Modelling and Analysis of Genetic Boolean Gates Using the INFOBIOTICS WORKBENCH

Savas Konur[1], Christophe Ladroue[2], Harold Fellermann[3], Daven Sanassy[3], Laurentiu Mierla[4], Florentin Ipate[4], Sara Kalvala[2], Marian Gheorghe[1], and Natalio Krasnogor[3]

[1] *Department of Computer Science, University of Sheffield, UK*
{s.konur,m.gheorghe}@sheffield.ac.uk

[2] *Department of Computer Science, University of Warwick, UK*
{c.ladroue,sara.kalvala}@warwick.ac.uk

[3] *School of Computing Science, Newcastle University, UK*
{harold.fellermann,d.sanassy,natalio.krasnogor}@newcastle.ac.uk

[4] *Department of Computer Science, University of Bucharest*
laurentiu.mierla@gmail.com,florentin.ipate@ifsoft.ro

Abstract

In this paper, we illustrate the use of the INFOBIOTICS WORKBENCH platform, designed to model and analyse stochastic P systems, by modelling basic synthetic Boolean gates and analysing them using some computational techniques: simulation, verification and biocompilation.

1 Introduction

Membrane computing [1] is a branch of natural computing inspired by the hierarchical structure of living cells with various compartments inside them, or the network of cells occurring in tissues and organs, and key functions describing molecular interactions of species and macromolecules. It emphasises the compartmentalised nature of biological systems, and supports the study of the computational power, complexity and efficiency of its models, and deals with their applications in various fields.

We base our system on a particular model in membrane computing called a *P system*, consisting of a membrane structure and rewriting rules operating on multisets of objects [1]. P systems mimic chemical reactions and transportation across membranes or cellular division or death processes by repeatedly applying rules until rules cannot be applied anymore. P systems provide a clear mapping of different regions and compartments of a biological systems into membranes. Each molecular species is associated with an object in the multiset corresponding to a membrane mapping the region or compartment where the molecule is located. Since P systems are close to biology, they are a suitable formalism for representing biological systems, especially (multi-)cellular systems and molecular interactions taking place in different locations of living cells [2].

In order to model different systems, several versions of P systems have been proposed. *Stochastic P (SP) systems* [3] are a probabilistic extension of P systems, where constants are associated with rules in order to compute their probabilities and execution time, respectively, according to the Gillespie algorithm [4]. SP systems are a natural, intuitive and amenable formalism, capturing stochastic dynamics of biological and chemical systems [5]. The INFOBIOTICS WORKBENCH (IBW) tool is a software platform designed to model and analyse stochastic P systems. IBW permits applying various computational techniques, such as simulation and verification.

In this paper, we illustrate how IBW utilises these techniques in the analysis of biological systems, in particular synthetic biology models. In addition, we will present our initial results on the automatic construction of genetic devices using a *biomatter compilation* module. We will illustrate our approach on two basic genetic devices: the AND and OR gates.

We note that the goal of this work is *in silico* modelling and analysis. Both *in vivo* and *in vitro* aspects are outside of the scope of this paper.

2 INFOBIOTICS WORKBENCH

The IBW tool [6, 7] enables prototyping systems and synthetic biology models exhibiting molecular interactions. The tool provides support for modeling, simulation, verification and optimisation of SP system models.

2.1 Modeling

In IBW, system models are constructed using stochastic P systems, augmented with a two-dimensional lattice representation to capture the spatial aspect of a biological system and the structure of membranes distributed geometrically within it. The geometrical representation of membranes permits modelling molecular exchange between adjacent cells.

IBW provides a dedicated DSL (Domain Specific Langauge) for SP systems, called *LPP* systems, supported by a graphical model editor. The language is very modular in the sense that modules and libraries can be reused by different SP system models, and multiple copies of SP systems can be distributed in different parts of a geometrical lattice, which facilitates the modelling of bacterial colonies containing different types of cells.

2.2 Simulation

IBW implements a stochastic simulation algorithm, called MCSS [5], based on a multi-compartmental extension of the Gillespie algorithm [4]. The compartmental nature of the algorithm allows performing simulations without a need to flatten models that have multiple cells distributed geometrically.

IBW displays simulation results in various formats, *e.g.* time series, histograms, bars, 3D heatmaps and animations, using a GUI. Users can view selected compartments and species based on all or selected simulation runs. Users can also configure the data units of the model components.

