AUTOMATED GENERATION OF DYNAMIC MODELS FOR GENETIC REGULATORY NETWORKS

by

Pedro Fontanarrosa

A thesis submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Master of Science

in

Bioengineering

Department of Biomedical Engineering

The University of Utah

December 2019

The University of Utah Graduate School

STATEMENT OF DISSERTATION APPROVAL

The dissertation of **Pedro Fontanarrosa**
has been approved by the following supervisory committee members:

Chris J. Myers ,	Chair(s)	**6-14-19** Date Approved
Tara L. Deans ,	Member	**6-14-19** Date Approved
Orly Alter ,	Member	**6-14-19** Date Approved

by **David W. Grainger** , Chair/Dean of
the Department/College/School of **Biomedical Engineering**
and by **David B. Kieda** , Dean of The Graduate School.

ABSTRACT

Synthetic biology is an engineering discipline in which biological components are assembled to form devices with user-defined functions. As in any engineering discipline, modeling is a big part of the design process, since it helps to predict, control, and debug systems in an efficient manner. Systems biology has always been concerned with dynamic models, and a recent increase in high-throughput of experimental data has made it essential to develop dynamic models that can be used for an iterative learning process in a design/build/test workflow.

In this thesis work, an automated model generator is created to automatically generate dynamic models for genetic regulatory networks, implemented in the genetic design automation tool, iBioSim. This automated model generator uses parameters stored at an online parts repository and encodes the mathematical models it generates using Systems Biology Markup Language. The automated model generator is then used to model and simulate genetic circuits created with the design environment referred to as *Cello*. The simulation of the mathematical models produces a dynamical response prediction of each of the circuits, which is unavailable with steady-state modeling. Some of these dynamical responses present unexpected behavior. Using the dynamic models generated with the automatic model generator of this work, an analysis of the predicted behaviors yielded insight into the underlying biology phenomena that cause the observed glitching behavior of these circuits.

The last chapter of this thesis is focused mainly on future enhancements to the automated model generator of this work to produce more accurate and precise models not only for genetic regulatory networks in *Escherichia coli*, but any organism where parametrization exists as proposed in this thesis work. It also explores different analysis that could be implemented into the automated model generator of this work, in order to expand the assessment done on genetic circuits.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

NOTATION AND SYMBOLS

$mRNA_i$	the mRNA produced by the i-th genetic gate
P_i	the protein produced by the i-th mRNA
$k_{mRNA_{i_{dim}}}$	mRNA rate of degradation or dilution
$k_{TF_{i_{dim}}}$	TF rate of degradation or dilution
y_{max}	maximum promoter activity
y_{min}	minimum (or basal) promoter activity, otherwise known as leakage
κ	coefficient of transcription factor activity
n	Hill's cooperation index
TU	a transcriptional unit
TF	a transcriptional factor
RPU	Relative Promoter Units
GRN	genetic regulatory network

ACKNOWLEDGEMENTS

First and foremost, I would like to thank Professor Chris Myers for being of persistent help and offering continuous direction. I would also like to thank Professors Tara Deans and Orly Alter, for all their efforts, and words of encouragement, into completing this thesis work and continuing this exciting academic journey. To Tramy Nguyen, Jet Mante, and Zach Zundel, thank you for your continued support, both professionally and personally. I would also like to extend my gratitude to Hamid Doosthosseini, without his support and collaboration this work would not have been possible. And last, but not least, to Moudi and Derek for making me feel like I am home.

CHAPTER 1

INTRODUCTION

Synthetic biology has strived to imprint engineering principles into classical genetic engineering. Some of those principles, like standardization and part characterization, are well on their way of development. Others, like decoupling, in where a complicated problem is separated into simpler independent problems, are harder to instill in this area of research. As a nascent discipline, genetic circuit design is reserved only for experienced researchers with a deep knowledge of biology. To a large degree, this is due to the inherent complexity of biological systems. However, as the field advances, so does the development of software that allows for more researchers to participate in synthetic biology. A key aspect of any engineering discipline is the ability to model and simulate designs. Modeling and simulation give the capacity to predict the outcome of a system and detect problems or malfunctions during the design process. This debugging capability is especially valuable in synthetic biology since it can take a long time and be expensive to genetically engineer an organism with a synthetic genetic circuit. Empirical data that differs from predictions can also shed light on part interactions or other biological phenomena not previously studied [2]. As synthetic biology advances and genetic circuits become more complex, so does modeling. This complexity, in turn, makes the modeling process an even more difficult challenge, and it is imperative to develop tools for easy and automated modeling of genetic circuits. The goal to engineer DNA as others would engineer electric circuits is what pushes some synthetic biologists to develop tools to design, model, build, and test genetic circuits in a more automated and seamless manner. This work aims to contribute to this research agenda.

