

# Synthesis of Genetic Circuits from Graphical Specifications

Nam-Phuong D. Nguyen, Nathan Barker, Hiroyuki Kuwahara, Curtis Madsen, Chris J. Myers  
University of Utah, Salt Lake City, UT 84112, USA  
{namphuon, barkern, kuwahara, cmadsen, myers}@vlsigroup.ece.utah.edu

## Abstract

*EDA tools have facilitated the design of ever more complex integrated circuits each year. Synthetic biology would also benefit from the development of genetic design automation (GDA) tools. Existing GDA tools require biologists to design genetic circuits at the molecular level, roughly equivalent to designing electronic circuits at the layout level. Analysis of these circuits is also performed at this very low level. This paper presents a first step at developing a GDA tool that supports higher levels of abstraction. In particular, this paper describes a graphical specification language from which molecular descriptions can be synthesized. These descriptions can then be abstracted to facilitate efficient analysis.*

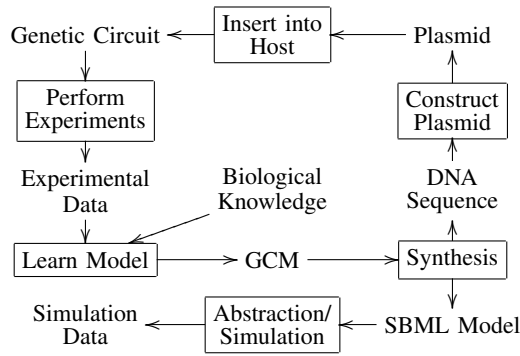
## I. Introduction

*Synthetic biology* is the study of genetically engineering new biological pathways that can potentially change the behavior of organisms in useful ways. An increasing number of laboratories are designing more ambitious and mission critical synthetic biology projects. Synthetic biology has the potential to help us better understand how microorganisms function by allowing us to examine how a synthetic pathway behaves *in vivo* as compared with simulation [21]. There are also numerous exciting potential applications. The Gates foundation is funding research on the design of pathways for production of antimalarial drugs [18]. Scientists have also been working on modifying bacteria to metabolize toxic chemicals [8], [9]. Finally, a number of labs are designing bacteria to hunt and kill tumors [4]. Since synthetic biology requires the construction of new, complex *genetic circuits*, all of these efforts would benefit from better *Genetic Design Automation* (GDA) tools.

*Genetic circuits* are biological circuits constructed from

DNA. MIT has created a registry of standard biological parts used to design synthetic genetic circuits [1]. A vital part of synthetic biology is the development of methods and tools to assist in the design and analysis of synthetic genetic circuits using these parts. One such tool is `BioJADE` which provides a schematic capture interface to the MIT parts registry [12]. To aid in the simulation of biological systems, a standard representation, the *Systems Biology Markup Language* (SBML) [3], has been proposed. Many simulation tools have been developed that accept models in the SBML format (see for example [2], [7], [10], [19]). SBML models biological systems at the molecular level. A typical SBML model is composed of a number of chemical *species* (i.e., proteins, genes, etc.) and reactions that transform these species. This is a very low level representation which is roughly equivalent to the layout level for electronic circuits. Designing and simulating genetic circuits at this level of detail is extremely tedious and time-consuming. Therefore, there is a need for higher-level abstractions for modeling, analysis, and design of genetic circuits.

To address this need, our group is developing the `BioSim` tool whose flow is depicted in Fig. 1. This tool can be utilized to either analyze an existing natural genetic circuit or assist in the design of a new synthetic genetic circuit. For an existing genetic circuit, time-series experimental data must first be collected using, for example, *microarrays* to gather gene expression data. Our tool can then analyze this data to learn the connectivity in the genetic circuit using the method described in [6]. The result is a *Genetic Circuit Model* (GCM) which is a graphical specification method that describes a genetic circuit at a level of abstraction above the molecular level. A GCM does not include every species and reaction explicitly rather it only includes the *influences* between the important species. We have developed a tool to transform a GCM into an SBML model which can then be analyzed using any SBML compliant simulation tool. In particular, `BioSim` incorporates an efficient temporal behavioral simulator that, whenever possible, abstracts out unnecessary details from the SBML model to substantially improve



**Fig. 1. The BioSim tool flow.**

the efficiency of simulation [14]. Such abstraction is of critical importance as genetic circuits typically involve very low molecule counts making stochastic simulation often essential [5]. To use this tool for design, one can begin with a GCM and modify it until it produces the desired simulation results. At this point, a DNA sequence can be generated that can be turned into a plasmid and inserted into a host cell.