2.3 Verification

IBW's verification component, called PMODELCHECKER, performs model checking using third party model checkers. Since SP models are stochastic, PMODELCHECKER currently employs the probabilistic model checking tools PRISM [8] and MC2 [9].

PRISM [8] is a widely used tool, developed to model check probabilistic systems, e.g. Continuous Time Markov Chains. In PRISM, properties are expressed using probabilistic temporal logics,

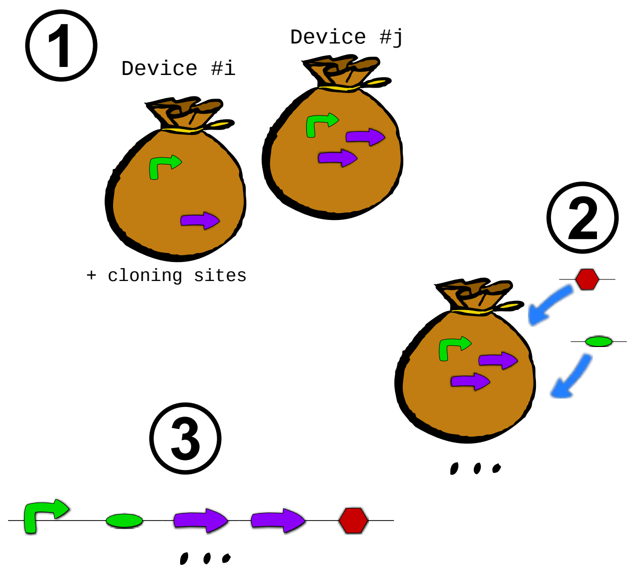


Figure 1: The biocompiler's workflow. From unstructured sets of parts and user requisites, to complete devices to viable DNA sequences.

e.g. Continuous Stochastic Logic (CSL) [10], which allows expressing quantitative information useful for a precise and fine grain analysis.

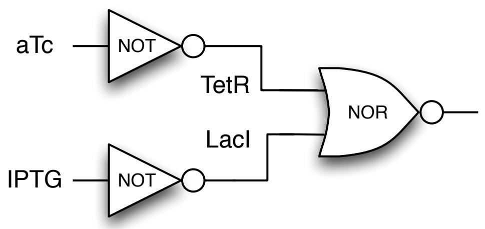
MC2 [9] is a *statistical* model checking tool, which evaluates properties against a set of simulation runs using statistical methods, e.g. Monte Carlo. Unlike numerical methods, this method prevents an *exhaustive* analysis, increasing the performance significantly. In addition to probabilistic aspect, MC2 also permits expressing quantities regarding 'maximum/minimum' values and 'derivative' of species' concentrations.

2.4 Biocompilation

Through simulation and verification, the user is able to design an *in silico* construct that meets their criteria. The next step is then to test the design *in vivo* in an actual organism. We are currently developing a *biomatter compilation* module, to be integrated into the new version of IBW (in progress). It combines known biology, a database of genetic parts, and user knowledge, to automatically build a viable DNA sequence that can be used in organism.

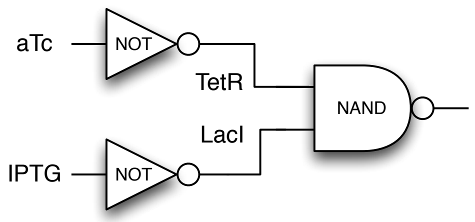
Once the user has specified the functional parts for the construct (*e.g.* promoters, protein coding sequences), the biocompiler uses built-in genetics knowledge to add parts not necessary for the design but mandatory for genetic sequences (*e.g.* RBS, spacers), as well as restriction enzyme sites for future experiments. All the parts are then arranged so as to make biological sense, through a mapping to a constraint solving problem. Each part is assigned a position (an integer) and all biological (*e.g.* non-overlapping devices, parts order within a device given their types) or user-defined requisites are translated into a integer programming problem. We use the Java library JaCoP ([11]) to find an optimal arrangement.

The user can also provide in-house knowledge to add constraints on the resulting sequence, for example enforcing the direction of a device or the relative position of parts. This is done through a very simple language (ATGC, for Assistant To Genetic Compilation [12]).



aTc	IPTG	GFP
0	0	0
0	1	0
1	0	0
1	1	1

(a) AND gate



aTc	IPTG	GFP
0	0	0
0	1	1
1	0	1
1	1	1

(b) OR gate

Figure 2: The logic AND and OR gates.