1.1 Synthetic Biology

Synthetic biology has attracted interest amongst a diverse group of researchers, from molecular biologists to computer scientists, to develop engineering methods for biology.

Though most of the development is done by synthetic biologists with substantial expertise, there has been a general effort to produce a more automated method to design genetic circuits for researchers without extensive knowledge on genetic design. Scientists have tried to implement foundational technologies that would make synthetic biology a genuine engineering discipline, where three of the most relevant methods to implement are *standardization*, *abstraction*, and *decoupling* [3]. Standardization is not only the use of data standard representations of genetic circuits and models for reproducibility of results, but also standards for the definition, description, and characterization of modular and reusable genetic parts [3, 4]. Abstraction is used for simplifying mathematical models, which reduces the effort of describing different genetic parts and reactions mathematically and, consequently, increases the number of components that can be used when modeling and increases simulation speed [5]. And finally, decoupling is the effort to separate a complicated problem, like engineering a synthetic organism with a specific function, into smaller, modular problems that can be worked independently. For synthetic biology, this would be decoupling designing and modeling constraints from the building technicalities of a synthetic genetic circuit [5, 6]. Standardization, abstraction, and decoupling are essential for model-based design of genetic circuits [6] and computer-aided design. Progress has been made with standardization, abstraction and decoupling [7, 8] in an effort to streamline the design/build/test pipeline (see **Figure 1.1**).

Applying engineering principles to biology is not something new, but many agree that a distinct biological engineering discipline, synthetic biology, started with the modeling, construction, and testing of two unique genetic circuits, a genetic toggle switch and a synthetic oscillatory network [9, 10]. There was something very particular about these two circuits: their function, rather than their output, was what interested the researchers. These circuits stemmed from mathematical models and concluded in design, which established a precedent where the design and construction of engineered *genetic regulatory networks* (GRNs) were facilitated by theory with predictive capacity. Nonetheless, there were some discrepancies between the theoretical model and the experimental results, as expected. Our understanding of how genetic circuits behave is reflected in the precision of our models, and thus any differences between predicted and observed results can be used as a tool to study further the dynamics of genetic networks [11–13].

This unique way of designing circuits has led to increased developments in modeling for GRNs (see Section 1.2) and of a library of orthogonal genetic parts with characterized behavior. This library is a key component for an automated procedure for the design of genetic circuits for synthetic biology, as well as its subsequent analysis, which would provide a more detailed build/test/design pipeline, as shown in Section 1.5. Such increased variety of models and library of orthogonal genetic parts has pushed model-based design of genetic circuits [14, 15], as well as tools for *computer aided design* (CAD) of genetic circuits, like Cello [16].

1.2 Modeling

Genetic circuits have reached such a degree of complexity that even experienced researchers have a difficult time considering all the interactions between parts of a system. Therefore, mathematical descriptions of genetic networks become a necessity and that is why genetic design is usually model-driven [17]. It is from these mathematical descriptions that one can model the system as a whole and obtain predictions of its behavior. The simulations and subsequent analyses can become instrumental not only to study the nature of genetics, but also to expose errors in design, parametrization, or the model itself.

Modeling of GRNs with appropriate use of parameters is expected to yield indispensable contributions and aid in complex genetic design, furthering the advancement of synthetic biology. It is common to find abstract mathematical models describing GRNs that use nonspecific (or generic) parameters obtained from the literature. The inaccuracy of many models stems from the use of these nonspecific parameters because even if the predictive accuracy of a model fits observation for a specific organism, using these parameters for other organisms/systems would produce inaccurate results [18], and thus it is essential to have a model generator that is accompanied by the correct set of parameters. Therefore, the model generator in the present research works with a specific set of gates and parameters from a repository used by Cello [16], a popular CAD tool in synthetic biology.