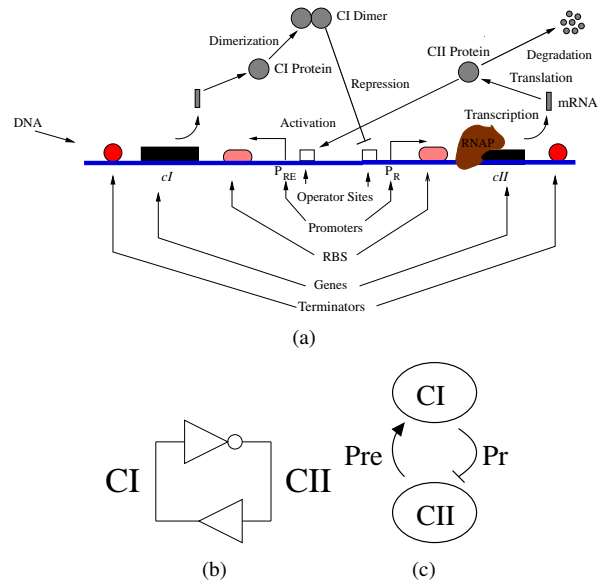
A critical element of this tool flow is the GCM which is the focus of this paper. This paper is organized as follows. Section II presents a brief overview of genetic circuits for those who are not familiar with them. Section III describes the GCM. Section IV describes the synthesis procedure from GCM to SBML. Section V gives a case study using this methodology, the design of a *genetic Muller C-element*. Finally, Section VI gives our conclusions.

## II. Genetic Circuit Basics

Fig. 2(a) depicts a simple genetic circuit found in the phage  $\lambda$  virus [17]. A genetic circuit is constructed from a strand of DNA. A coding sequence of DNA includes several key regions: *genes*, *promoters*, *operator sites*, *ribosome binding sites (RBS)*, and *terminators*. Genes are regions that code for a protein. In Fig. 2(a), the genes are *cl* and *cII* which code for proteins CI and CII, respectively. Promoters are regions on the DNA to which *RNA polymerase (RNAP)* binds to start transcribing the gene. In Fig. 2(a), the promoters are  $P_{RE}$  and  $P_R$ . Operator sites are regions to which proteins known as *transcription factors* bind to increase or decrease the affinity of the promoter. In Fig. 2(a), the transcription of *cl* is *activated* by CII binding to the operator site near the *cl* gene, and transcription of *cII* is *repressed* by the CI dimer binding to the operator site near the *cII* gene. RNAP binds to the promoter and starts *transcription*. Transcription is the process in which a strand of *messenger RNA (mRNA)* is created from the DNA. The RBS is the location on the

mRNA to which the ribosome binds to start *translation*. Translation is the process in which a protein is created from mRNA. Terminator sites signal the RNAP to stop transcription and unbind from the DNA strand. For a more technical review of genetic circuits, please see [13], [20].

The behavior of the genetic circuit shown in Fig. 2(a) is as follows. Initially, there are no molecules of CI or CII in the host cell. In this state, transcription of *cII* is very active while transcription of *cl* is not very active as its transcription must be facilitated by a CII molecule bound to its operator site. As CII molecules begin to build up in the cell, CII activates the transcription of *cl*. As CI molecules are produced, pairs of them bind together (i.e., *dimerize*) to form the CI dimer. A CI dimer can bind to the operator site for *cII* which represses the transcription of *cII*. Without further production of CII, the level of CII decreases due to degradation. This lowers the amount of CII available to activate CI production, so CI also begins to disappear. Eventually, only small amounts of CII and CI remain, returning the circuit to its initial state and the process repeats. This process is familiar to any digital designer as a simple oscillator as shown in Fig. 2(b).



**Fig. 2. (a) Simple genetic circuit. (b) Digital representation. (c) Graphical model.**

Genetic circuits can also be represented graphically. The nodes of the graph are the species. The edges are the influences between the species. Activation is denoted by an edge with a normal arrowhead while repression is denoted by an edge with a flat arrowhead. Fig. 2(c) is a graphical representation for the genetic circuit in Fig. 2(a). This type of representation is the motivation for the GCM representation described in the next section.