For example, if the user wants to enforce the relative positions of two promoters, they use the command:

```
ATGC ARRANGE Promoter2 Promoter1
```

They can also ask for a device to the implemented in the reverse direction:

```
ATGC myDevice DIRECTION REVERSE
```

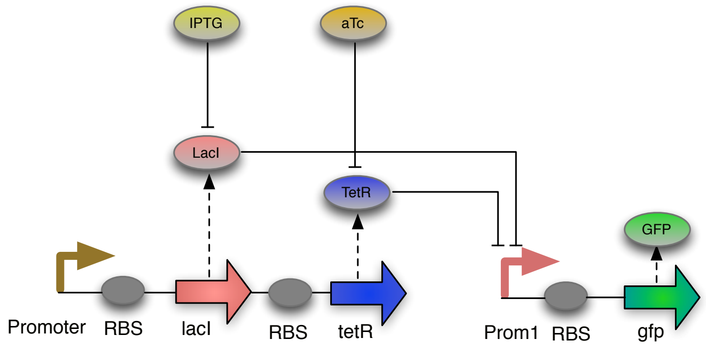
In this case, the parts are re-arranged so that the device starts with a terminator and ends with a promoter, and the sequences are reverse-complemented.

Figure 1 describes the compiler's work flow. From an unstructured set of parts and user-defined constraints (1), the biocompiler completes the devices with extra parts (2) and finds an optimal arrangement (3). The part sequences are found in public databases ([13, 14, 15]) or provided by the user.

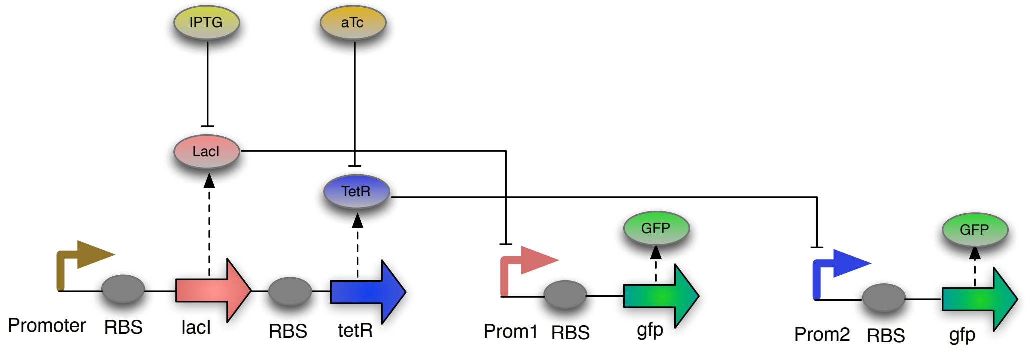
3 Two Genetic Logic Gates

Synthetic Boolean logic gates have been studied in various papers, including [16, 17, 18]. The devices discussed in this paper are constructed using the genetic parts of the XOR gate designed in [16], where the analysis of the circuits was done using the synthetic biology tools GEC [19], Eugene [20] and Proto [21]. In this paper, we will use the IBW tool, which differs from these tools in that it allows multicellular modelling and analysis, and integrates third-party model checking tools to support verification.

Here, we consider two basic logic gates: AND and OR, whose logic diagrams and truth tables are given in Figure 2. Both gates use two inducers, `aTc` and `IPTG`, as input (provided in the beginning) and use `GFP` as output. `aTc` and `IPTG` disable the activities of `TetR` and `LacI` proteins, respectively. The genetic designs of the gates are presented in Figure 3.



(a) AND gate



(b) OR gate

Figure 3: The genetic devices functioning as an AND and OR gate.

Figure 3a illustrates a genetic AND gate, which receives two input signals: `aTc` and `IPTG`. In this system, the transcription factors `LacI` and `TetR` are expressed by a gene controlled by the same promoter. The `aTc` molecules repress `TetR`, and `IPTG` molecules repress `LacI`, to prevent them from inhibiting the production of `GFP` by binding to the corresponding promoter which upregulates the expression of `GFP`. If both `IPTG` and `aTc` are set to high, then neither `LacI` nor `TetR` can inhibit the `GFP` production.