Biological systems are highly complex due to the uncountable number of interactions and interconnectivity, which makes the prediction of behavior almost impossible without the use of models and simulations. Even though mathematical models and correct

parametrization can help simplify the design process, experimentation and comparison of the results are imperative to correct the models, fine-tune the parameters, and unearth unknown interactions. Computer simulations using mathematical models can help scientists understand the biological mechanisms and unknown phenomena [2], as well as help to bridge the gap between predictions and experimental results, denoting previously missing experimental data.

As the synthetic biology community deepens its knowledge of genetic interactions, more sophisticated modeling tools can be created to predict their behavior with higher accuracy and fidelity. However, there is a drawback in these types of models if the number of equations and parameters used to describe the system becomes overwhelming. To reduce complexity, there are different assumptions and simplifications that can be made to abstract the model and simplify it without losing predictive capabilities or accuracy. Abstraction can help scientists produce more complex and novel genetic designs for various applications in the industry [5]. Some studies have done this not only by simplifying the equations but also by combining and redesigning genetic components into bigger, composite parts to reduce variability and increase accuracy [3, 16].

Many different approaches have been developed to model and simulate genetic regulatory systems [19–24] with different focuses and aims in mind and each having different advantages/disadvantages. It has been shown that kinetic modeling is an appropriate way to model genetic regulatory networks [19, 25]. Starting with a kinetic model, a common way to describe GRNs is by utilizing *ordinary differential equations* (ODEs), which represent the rate of change of species and concentrations of the system. For a system of ODEs, a steady-state model assumes that all the rate equations are in equilibrium, effectively removing time from the model and focusing on the stable states of the system. Instead, when using quasi-steady-state assumptions, a model can produce different time points, and, as such, effectively predict information on the concentration of molecular species over time. Dynamic modeling, as it is called, has a significant advantage for modeling the dynamics of the circuits and their transition states. Dynamic modeling provides a more detailed description of GRN dynamics that can be used to determine circuit failures, to optimize for speed of transitions or cell source-allocation, and to better understand how the different genetic components interact. Since dynamic modeling can predict transition states, it can

also be used to determine unwanted transition states between different stable-states (also called *hazards*) due to gate propagation delay in that circuit.

With this in mind, the primary objective of this work is create an automatic generator of a two-step kinetic model using equations similar to the Hill-equations [26, 27] following the parametrization used in the original Cello paper [16] to predict mRNA and protein production over time. This thesis offers a new standard of constructing genetic components, and parameterizing them, as well as providing a simple way for nonengineers to automatically generate mathematical models with predictive capabilities for both mRNA species concentrations as well as protein outputs.

1.3 Standards

A critical piece of the puzzle for model-based design is the use of globally accepted standards. Standards would not only allow for the sharing and contrasting of knowledge but also tackle one of the most significant obstacles facing any engineering discipline today: the problem of reproducibility [28, 29]. For synthetic biology, reproducibility not only requires the use of standards for modeling and simulation but also for experimental setups, DNA design, and parametrization. Furthermore, the use of data standards are necessary to allow for data exchange through different online data repositories and a higher sharing of knowledge amongst different research groups. For these reasons, this work uses data standards for the genetic designs, the mathematical models that describe their dynamics, and their simulation environments so that anyone can reproduce the results obtained here. Data standards used in this thesis work are the *Synthetic Biology Open Language* (SBOL) [30] for the representation of genetic designs and their function; the *Systems Biology Markup Language* (SBML) [31, 32] for the mathematical model representation of the different genetic circuits and their interactions; and the *Simulation Experiment Description Markup Language* (SED-ML) [28] for the simulation description of the mathematical models. Additionally, this thesis proposes a standard of representation for genetic parts and their parametrization by following the architecture of the circuits designed in Cello [16].

1.4 Contribution

Facilitated dynamic modeling of genetic circuits would be an instrumental technique for synthetic biologists, especially if it can be accompanied by a circuit design automation tool, such as Cello. This would not only help automation in synthetic biology but also provide a way to debug circuit designs before construction and compare predictions with experimental data once the synthesized circuit is implemented. Since sequencing technologies are becoming cheaper and faster, there is an increased availability of large-scale data sets such as *RNA-Seq* (sequencing that reveals the presence and quantity of RNA in a biological sample) and *Ribo-seq* (sequencing technology that uses mRNA sequencing to determine which mRNAs are being actively being translated) data. RNA-seq can help elucidate part interactions, resource allocation and interference of the circuit with the host organism. These can be used to debug malfunctioning circuits [13]. There are many model generators for genetic circuits [33–38], but none of them are suited to utilize Cello parametrization of genetic parts, and thus they would be unsuccessful in modeling the dynamics of their output designs. For this, our method implements an automatic dynamic model generator that uses Cello's parametrization to generate a two-step model for the transcription and translation processes of genetic circuits.

