP1}} \xrightarrow{k</em>{11}} \text{protein}_{\text{GFP1}} + \text{GFP}$</td>
<td>$k_{11} = 0.1$</td>
</tr>
<tr>
<td>$r_{12}$</td>
<td>$\text{gene}<em>{\text{GFP2}} \xrightarrow{k</em>{12}} \text{gene}<em>{\text{GFP2}} + \text{protein}</em>{\text{GFP2}}$</td>
<td>$k_{12} = 0$</td>
</tr>
<tr>
<td>$r_{13a}$</td>
<td>$\text{gene}<em>{\text{GFP2}} + \text{LacI} \xrightarrow{k</em>{13a}} \text{gene}_{\text{GFP2-LacI}}$</td>
<td>$k_{13a} = 1.0$</td>
</tr>
<tr>
<td>$r_{13b}$</td>
<td>$\text{gene}<em>{\text{GFP2-LacI}} \xrightarrow{k</em>{13b}} \text{gene}_{\text{GFP2-LacI}} + \text{LacI}$</td>
<td>$k_{13b} = 0.5$</td>
</tr>
<tr>
<td>$r_{14}$</td>
<td>$\text{gene}<em>{\text{GFP2-LacI}} \xrightarrow{k</em>{14}} \text{gene}<em>{\text{GFP2-LacI}} + \text{protein}</em>{\text{GFP2}}$</td>
<td>$k_{14} = 0.1$</td>
</tr>
<tr>
<td>$r_{15a}$</td>
<td>$\text{gene}<em>{\text{GFP2}} + \text{TetR} \xrightarrow{k</em>{15a}} \text{gene}_{\text{GFP2-TetR}}$</td>
<td>$k_{15a} = 1.0$</td>
</tr>
<tr>
<td>$r_{15b}$</td>
<td>$\text{gene}<em>{\text{GFP2}} - \text{TetR} \xrightarrow{k</em>{15b}} \text{gene}_{\text{GFP2}} + \text{TetR}$</td>
<td>$k_{15b} = 0.01$</td>
</tr>
<tr>
<td>$r_{16}$</td>
<td>$\text{gene}<em>{\text{GFP2}} - \text{TetR} \xrightarrow{k</em>{16}} \text{gene}<em>{\text{GFP2}} - \text{TetR} + \text{protein}</em>{\text{GFP2}}$</td>
<td>$k_{16} = 0.0$</td>
</tr>
<tr>
<td>$r_{17}$</td>
<td>$\text{protein}<em>{\text{GFP2}} \xrightarrow{k</em>{17}} \text{protein}_{\text{GFP2}} + \text{GFP}$</td>
<td>$k_{17} = 0.1$</td>
</tr>
<tr>
<td>$r_{18}$</td>
<td>$\text{GFP} \xrightarrow{k_{18}}$</td>
<td>$k_{18} = 0.01$</td>
</tr>
<tr>
<td>$r_{19}$</td>
<td>$\text{LacI} \xrightarrow{k_{19}}$</td>
<td>$k_{19} = 0.01$</td>
</tr>
<tr>
<td>$r_{20}$</td>
<td>$\text{TetR} \xrightarrow{k_{20}}$</td>
<td>$k_{20} = 0.01$</td>
</tr>
<tr>
<td>$r_{21}$</td>
<td>$\text{protein}<em>{\text{GFP1}} \xrightarrow{k</em>{21}}$</td>
<td>$k_{21} = 0.001$</td>
</tr>
<tr>
<td>$r_{22}$</td>
<td>$\text{protein}<em>{\text{GFP2}} \xrightarrow{k</em>{22}}$</td>
<td>$k_{22} = 0.001$</td>
</tr>
<tr>
<td>$r_{23}$</td>
<td>$\text{protein}<em>{\text{LacI}} \text{TetR} \xrightarrow{k</em>{23}}$</td>
<td>$k_{23} = 0.001$</td>
</tr>
</tbody>
</table>
(r_6, r_8, r_{12}, r_{16}), are completely removed. The initial values are kept the same as in the stochastic case. The rules are executed in a non-deterministic manner, as described earlier in this paper – see Note 2.

This non-deterministic model allows to describe all chains of reactions, observe various interactions between species and determine various dependencies between molecules. It will be used in this respect as the basic model for qualitative analysis. As in this case we are interested in an efficient behaviour of the system, some simplifications will be made to the model, whereby the number of molecules will be bounded.

4. Experiments

In this section, we will provide a computational analysis to infer the system dynamics of the genetic XOR gate. This approach complements previous in vitro or in silico implementations of this gate [28, 21, 4] with a set of qualitative and quantitative properties and results. The stochastic model introduced in Section 3.1 and non-deterministic model from Section 3.2 will be used as specifications for the experiments that follow. We note that the complete model and experimental results of the XOR gate can be accessed at [30].