### III. Genetic Circuit Model

The GCM is a graphical specification language that was developed to aid designers in creating SBML models of genetic circuits by providing a method of describing them at a higher level of abstraction. In particular, in a GCM, only important species and their influence upon each other are represented which greatly reduces the model size.

A GCM is a tuple  $\langle S, P, G, I, S_d \rangle$  where:

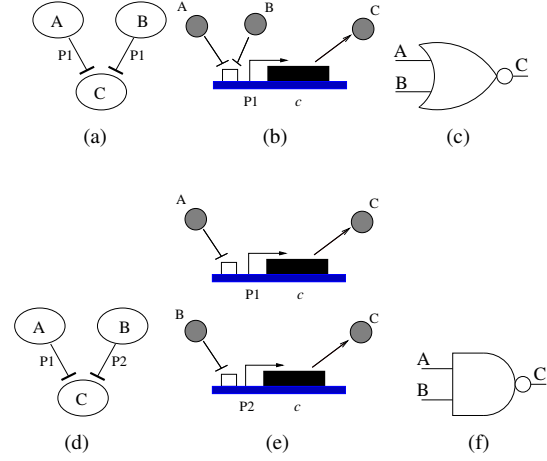
- $S$  is a finite set of species;
- $P$  is a finite set of promoters;
- $G : P \mapsto 2^S$  maps promoters to sets of species;
- $I \subseteq S \times P \times \{a, r\}$  is a finite set of influences;
- $S_d \subseteq S$  is a set of species that influence as dimers.

Species in  $S$  are the proteins in the genetic circuit, and the promoters in  $P$  are the locations on the DNA that initiate transcription of genes that produce these proteins. The set of species that are produced from a promoter is represented by the function  $G$ . Species can have influences on promoters. These influences can either be of type activation (denoted “a”) or repression (denoted “r”). If the type is activation, then as the amount of the species increases, the rate of production of the species associated with the influenced promoter also increases. If the type is repression, then as the amount of the species increases, the rate of production of the species associated with the promoter decreases. The species in  $S_d$  only influence other species after forming a dimer.

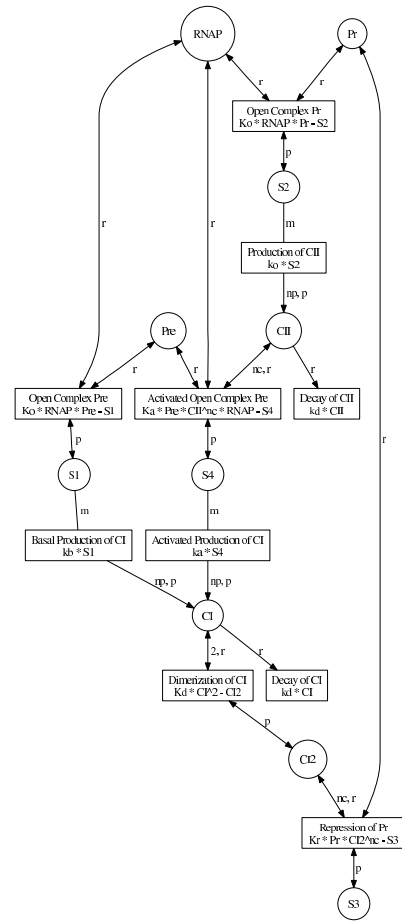
A graphical representation of a GCM is a bipartite graph with species and promoters as the two types of nodes. Species are connected to promoters using the influences in  $I$ , and promoters are connected to species using the function  $G$ . To simplify presentation, the graphs shown in this paper use only species as nodes, edges are inferred using  $I$  and  $G$ , and edges are labeled with the promoter that links the species. The graphical representation of the GCM for the simple genetic circuit is shown in Fig. 2(c).

Promoters group influences together that are associated with the same gene(s). As an example, consider the GCM shown in Fig. 3(a) which represents the genetic circuit shown in Fig. 3(b). With the same promoter associated with both influences, if either A or B is present, then C is repressed. In other words, this behaves like a NOR gate. Another example is shown in Fig. 3(d) which represents the genetic circuit shown in Fig. 3(e). With a different promoter for each influence, both A and B must be present to repress C. In this case, the behavior is like a NAND gate.