This automatic model generator will help researchers in their design, construction, and testing of genetic circuits and expand the design/build/test pipeline (see **Figure 1.1**) into a more enriched, automated workflow, as shown in **Figure 1.2**. This workflow begins with the parametrization of parts and circuits to form a repository of genetic parts. This repository can be encoded on a data standard (like SBOL), which can then be used as a library of components for a genetic design tool (such as Cello). These design tools would produce a genetic circuit to be modeled and simulated to predict functions, outcomes, or problems in the design. The modeling and simulation can be used as a screening step to determine whether the circuits behaves in a manner that meets the researcher's intention. After the selection of genetic designs, one could proceed to construct these gene circuits and transform a host with them. After successfully engineering a host organism with the synthesized genetic circuit, it is necessary to perform experimental studies (like RNA-Seq or Ribo-Seq) to confirm the correct functionality and expected behavior of the process and, in turn, work as a debugging step for the designed circuits.

All of this while utilizing standards for the representation of genetic circuits (i.e., SBOL), as well as modeling and simulation (i.e., SBML and SED-ML). With this proposed workflow, we expect to create a new kind of standardization in which the researcher uses and expands on a library of parts using a Cello-gate approach to construct and parameterize genetic elements. This work should not only greatly facilitate the design and construction of genetic circuits but also serve as a tool to spot failures in design as well as unexpected interactions with the host organism.

1.5 Thesis Overview

Chapter 2 goes further describing the necessary background to set the context of this work. Firstly, it introduces genetic circuits (Section 2.1), their uses, and some examples of their applications in synthetic biology. This chapter continues to explain how these genetic circuits are constructed from well defined and characterized genetic parts, and how synthetic biologists are exploring new parts and creating a library of orthogonal components so that others can use them in their designs (Section 2.5). These parts repositories are built to be shared amongst researchers, and thus this chapter also explains the importance of standards (Section 2.4), and why it is so imperative to develop globally accepted synthetic biology standards, as this fosters not only reproducibility but also sharing of knowledge between different labs for the greater advancement of science. This chapter also introduces the reader to the mathematical modeling of genetic circuits (Section 2.3), specifically the Hill-equation based models that have been used extensively for genetic circuit design (Section 2.6), and in particular for the Cello project (Section 2.6.1).

Chapter 3 presents our dynamic model generation procedure. This chapter goes into detail of how our procedure handles the different parameters and gene dynamics to construct a mathematical model and, ultimately, simulate it to get meaningful results.

Chapter 4 uses the model generator to remodel all of the circuits from the original Cello paper [16], and analyzes them in new light. Furthermore, an analysis of how to use the dynamic model generated to understand biological phenomena is described.

Finally, Chapter 5 presents a summary of our results, workflow, and our conclusions, and explains future proposed standards for dynamic modeling using our model generator. Lastly, this chapter presents future venues that this work opens up, and ways that our

model generator could be expanded to include more types of analysis.

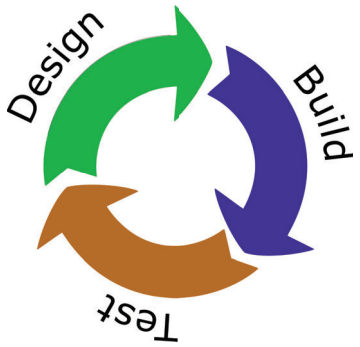


Figure 1.1. Design/Build/Test pipeline for synthetic biology. This depicts what is described in [39].