Before presenting the experiments, we briefly explain the main symbols (F, G and U) of the logic formulae. F p states that p holds in some future state; G p states that p always holds in all future states; and p U q states that p holds until q holds. Probabilistic properties are expressed using the formulas of the type $P_{\mathcal{D} \ll r} [\phi]$, which informally means that the probability of taking a branch satisfying $\phi$ meets $\mathcal{D} \ll r$ (where $\mathcal{D} \in \{<, >, =, \leq, \geq\}$ and $r \in [0, 1]$). The PRISM model checker also features reward-based formulas to calculate expected values. For example, the reward formula $R\{"X"\}_\mathcal{D} [I = t]$ calculates the expected value of X at the time instant $t$.

4.1. Non-deterministic Model

The non-deterministic model, discussed in Section 3.2 and obtained from the SP system described in Table 1, will form the basis for translation into SPIN. The experiments made and reported in this section refer to relationships between species occurring on various reaction pathways. First we verify generic relationships between species. The property

“The GFP is preceded by the production of at least one of LacI or TetR”

is formally expressed as

$$F (\text{GFP} > 0) \rightarrow \neg((\text{LasR} = 0 \land \text{TetR} = 0) \lor \text{GFP} > 0),$$

and the result of this property is true.

The Boolean gate is implemented in such a way that the presence of at least one of the input signal molecules, aTc or IPTG, at a certain concentration, leads to the production of the GFP. It turns out that in the non-deterministic case, the GFP production does not depend in any way of these signal molecules, but it is regulated by the proteins TetR and LacI. We can be more specific with respect to the above relationships and refer to the production of a transcription factor and its role as a repressor. More specifically, we verify the property.
“When there is no TetR in the system and the LacI represses gene_GFP1 then GFP is produced only by the activation of gene_GFP2”

which is formally expressed as

\[
F (\text{GFP} > 0 \land \text{gene\_GFP1-LacI} > 0) \rightarrow \neg((\text{TetR} = 0 \land \text{gene\_GFP2-LacI} = 0) \\
U (\text{GFP} > 0 \land \text{gene\_GFP1-LacI} > 0)).
\]

The result of this property is true. We can formulate a similar property for TetR acting as a repressor on gene_GFP2.

### 4.2. Stochastic Model

For the stochastic analysis, we have constructed a system model based on SP systems, the modelling language of the IBW system, using the set of rules discussed in Section 3.1. Below, we summarise some of the experiments that we have carried out using the computational tools integrated into IBW.

#### 4.2.1. Simulation results

Figure 2 illustrates the simulation results of the XOR system, performed using the IBW’s Mcss tool, a simulator for multi-compartment SP system models [23]. IBW provides a GUI to view the simulation results in various formats, e.g. time series, bars, histograms and 3D heat-map animations.
Figure 3: Expected amount of some species based on different initial amounts of aTc and IPTG.

Figure 2 comprises the screenshots of the 3D animation at different time instants. At the top and bottom corners of the grid both input signals (i.e. aTc and IPTG) are simultaneously set to very low (i.e. 0) and very high concentrations (i.e. 10000), respectively. Meanwhile, at the left and right corners, one signal is set to very high while the other one is set to very low. As illustrated in the figure, only left and right corners yield a sharp increase in the GFP concentration, ensuring that the designed circuit shows an XOR gate behaviour.

4.2.2. Model checking results

As discussed above, IBW also permits formal verification of a system using model checking techniques. Since the SP systems allow modelling stochastic models, IBW uses probabilistic model checking tools, currently PRISM and MC2.
**Figure 4:** Probability that GFP exceeds the threshold.

**PRISM results**

We first analyse the amounts of different species over time with four combinations of inputs. The informal property

“*What is the expected concentration of X at the time instant t?*”

is formally expressed as a reward-based formula,

\[ R \{ X \} = ? [ I = t ] . \]

Figure 3 illustrates the expected amounts of GFP, aTc, LacI and TetR. The results obtained are similar to that of [4]. As shown in Figure 3a, the input combinations aTc=0 – IPTG=10000 and aTc=10000 – IPTG=0 result in a sharp increase in the GFP concentration, whereas the combinations aTc=0 – IPTG=0 and aTc=10000 – IPTG=10000 cause the GFP concentration to stay in low levels, confirming the behaviour of the XOR gate.

Figure 3b shows that if the aTc concentration level is initially set to high, the concentration reduces until it becomes 0. As can be seen in Figure 3d, aTc suppresses the TetR protein by binding to it. After aTc molecules are totally consumed, the TetR concentration starts increasing. We can observe a similar behaviour to Figure 3b, when the IPTG concentration is set to high. IPTG molecules suppress the LacI protein as shown in Figure 3c. These results are in line with the system behaviour, described in Section 3.1.