The GCM model for our simple genetic circuit includes just two nodes and two edges as shown in Fig. 2(c). To describe this same circuit using SBML requires ten nodes (i.e., species) and ten edges (i.e., reactions) as shown in Fig. 4. This SBML model can be automatically synthesized from the GCM representation as described next.



**Fig. 3. (a) GCM, (b) genetic circuit, and (c) logical behavior with same promoter. (d) GCM, (e) genetic circuit, and (f) logical behavior with different promoters.**



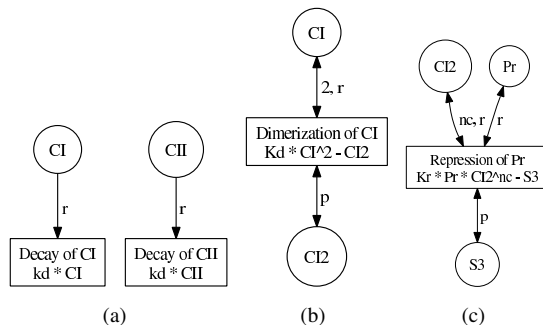
**Fig. 4. Species and reactions from the SBML model for the simple genetic circuit example.**

## IV. SBML Generation

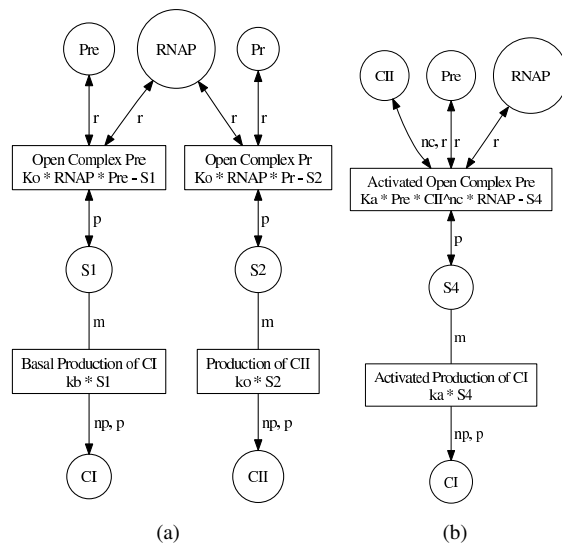
SBML has become a standard representation to model biological systems such as genetic circuits, and it is the input to many simulation tools. Therefore, it is extremely important to be able to synthesize an SBML model from a GCM. The key parts of an SBML model are the species and reactions. The ten species and ten reactions for the SBML model of the genetic circuit in Fig. 2 are shown in Fig. 4. In this figure, species are represented with circles and reactions are represented with boxes. Boxes contain a *kinetic law* which specifies the rate or probability of the reaction. An edge labeled by an “r” means that a species is a *reactant* for a reaction (i.e., it is consumed by the reaction). An edge labeled by a “p” means that a species is a *product* for a reaction (i.e., it is produced by the reaction). An edge labeled by an “m” means that a species is a *modifier* for a reaction (i.e., it is neither produced nor consumed by the reaction). Edges are also annotated with a number to represent the *stoichiometry* for this species in the reaction (i.e., how many molecules are consumed or produced by the reaction). When the value is 1, it is omitted from the figure. When an edge is bidirectional, this is a shorthand to indicate that the reaction is *reversible*. In other words, there are actually two reactions represented. One reaction converts the reactants to products while the second reaction converts the products back to reactants.

While BioSim and other tools allow a user to create SBML models, these models are extremely tedious and time-consuming to construct. Clearly, there is a need for the higher level of abstraction that GCM provides. However, an algorithm to synthesize an SBML model is necessary for analysis. The algorithm that synthesizes SBML from a GCM begins with the species  $S$ , and a degradation reaction is created for each species as shown in Fig. 5(a). Next, a species is created for RNAP and each promoter in  $P$ , a new complex species is created for each promoter combined with RNAP, and reactions are added to create and destroy this complex as shown in Fig. 6(a). Reactions are also added that use this complex to produce the proteins that are associated with this promoter. If this promoter can be activated, then a basal production rate is associated with this reaction. Otherwise, the normal open complex production rate is associated with this reaction. Next, for each species that influences in dimer form, a dimer species is created and reactions to create and destroy them are generated as shown in Fig. 5(b). All influences that use this species are updated to use the dimer species instead. Next, for each repression influence, a complex species is created that is composed of the promoter and repressor species, and reactions are added to create and destroy this complex. For example, CI represses CII production using the reaction shown in Fig. 5(c).