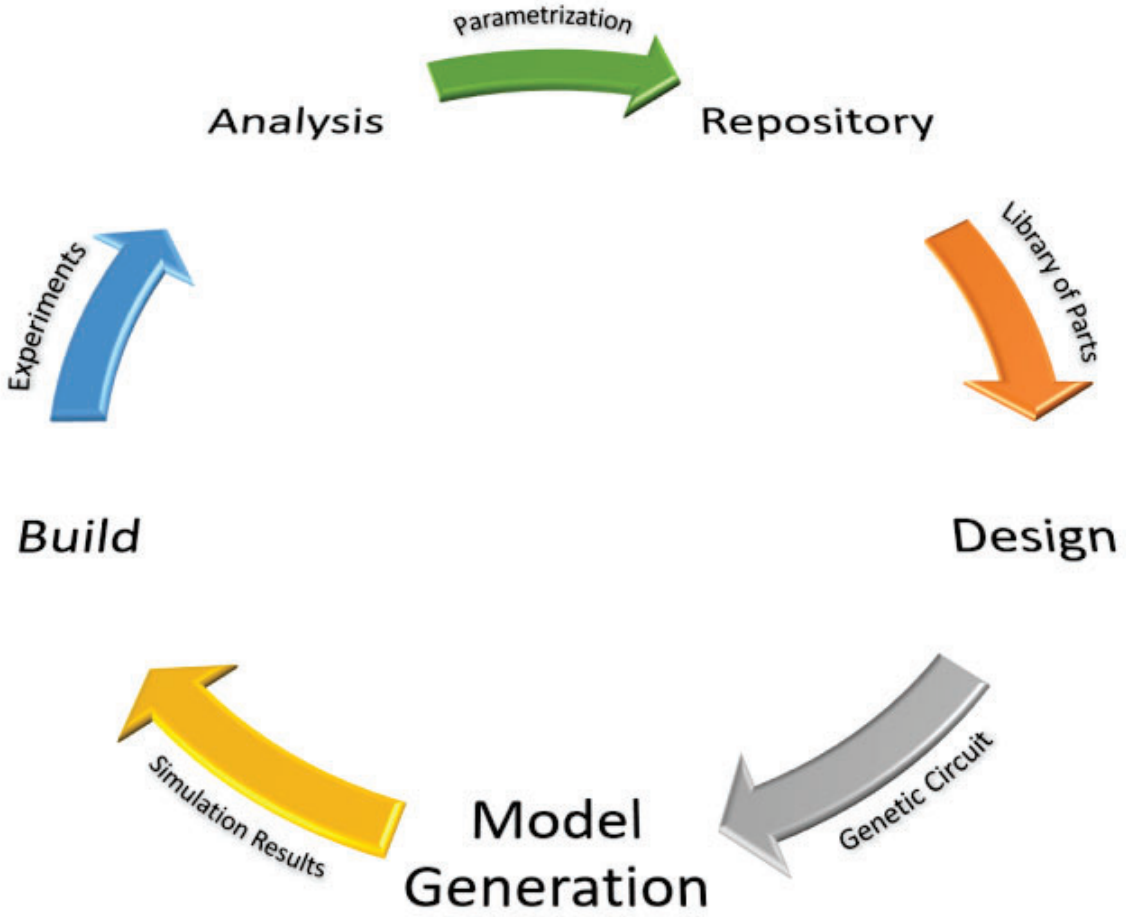


Figure 1.2. Design/Model/Build/Test/Learn workflow. An expanded version of the Design Build Test pipeline [39] and a proposed workflow for model-based design of genetic circuits.

CHAPTER 2

BACKGROUND

This chapter outlines the background research for the work presented in this thesis. The chapter describes what genetic circuits are (Section 2.1), and how synthetic biologists set out to design them using standardized genetic parts (Section 2.2). Once we know the object of study of Synthetic Biology, the chapter continues explaining the different ways to mathematically describe the relationships and interactions these genetic circuits undergo (Sections 2.3), and how these mathematical models and circuit specifications are encoded in data standard files shared by the Synthetic Biology community (Section 2.4). The chapter continues describing the online repositories (Section 2.5) where parts/genetic circuits/models are uploaded and shared across different research laboratories, and how this can be used to spur reproducibility in the synthetic biology community. Finally, this chapter describes different capabilities of software tools used to design/model/simulate GRNs (Section 2.6).