Figure 3b illustrates a genetic OR gate, comprising two mechanisms. Each mechanism leads to the production of `GFP`, when it is activated. The first mechanism is repressed by `LacI` while the second is repressed by `TetR`. Therefore, `GFP` can be produced from the former when `IPTG` is set to high and from the latter when `aTc` is set to high.

The stochastic model comprises a set of SP system rules, governing the kinetic and stochastic behaviour of the system. Table 1[1] presents the rewriting rules and the kinetic constants (taken from [16]) of the devices described above. If we consider the AND gate, Rules r_1 to r_3 describe the expression the `LacI` and `TetR` proteins from `gene_LacI_TetR`, regulated by the same promoter.

[1]Here, we assume that `LacI` can't bind `PLac` while `TetR` is bound to `PTet` and vice versa, and that one repressor blocks recruitment at both sites.

Table 1: Kinetic rules for the Boolean gates.

(a) AND gate

Rule		**Kinetic constant**
r_1 :	gene_LacI_TetR $\xrightarrow{k_1}$ gene_LacI_TetR + mRNA_LacI_TetR	$k_1 = 0.12$
r_2 :	mRNA_LacI_TetR $\xrightarrow{k_2}$ mRNA_LacI_TetR + LacI	$k_2 = 0.1$
r_3 :	mRNA_LacI_TetR $\xrightarrow{k_3}$ mRNA_LacI_TetR + TetR	$k_3 = 0.1$
r_4 :	LacI + IPTG $\xrightarrow{k_4}$ LacI-IPTG	$k_4 = 1.0$
r_5 :	TetR + aTc $\xrightarrow{k_5}$ TetR-aTc	$k_5 = 1.0$
r_{6a} :	gene_GFP + LacI $\xrightarrow{k_{6a}}$ gene_GFP-LacI	$k_{6a} = 1.0$
r_{6b} :	gene_GFP-LacI $\xrightarrow{k_{6b}}$ gene_GFP + LacI	$k_{6b} = 0.01$
r_{7a} :	gene_GFP + TetR $\xrightarrow{k_{7a}}$ gene_GFP-TetR	$k_{7a} = 1.0$
r_{7b} :	gene_GFP-TetR $\xrightarrow{k_{7b}}$ gene_GFP + TetR	$k_{7b} = 0.01$
r_8 :	gene_GFP $\xrightarrow{k_8}$ gene_GFP + GFP	$k_8 = 1.0$
r_9 :	GFP $\xrightarrow{k_9}$	$k_9 = 0.001$
r_{10} :	LacI $\xrightarrow{k_{10}}$	$k_{10} = 0.01$
r_{11} :	TetR $\xrightarrow{k_{11}}$	$k_{11} = 0.01$
r_{12} :	mRNA_LacI_TetR $\xrightarrow{k_{12}}$	$k_{12} = 0.001$

(b) OR gate

Rule		**Kinetic constant**
$r_1 - r_5$	same as the rules $r_1 - r_5$ of the AND gate	
r_{6a} :	gene_GFP1 + LacI $\xrightarrow{k_{6a}}$ gene_GFP1-LacI	$k_{6a} = 1.0$
r_{6b} :	gene_GFP1-LacI $\xrightarrow{k_{6b}}$ gene_GFP1 + LacI	$k_{6b} = 0.01$
r_{7a} :	gene_GFP2 + TetR $\xrightarrow{k_{7a}}$ gene_GFP2-TetR	$k_{7a} = 1.0$
r_{7b} :	gene_GFP2-TetR $\xrightarrow{k_{7b}}$ gene_GFP2 + TetR	$k_{7b} = 0.01$
r_8 :	gene_GFP1 $\xrightarrow{k_8}$ gene_GFP1 + GFP	$k_8 = 1.0$
r_9 :	gene_GFP2 $\xrightarrow{k_9}$ gene_GFP2 + GFP	$k_9 = 1.0$
$r_{10} - r_{13}$	same as the rules $r_9 - r_{12}$ of the AND gate	

Rules r_4 and r_5 describe the binding of LacI and IPTG and TetR and aTc, respectively. Rules r_{6a} and r_{6b} describe the inhibition activity of LacI, i.e. its binding to the promoter that upregulates the GFP production. Rules r_{7a} and r_{7b} define the same process for TetR. Rule r_8 describes the expression of GFP. Rules r_9 to r_{12} define the degradation process of various molecular species.