Finally, for each activation influence, a complex species is created that is composed of the promoter, the activator, and RNAP, reactions are added to create and destroy this species, and a reaction must also be added that uses this complex species to produce the proteins associated with this promoter. This reaction uses the activated production rate. In the example, CII activates CI production using the reactions shown in Fig. 6(b). Finally, the complete SBML model is shown in Fig. 4.



**Fig. 5. (a) Degradation of CI and CII. (b) Dimerization reaction for CI. (c) Reactions for CI dimer repression of CII.**



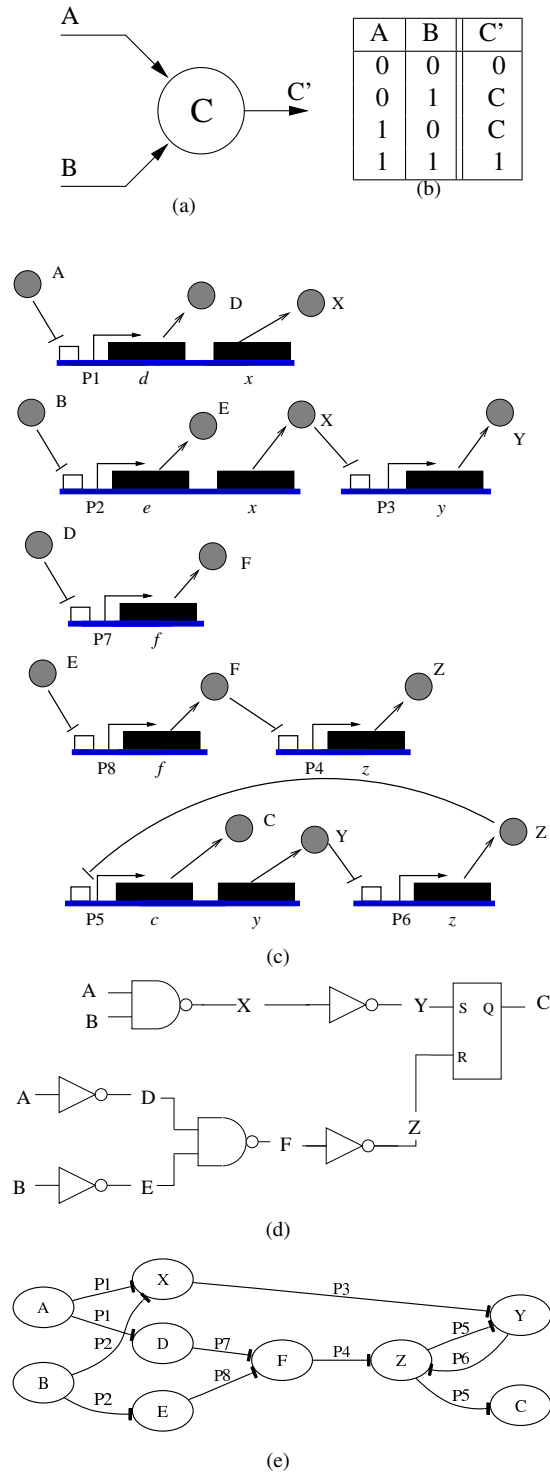
**Fig. 6. (a) CI and CII's open complex formation reactions. (b) Reactions for CII activation of production of CI.**

## V. Case Study: Genetic Muller C-element

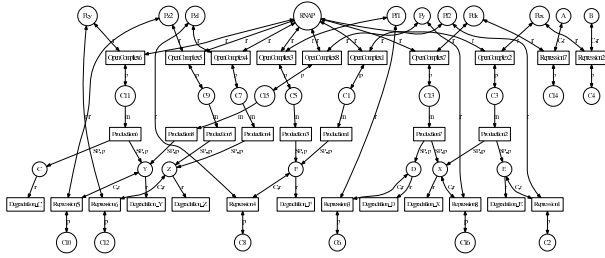
The Muller C-element is a state-holding gate that is often used in asynchronous circuits to synchronize signals without using any clock. Its schematic symbol is shown in Fig. 7(a), and a truth table describing its behavior is shown in Fig. 7(b). In particular, when both inputs  $A$  and  $B$  are low, the output  $C$  should be low. When both inputs are high, the output should be high. Finally, if the inputs are different, the output should retain its old state.