2.1 Genetic Circuits

Genetic circuits are designed GRNs, composed of synthetic genetic parts, that perform a specific function specified by a researcher. In recent years, there has been a plethora of new engineered genetic circuits with specific functions, an expansion of reusable modular genetic parts and sensors, and an exploration of synthesizing genetic circuits in organisms other than bacteria [12, 40–43]. Classically, genetic engineers would modify/add/knockout one, two, or a thousand genes of a living organism in order to study their role in the organism or to try to introduce a new feature or function to it. In the area of synthetic biology, characterized modular genetic parts or gates are combined to compose a genetic circuit with a user-defined function or implementation. It is this bottom-up design approach of genetic manipulation that distinguishes synthetic biology from classical genetic engineering.

Synthetic biologists draw inspiration from other engineering disciplines in order to have a more modular, predictable methodology for the design of genetic circuits. The first such genetic circuits, which were model-driven designed, are the genetic toggle switch [9] and a synthetic oscillatory network [10]. These genetic circuits were designed with a specific function in mind, and later built to test the design using genetic parts available. Nowadays, the design of genetic circuits enables researchers to engineer cells to process input signals, make logical decisions, implement memory, and to communicate with each other [44]. These circuits can also be designed to produce an output with a range of different purposes like inciting a biological response within or with other cells, producing a chemical for the environment or for industrial purposes, and many others.

Transcription and translation regulators that influence the flux of RNA and protein production are commonly used to carry out Boolean logic and therefore are called *logic gates*. These logic gates can be built on the basis of different regulator types using DNA-binding proteins, recombinases, CRISPRi regulation, RNA regulation, or protein-protein interactions [45]. Each gate can be designed to perform a specific logic function, like for example an AND, OR, NOT, NOR, NAND gates, and many others. With this, a researcher can link these logic gates to various cellular or environmental sensors and actuators, to generate circuits with precise desired behaviors in response to specific inter and intra-cellular signaling inputs [46]. However, to design these genetic circuits, there is a need for libraries of well-characterized, modular and standardized genetic parts and computational tools for easier design and to tune them [45], which is the topic of the next sections.

2.2 Genetic Parts or Gates

Genetic circuits represent how information is going to be processed and which decisions are going to be made, and these are composed of *genetic parts or gates* such as sensors, actuators, and logic gates [44]. These genetic parts are used to specify when, where, and how the different parts interact and under what conditions are genes expressed [44]. However, a difficult challenge in the area of genetic engineering is the unpredictability of the behavior of assembled genetic parts in different genetic contexts [45, 47]. There are efforts to design genetic parts or gates with predictable and modular functions [48–50] with predictable gene expression and extensive characterization like in [51–56], to cite a few.

Endy [3] suggests that the biological engineering community would benefit not only from building a library of characterized modular parts, like the "iGEM Registry of Standard Biological Parts,"[1] but also with the promulgation of standards that support the definition, description, and characterization of these biological parts.

Standardization of genetic parts is essential for reproducible experiments, reusability of genetic gates, and for model-driven design of circuits. Standardization can range from building techniques and gate conformation [4, 54, 57–59] to gate parametrization [52, 55]. A common method to characterize genetic gates in synthetic biology is using a standard for promoter activity, the *Relative Promoter Unit* (RPU) [60]. RPUs are used in many projects like iGEM and Cello (described in Section 2.6.1), and can be measured using a standardized kit for experiments, which makes it easy to adopt by different laboratories. This method is an effort to begin to address the challenge of characterizing promoters (and other types of standard biological parts) across the interdisciplinary community of synthetic biology [60].

The construction of genetic circuits requires a library of basic components with shared inputs/outputs, which permits the composition into more complex devices and circuits [61]. As well, characterized genetic parts and modularity are an integral concept in synthetic biology, and it is essential for model-driven design of genetic circuits [51, 62]. One such library of well characterized modular parts with shared inputs and outputs is the Cello genetic gate library [16], and therefore it is a good choice to develop models using them.

2.3 Mathematical Models of Genetic Regulatory Networks

A mathematical model can provide mechanistic understanding of a GRN. Models that accurately predict behavior of a system allow engineers to design genetic circuits *in silico* before going to the laboratory, avoiding large numbers of trial-and-error experiments [62]. There are many advantages in modeling: predicting, even in a limited manner, how a system will behave under novel conditions, understanding how highly nonlinear systems work, revealing deeply hidden properties of a system, understanding where does the design fail when predicted behavior is not what it is intended, and many other reasons [63].