Model Specification

The LPP language allows rules to be grouped into higher level units, called modules. A simplified expression of a protein can be defined as

```
simpleProteinExpression({X,Y},{c_1},{l}) =
     {
     rules:
     r1: [ gene_X ]_l -c_1 -> [ gene_X + Y ]_l
     }
```

and similarly the expression of two proteins is given as

```
constitutiveProteinExpressionTwoGenes({X,Y},{c_1,c_2,c_3},{l}) =
     {
     rules:
     r1: [ gene_X_Y ]_l -c_1-> [ gene_X_Y + mRNA_X_Y ]_l
     r2: [ mRNA_X_Y ]_l -c_2-> [ mRNA_X_Y + X ]_l
     r3: [ mRNA_X_Y ]_l -c_3-> [ mRNA_X_Y + Y ]_l
     }
```

In a similar way there are defined modules for protein binding and debinding and protein degradation. With these generic modules, the AND gate, for instance, can be defined using modules for the expression of `LacI` and `TetR`, for binding `LacI` to `IPTG`, `TetR` to `aTc`, `LacI` to `gene_GFP`, `TetR` to `gene_GFP`, and debinding, the expression of `GFP` and finally degradation reactions for `GFP`, `LacI`, `TetR` and `mRNA_LacI_TetR`. The OR gate is defined in a similar way. The complete formalization including the auxiliary modules can be found online[2].

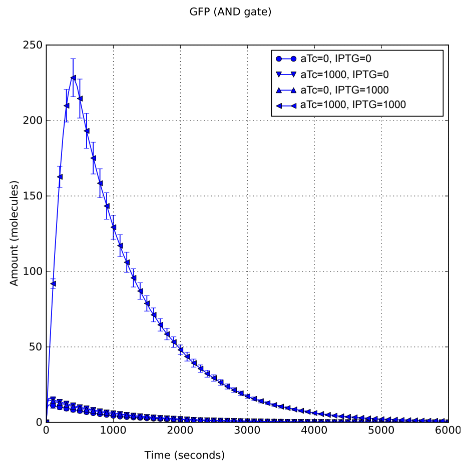
4 Analysis

In this section, we present the results of various analyses, which can be used to infer whether the devices function according to their desired behaviour. The stochastic model based on Table 1 will be considered as specifications for the experiments to follow. The complete models and experimental results can be accessed online[2].

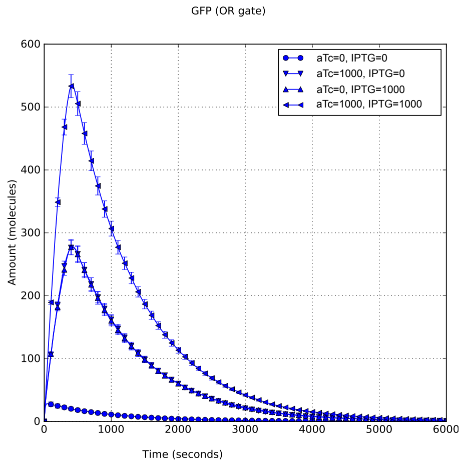
4.1 Simulation

Figure 4 illustrates the simulation results (based on 100 simulation runs), plotting the (mean) `GFP` amount over time for the stochastic models describing an AND and OR gate. The simulation results are obtained using IBW's simulator, MCSS, which can also display results as (3D) heat-map animations for a better visualisation. Figure 5 illustrates the same simulation results as a heat-map (the snapshot was taken at the time point 500 seconds). In the figure, the corners of the lattice represent different Boolean combinations. Namely, top-left represents `aTc=0` and `IPTG=0`; top-right represents `aTc=0` and `IPTG=1000`; bottom-left represents `aTc=1000` and `IPTG=0`; and bottom-right represents `aTc=1000` and `IPTG=1000`. As can be seen from Figures 4 and 5 , the genetic AND and OR devices behave like an AND and OR gate, respectively.