We now measure the likelihood that the GFP concentration exceeds a certain threshold in any input combination. The property

“*What is the probability that GFP exceeds Thr within t seconds?*”

is formally expressed as

\[ P = ? [ F \leq t \text{ GFP} > \text{Thr} ] . \]

Figure 4 illustrates the probability values calculated for a threshold value of 100 over different time instants. Clearly, it is almost certainly that GFP exceeds the threshold value for the input combinations aTc=0 – IPTG=10000 and aTc=10000 – IPTG=0. This confirms the desired behaviour.
We now consider a more complex property. Assume that GFP\(_{ij}\) (where \(i, j \in \{0, 1\}\) represents the state of αTc and IPTG, respectively) denotes the GFP concentration for different input combinations. Namely, if \(i=0\) (resp. \(j=0\)), then αTc=0 (resp. IPTG=0), and if \(i=1\) (resp. \(j=1\)), then αTc=10000 (resp. IPTG=10000). Then, the property

**What is the probability that GFP\(_{01}\) and GFP\(_{10}\) are at least \(k_1\) times more than GFP\(_{00}\) and GFP\(_{11}\), and GFP\(_{01}\) is within the range of GFP\(_{10}\) ± \(\frac{k_2}{10}\)?**

is formally specified as

\[
P_{=?}[F \leq t] \quad \text{GFP}_{\text{in}} \geq k_1 \ast \text{GFP}_{\text{in}} \land \text{GFP}_{\text{in}} \geq k_1 \ast \text{GFP}_{\text{in}} \land \text{GFP}_{\text{in}} \geq k_1 \ast \text{GFP}_{\text{in}} \land \text{GFP}_{\text{in}} \geq (1 - \frac{k_2}{10}) \ast \text{GFP}_{\text{in}} \land \text{GFP}_{\text{in}} \leq (1 + \frac{k_2}{10}) \ast \text{GFP}_{\text{in}}].
\]

Figure 5 shows the plot based on different \(k_1\) and \(k_2\) values. As expected, the probability becomes higher when \(k_1\) is lower and \(k_2\) is higher, because the formula becomes less strict.

**MC2 results**

We now consider a property describing the behaviour in Figure 3a. We want to query

**“What is the probability that GFP\(_{10}\) reaches a concentration level of at least \(l_1\) times more than the maximum concentrations of GFP\(_{00}\) and GFP\(_{11}\); the concentration then starts decreasing and reduces until it becomes one \(l_2\) ration of its maximum level.”**
This formula is expressed in PLTL_ε as follows:

\[ P = \neg F \left( [GFP_{10}] \geq l_1 \cdot \max[GFP_{00}] \land [GFP_{10}] \geq l_1 \cdot \max[GFP_{11}] \land (F (d[GFP_{10}] \leq 0 \land (F [GFP_{10}] \leq \max[GFP_{10}] / l_2))) \right). \]

We have analysed the property for \( l_1 = l_2 = 5 \), and the probability value returned is 1.0, confirming the behaviour in Figure 3a. We can verify the same property for GFP_{01}, which returns the same result.

5. Conclusions

In this paper we have presented a stochastic P systems model and a non-deterministic one for specifying and studying the behaviour of a genetic XOR gate. These two models are formally analysed using model checking methods revealing qualitative aspects, like expected chain of reactions and dependencies of various species, as well as the quantitative aspects regarding the concentration of certain products with respect to the amount of signal molecules, time to reach certain concentration of molecules or comparisons between maximum concentration achieved for certain species. Our approach is orthogonal to many other unconventional computational investigations or implementations of genetic Boolean gates.

In this line of research, we aim to expand to some other unconventional models, starting with the two genetic XOR gate approaches already mentioned in [28, 21]. We aim also to clarify better the role of various model checkers and types of properties with respect to various systems.

In [29] an interesting prediction and programmability problem for non-DNA molecular self-assembly using porphyrin tiles is investigated. In one of the investigated cases, the self-assembly process is defined as a simple two state probabilistic automaton. This model can be directly represented in PRISM and properties regarding the distribution of the two colours on each diagonal or across the lattice are verified. In a forthcoming paper we will be investigating this problem in more detail.

Acknowledgements.

SK and MG acknowledge the EPSRC support (grant number: EP/I031812/1) support; NK’s work is supported by EPSRC (grant numbers: EP/I031642/1, EP/J004111/1, EP/L001489/1). MG and FI are partially supported by CNCS UEFISCDI (grant number: PN-II-ID-PCE-2011-3-0688). CD acknowledges an EPSRC studentship.

References