---

[1] `http://parts.igem.org/`

However, models only have a limited capacity of prediction and newer models usually replace older approaches when their predictive capabilities increase.

Dynamical modeling can be described as the "classical" way to mathematically model GRNs [23]. The objective of these models is to describe the dynamic behavior of a set of genes with interconnected expression levels, and predicting the behavioral response to various environmental changes and stimuli.

Quantitative PCR, microarrays, Northern blotting, and other techniques are getting cheaper and can typically measure average concentration of mRNA in a population of cells [64]. Western blotting can do the same for proteins [65]. Therefore, a mathematical model that deals with concentration averages over time, and the proportional amount of time in which a promoter is being occupied is needed. A very common method to do so is to describe a GRN using the law of mass action, classical chemical kinetics, and Hill functions; all of which are explained in the next three sections.

2.3.1 Law of Mass Action

There is an associated *rate constant* for each chemical interaction (i.e., a parameter that is proportional to the frequency a reaction occurs). The *law of mass action* states that the rate of a chemical reaction is directly proportional to the product of the reactant concentrations, to the power of their stoichiometry. This means that the change in concentration of the reactants per unit of time (velocity of a reaction), or in other words, time derivative to reactant concentrations, is proportional to the product of these reactant concentrations. This quantity accounts for the probability of collisions amongst reactants under the assumption of a well-stirred system [25]. This can be used to convert a chemical reaction network into a set of *ordinary differential equations* (ODEs) that can be analyzed using *classical chemical kinetics* (CCK) model, which is the subject of the next section.

2.3.2 Kinetic-Based Models

Once all the reaction and species that comprise a GRN are identified, a mathematical model can be constructed by determining how they interact [25, 62]. Using the law of mass action, a set of ODEs can be derived that describe the change of species over time. This set of ODEs that track the concentrations of each chemical species is known as a *kinetic based model*, and the differential equations that compose it are known as *reaction rate*

equations. The model assumes that reactions occur continuously and deterministically [19]. This deterministic framework is appropriate to describe the mean behavior of biochemical systems [24]. While mass-action kinetics are strictly only valid for elementary reactions, they are widely and successfully applied in many fields of mathematical modeling in biology [66].

Reactions in biological systems are not only regulated by reactants and products, but also of other compounds that regulate the activity of these reactions like enzymes, often without being consumed during the reaction. In the next section, a description of a method that has proven to be appropriate to model enzymatic reactions [25] is explained.

2.3.3 Hill Equations

The Hill equation is a standard for characterization of regulated promoters because it demands only two parameters: the *Hill coefficient* (n) and the *Hill constant* (K_H or κ). These two parameters can be faithfully determined with experiments and represent an appropriate characterization of promoter/transcription-factor dynamics.

The Hill equation stems from the kinetic based model and assumes that the promoter of a transcription factor is momentarily occupied by transcription factors in a reversible reaction. Under the assumption that the concentration of the transcription factors and promoters is constant and under a steady-state condition, we can obtain two different forms of the Hill equation, depending if the transcription factor activates (2.1) or represses (2.2) the promoter:

$$P^* = \frac{(\frac{A}{\kappa})^n}{1 + (\frac{A}{\kappa})^n} \cdot P_T \ , \tag{2.1}$$

$$P^* = \frac{1}{1 + (\frac{R}{\kappa})^n} \cdot P_T \ . \tag{2.2}$$

In which P^* is the concentration of promoter bound with a transcription factor, A and R are the concentration of transcription factors that activate or repress transcription respectively, P_T is the total amount of promoters in the system, κ is the *Hill constant*, and n the *Hill coefficient*. In this model, κ gives the necessary concentration to activate or repress half of the promoters of the system, and n quantifies the *cooperativity* amongst activators (or repressors) when binding to a promoter [19, 25].

Kinetic-based models can take additional assumptions to further simplify the model without significantly affecting the ability to reproduce expected behavior [62]. One such common assumption is the *steady-state assumption*, which is described in the next section.

2.3.4 Steady-State Modeling of Genetic Regulatory Networks

When the production and degradation rate of a chemical species are equal, its concentration will not change and is at *equilibrium*. The steady-state assumption assumes that all the chemical reactions of a system are at equilibrium or *steady-state*, meaning that the concentration of the chemical species do not change over time. For a system to be at steady-state, each variable in that system must be at steady-state. Such steady-states are found by setting all the first derivatives in a the kinetic-based model equal to zero and solving the resulting set of algebraic equations [63]. Steady-State modeling has been shown to be appropriate for genetic regulatory network modeling [25].