The design of a genetic Muller C-element makes it theoretically possible to build any asynchronous finite state machine in a genetic technology. In [16], potential designs are modeled and analysed at a low level using SBML. This section considers using GCM to model and analyze the genetic toggle Muller C-element from [16]. The genetic circuit schematic for this gate is shown in Fig. 7(c), and its logical representation is shown in Fig. 7(d). The C-element is constructed from two NAND gates, several inverters, and a SR flip-flop which is known as a *toggle switch* in synthetic biology literature [11]. This toggle is controlled by the intermediate species  $Y$  and  $Z$  that are produced from two different sources, one within the toggle to retain state and another outside to set the state.

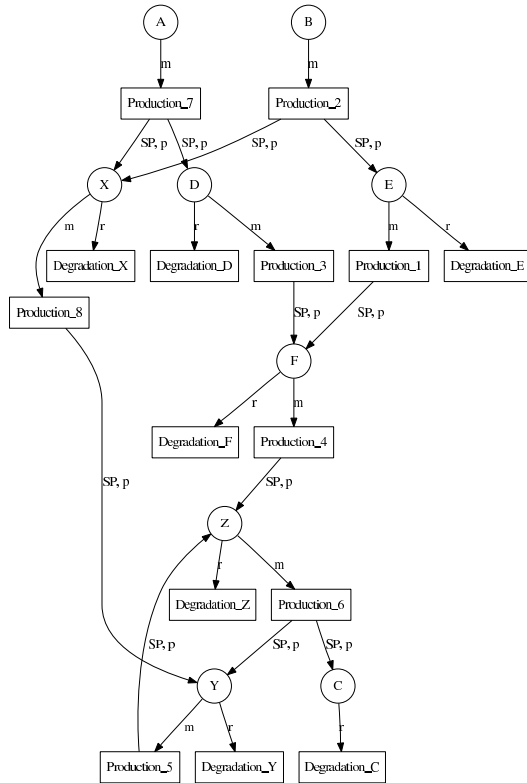
The GCM graph for the C-element is shown in 7(e). From the GCM, our synthesis tool produces the SBML model shown in Fig. 8(a) which includes 34 species and 31 reactions. There are several advantages to using GCM instead of SBML. First, it greatly increases the speed of model development and reduces the number of errors in the resulting model. Second, it allows efficient exploration of the affects of parameter variation as changing parameters is much easier due to the substantially smaller and more organized form of a GCM. Finally, since the algorithm controls the structure of the SBML output, the resulting model can be more easily abstracted using the methods in [14], [15] substantially improving simulation time. In this example, automatic abstraction reduces the model to only 9 species and 15 reactions as shown in Fig. 8(b). The average simulation result for 100 stochastic simulation runs for both the abstracted and unabstrated models are shown in Fig. 8(c). Initially, both  $A$  and  $B$  are low, and  $C$  remains low. At 2500 seconds,  $A$  goes high which does not cause a significant change in  $C$ . At 5000 seconds,  $B$  goes high causing  $C$  to rise. At 7500 seconds,  $A$  is removed, and  $C$  remains high. Finally, at 12500 seconds,  $B$  is removed and  $C$  rapidly falls. The simulation results show that this genetic C-element behaves as desired. Furthermore, the abstracted and unabstrated simulation results closely match. The unabstrated simulation result, however, requires 312 seconds of simulation time compared with only 20 seconds for the abstracted model. This is a speedup of more than an order of magnitude.