[2] `http://www.dcs.shef.ac.uk/~konur/models/genetic-gates`

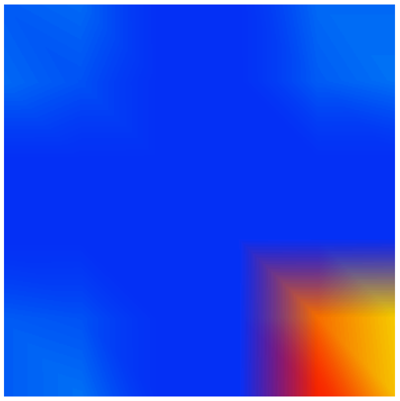


(a) AND gate

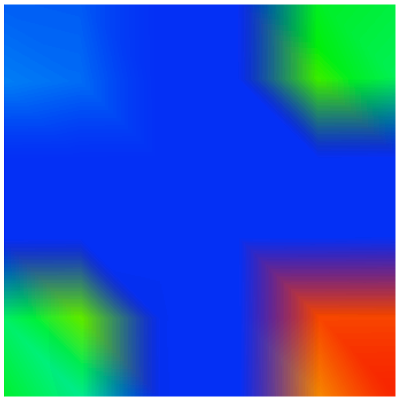


(b) OR gate

Figure 4: Simulation results of `GFP` amount over time for the stochastic models describing an AND and OR gate.



(a) AND gate



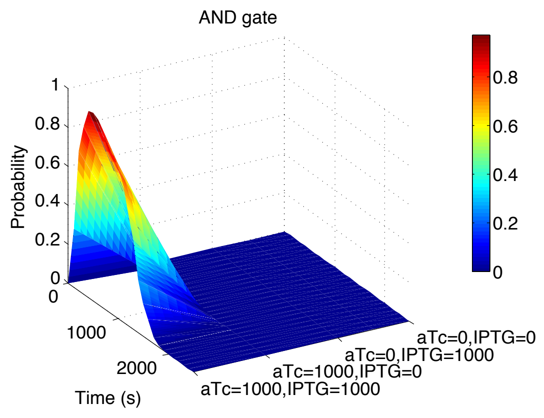
(b) OR gate

Figure 5: Heat-map visualisation of the simulation results.

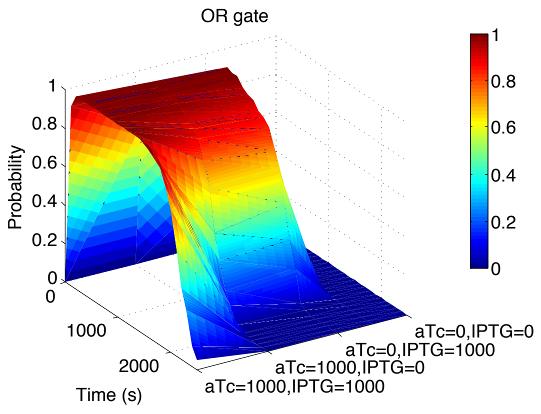
4.2 Verification

IBW allows users to perform formal verification, using model checking. By verifying a formal property against a formal model, we can *exhaustively* analyse the likelihood that the system satisfies the system requirements. IBW currently integrates the PRISM and MC2 model checkers. IBW automatically translates SP system models to the input format that the model checking tools requires. To facilitate the construction of formal properties we have developed a natural language query (NLQ) tool [6], providing assistance to non-expert users to build logical properties from a set of natural language queries.

In standard logic gates, a threshold value at the input to a logic gate determines whether a particular input is interpreted as 0 or 1. For example, any voltage value above 3 V is considered as



(a) AND gate



(b) OR gate

Figure 6: Verification results.

1. Since there is not such standard threshold values for genetic gates, we choose a value for this particular design. To analyse the behaviour of the genetic devices formally, we verify the following property using PRISM:

> *"What is the probability that* GFP *exceeds Thr at time t?"*

which is expressed in CSL as

$$\mathrm{P}_{=?}\left[true\ \mathrm{U}^{[t,t]}\ \mathtt{GFP} \geq Thr\right].$$

Figure 6 shows the model checking results for a threshold value of 100 over the time points up to 2500 seconds. The results clearly confirm the desired behaviour.

Instead of a threshold value, we can also compare the relative GFP concentrations to observe the behaviour. Assume that $\mathtt{GFP}_{ij}$ denotes the GFP concentration for different input combinations. Namely, if i=0 (resp. j=0), then aTc=0 (resp. IPTG=0), and if i=1 (resp. j=1), then aTc=1000 (resp. IPTG=1000). Then, the property

> *"What is the probability that* $\mathtt{GFP}_{11}$ *is at least 5 times more than* $\mathtt{GFP}_{10}$, $\mathtt{GFP}_{01}$ *and* $\mathtt{GFP}_{00}$*?"*

is formally expressed as

$$\mathrm{P}_{=?}\left[true\ \mathrm{U}^{[t,t]}\ (\mathtt{GFP}_{11} \geq 5.\mathtt{GFP}_{10} \wedge \mathtt{GFP}_{11} \geq 5.\mathtt{GFP}_{01} \wedge \mathtt{GFP}_{11} \geq 5.\mathtt{GFP}_{00})\right].$$

For this query, we have obtained 0.96 (for $t = 500$ seconds), confirming the desired behaviour.

4.3 Biocompilation

Figure 7 shows the result of the biocompilation from the specifications decided upon during the design stage. The only functional parts specified by the user are promoters and genes. No extra constraints were required. The biocompiler automatically completed the devices with RBS and terminators and found a viable arrangement for the parts. The sequences for the parts were looked up in the biobricks database, and are as following:

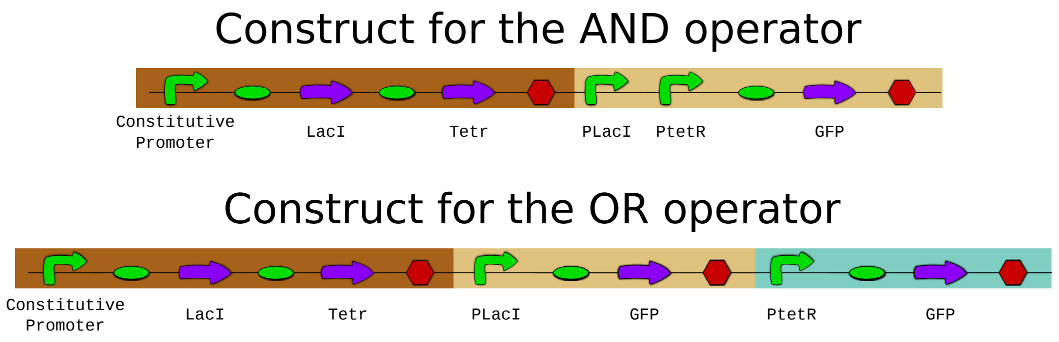


Figure 7: Constructs resulting from the biocompilation. Each operator is made of two or three devices. RBS and terminators have been added to complete the devices, and the sequences were found from the BioBricks database.

- Generic constitutive promoter: (biobricks entry: `BBa_J23100`)
- LacI: lacI repressor from E. coli (biobricks entry: `BBa_C0012`)
- Tetr: tetracycline repressor from transposon Tn10 (biobricks entry: `BBa_C0040`)
- PLacI: lacI regulated promoter (biobricks entry: `BBa_R0010`)
- PtetR: TetR repressible promoter (biobricks entry: `BBa_R0040`)
- GFP: green fluorescent protein derived from jellyfish Aequeora victoria wild-type (biobricks entry: `BBa_E0040`)

5 Conclusions

In this paper, we have presented how the INFOBIOTICS WORKBENCH software platform utilises the modelling and analysis of biological systems expressed in stochastic P systems using various computational techniques, e.g. simulation and verification. We have also presented our initial results on the automatic construction of genetic devices using a biomatter compilation module. We have illustrated our approach on two basic genetic devices: the AND and OR gates. The genetic parts used in the design of these gates are based on those used in [16]. We are planning to use other designs (based on different genetic parts), e.g. [17, 18].

We are currently working on a new version of IBW, which will incorporate a set of methods for specifying, modeling, testing and simulating biological systems, and will facilitate these processes for biologists. In addition, the new version will also include the biocompilation module that we are currently developing. The future version of the biocompiler will have an optimisation module, with an integrated RBS calculator [22] and codon optimisation [23], with the overall aim to be as close to the kinetic parameters decided upon in the simulation stage.

Acknowledgements. SKo and MG acknowledge the EPSRC (EP/I031812/1) support. CL and SKa are supported by EPSRC (EP/I03157X/1). HF, DS and NK's work is supported by EPSRC (EP/I031642/1). MG and FI are partially supported by CNCS-UEFISCDI (PN-II-ID-PCE-2011-3-0688).