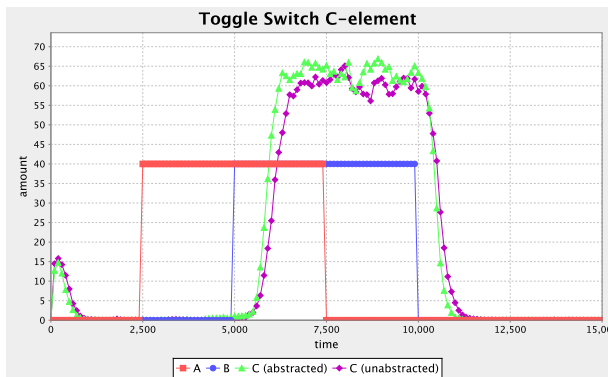
**Fig. 7. (a) Schematic symbol and (b) truth table for a Muller C-element. (c) Logical model, (d) genetic circuit, and (e) GCM graph for the toggle switch C-element.**



(a)



(b)



(c)

**Fig. 8. SBML for (a) unabstracted and (b) abstracted toggle switch genetic Muller C-element, and (c) simulation results.**

## VI. Conclusion

Current techniques of developing genetic circuits at the reaction level is too time consuming and prone to error. This paper presents the first steps in developing an HDL for genetic circuits. In particular, this paper presents the GCM format and a synthesis method from GCM to SBML models which have been incorporated into our GDA tool, BioSim. Our GCM format allows not only efficient model creation but it also facilitates abstraction leading to substantial improvements in simulation time. While this paper presents a good first step, there is still much work left to be done. The next important step is to add more levels of hierarchy into the GCM format. In particular, we plan to create structural constructs that allow us to connect GCM's for separate modules through species ports.

## References

- [1] Biological parts registry. <http://parts.mit.edu/>.
- [2] BioSPICE. <http://biospice.sourceforge.net/>.
- [3] Systems Biology Workbench. <http://www.sbw-sbml.org/>.
- [4] J. C. Anderson, E. J. Clarke, and A. P. Arkin. Environmentally controlled invasion of cancer cells by engineering bacteria. *J. Mol. Biol.*, 355:619–627, 2006.
- [5] A. Arkin, J. Ross, and H. McAdams. Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected escherichia coli cells. *Genetics*, 149:1633–1648, 1998.
- [6] N. Barker. *Learning Genetic Regulatory Network Connectivity from Time Series Data*. PhD thesis, U. of Utah, 2007.
- [7] Biopathwise. <http://bioanalyticsgroup.com/>.
- [8] G. M. Brazil *et al.* Construction of a rhizosphere pseudomonad with potential to degrade polychlorinated biphenyls and detection of *bph* gene expression in the rhizosphere. *Applied and Environmental Microbiology*, 61(5):1946–1952, 1995.
- [9] I. Cases and V. de Lorenzo. Genetically modified organisms for the environment: stories of success and failure and what we have learned from them. *International Microbiology*, 8:213–222, 2005.
- [10] Celldesigner. <http://celldesigner.org/>.
- [11] T. S. Gardner, C. R. Cantor, and J. J. Collins. Construction of a genetic toggle switch in *e. coli*. *Nature*, 403:339–342, 2000.
- [12] J. A. Goler. BioJADE: A design and simulation tool for synthetic biological systems. Technical report, Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 2004. AI Technical Report 2004-003.
- [13] J. Hasty, D. McMillen, and J. J. Collins. Engineered gene circuits. *Nature*, 420:224–230, November 2002.
- [14] H. Kuwahara. *Model Abstraction and Temporal Behavior Analysis of Genetic Regulatory Networks*. PhD thesis, U. of Utah, 2007.
- [15] H. Kuwahara, C. Myers, N. Barker, M. Samoilov, and A. Arkin. Automated abstraction methodology for genetic regulatory networks. *Trans. on Comput. Syst. Biol.* IV, pages 150–175, 2006.
- [16] N. Nguyen, H. Kuwahara, C. Myers, and J. Keener. Design of a genetic muller c-element. In *13th IEEE Int. Sym. on Async. Circuits and Systems*, pages 95–104. IEEE Computer Society, 2007.
- [17] M. Ptashne. *A Genetic Switch*. Cell Press, 1992.
- [18] D.-K. Ro *et al.* Production of the antimalarial drug precursor artemisinic acid in engineered yeast. *Nature*, 440, 2006.
- [19] Simbiology. <http://www.mathworks.com/>.
- [20] M. L. Simpson, C. D. Cox, G. D. Peterson, and G. S. Saylor. Engineering in the biological substrate: Information processing in genetic circuits. *92(5):848–863*, 2004.
- [21] D. Sprinzak and M. B. Elowitz. Reconstruction of genetic circuits. *Nature*, 438:443–448, 2005.