References

[1] Păun, G.: Computing with membranes. Journal of Computer and System Sciences **61**(1) (2000) 108–143

[2] Frisco, P., Gheorghe, M., Pérez-Jiménez, M.J., eds.: Applications of Membrane Computing in Systems and Synthetic Biology. Springer (2014)

[3] Romero-Campero, F.J., Twycross, J., Cao, H., Blakes, J., Krasnogor, N.: A multiscale modeling framework based on P systems. In: Membrane Computing. Volume 5391 of LNCS. Springer (2009) 63–77

[4] Gillespie, D.: A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. Journal of Computational Physics **22**(4) (1976) 403–434

[5] Romero-Campero, F.J., Twycross, J., Camara, M., Bennett, M., Gheorghe, M., Krasnogor, N.: Modular assembly of cell systems biology models using P systems. International Journal of Foundations of Computer Science **20**(3) (2009) 427–442

[6] Blakes, J., Twycross, J., Konur, S., Romero-Campero, F.J., Krasnogor, N., Gheorghe, M.: Infobiotics Workbench: A P systems based tool for systems and synthetic biology. In: [2]. Springer (2014) 1–41

[7] Blakes, J., Twycross, J., Romero-Campero, F.J., Krasnogor, N.: The Infobiotics Workbench: An integrated in silico modelling platform for systems and synthetic biology. Bioinformatics **27**(123) (2011) 3323 – 3324

[8] Hinton, A., Kwiatkowska, M., Norman, G., Parker, D.: PRISM: A tool for automatic verification of probabilistic systems. In: Proc. TACAS. Volume 3920 of LNCS. Springer (2006) 441–444

[9] Donaldson, R., Gilbert, D.: A Monte Carlo model checker for probabilistic LTL with numerical constraints. Technical report, Bioinformatics Research Centre, University of Glasgow, Glasgow (2008)

[10] Baier, C., Haverkort, B., Hermanns, H., Katoen, J.: Model-checking algorithms for continuous-time Markov chains. IEEE Transactions on Software Engineering **29**(6) (2003) 524–541

[11] Kuchcinski, K.: Constraints-driven scheduling and resource assignment. ACM Trans. Des. Autom. Electron. Syst. **8**(3) (July 2003) 355–383

[12] Ladroue, C., Kalvala, S.: ATGC: Assistant To Genetic Compilation. In: Synthetic Biology, SB6.0, Biobricks Foundation (2013)

[13] iGem: Parts Registry. http://partsregistry.org/

[14] Biofab: International Open Facility Advancing Biotechnology. http://biofab.synberc.org

[15] Roberts, R.J., Vincze, T., Posfai, J., Macelis, D.: REBASE–a database for DNA restriction and modification: enzymes, genes and genomes. Nucleic acids research **38**(Database issue) (2010) D234–D236

[16] Beal, J., Phillips, A., Densmore, D., Cai, Y.: High-level programming languages for biomolecular systems. In: Design and Analysis of Biomolecular Circuits. Springer New York (2011) 225–252

[17] Tamsir, A., Tabor, J.J., Voigt, C.A.: Robust multicellular computing using genetically encoded NOR gates and chemical 'wires'. Nature **469**(7329) (2011) 212–215

[18] Regot, S., Macia, J., Conde, N., Furukawa, K., Kjellen, J., Peeters, T., Hohmann, S., de Nadal, E., Posas, F., Sole, R.: Distributed biological computation with multicellular engineered networks. Nature **469**(7329) (2011) 207–211

[19] Pedersen, M., Phillips, A.: Towards programming languages for genetic engineering of living cells. Journal of The Royal Society Interface **6**(Suppl 4) (2009) S437–S450

[20] Bilitchenko, L., Liu, A., Cheung, S., Weeding, E., Xia, B., Leguia, M., Anderson, J.C., Densmore, D.: Eugene - a domain specific language for specifying and constraining synthetic biological parts, devices, and systems. PLoS ONE **6**(4) (2011) e18882

[21] Beal, J., Lu, T., Weiss, R.: Automatic compilation from high-level biologically-oriented programming language to genetic regulatory networks. PLoS ONE **6**(8) (2011) e22490

[22] Salis, H.M. In: The Ribosome Binding Site Calculator. Volume 498. Elsevier (2011) 19–42

[23] Welch, M., Villalobos, A., Gustafsson, C., Minshull, J.: You're one in a googol: optimizing genes for protein expression. Journal of the Royal Society, Interface / the Royal Society **6 Suppl 4**(Suppl 4) (2009) S467–S476