When steady-state models fail to reproduce observed behaviors or predict dynamical behavior before reaching steady-state, some of the assumptions must be revisited [62] and allow for more relaxed assumptions to take place. The following section describes a different set of assumptions that allow for dynamical modeling.

2.3.5 Dynamic Modeling of Genetic Regulatory Networks

As the complexity of a GRN increases, so does the behavior it presents, and therefore there is a need for a more accurate modeling technique [67, 68]. Instead of assuming that *all* species reach equilibrium as in steady-state modeling, some models only assume that some species (those that are involved in the fastest reactions) reach equilibrium before others. This would remove equations from the ODE system, which describe the evolution of the variables at steady-state [69]. This assumption is usually applied to enzyme interactions [63, 70], since in many GRNs, the protein-protein dynamics are much faster than the transcription or translation process, meaning that the protein interactions reach equilibrium much faster than other interactions. However, depending on the system, this *quasi-steady-state assumption* can be made for any species in the system in which the modeler thinks there is a faster dynamics. This allows one to study the dynamics of species that are not at steady state with more precision. However, the quasi-steady-state assumption can only be safely made when the difference in time-scales for the dynamics

of species that reach equilibrium fast and those that do not are considerable [70].

2.4 Data Standards

Reproducibility is a critical issue for synthetic biology [28, 29, 71]. This rapidly advancing field has allowed for novel genetic circuit designs, modeling software, and assembly techniques. However, all of these developments are very labor-intensive with inputs from researchers with a multitude of different backgrounds, making the reusability of this information complicated. More mature engineering disciplines have tackled this issue with *standardization*, *abstraction*, and *decoupling* [3, 6, 50]. Some of these strategies, like abstraction and decoupling, are well under way of development. However, there is a growing awareness that the need for standardization is essential for the field to grow into a more predictable engineering discipline [50].

Standardization in synthetic biology ranges from standardized genetic parts and characterization [60], to standards for designing and visualizing genetic circuits, assembly methods, screening methods, reporting (modeling and simulation), and sharing (data repositories). Institutes like the National Institute of Standards and Technology (NIST) have dedicated efforts to define and develop standardization in synthetic biology, from DNA building blocks to documentation of experiments [72].

A key element for model-based design in synthetic biology is to develop *data representation formats* to allow for sharing of designs, models, and simulations in order to foster the interdisciplinary approach that is characteristic of this discipline [6]. A major initiative to encode biological information and to coordinate the development of data standards happens under the "COmputational Modeling in BIology NEtwork" (COMBINE [2]) initiative [6, 7, 71].

In this section, a description of the three data standards curated under COMBINE and used by the model generator of this work is described. First, the data standard for specification of genetic circuits and their function (Section 2.4.1), followed by the standard used to describe mathematically the biological behavior of these genetic circuits (Section 2.4.2), and finally the data standard to report simulation results of the mathematical models developed (Section 2.4.3).

[2] `http://co.mbine.org/`

2.4.1 Synthetic Biology Open Language (SBOL)

The *Synthetic Biology Open Language* (SBOL) is an open standard for the representation of *in silico* biological designs[3] [30]. SBOL is a free and community-driven data standard used to encode structure and function of a genetic circuit or parts, with a focus on abstraction and composition [71]. SBOL is used to represent not only the sequences of genetic designs, but also functional interactions, proteins, metabolites, and biological chassis. This is a big advantage over other standards that encode DNA sequences like the FASTA format [73], GenBank's flat file format[4], and the Generic Feature Format[5], since researchers can describe a biological design or circuit without knowing necessarily the DNA sequences that will compose it. In this way, synthetic biologists can share designs of genetic circuits, their expected function, and interactions without even having to go to the laboratory and sequence DNA. Furthermore, SBOL allows for hierarchical designs that organize genetic parts into a more complex structure to describe a desired function, annotate environmental or experimental context information, computational models of behavior, and measurements of performance characteristics [30]. To guarantee interoperability and sharing between tools, SBOL permits to assign one role and type to the functional components that compose the genetic design from ontologies, such as the *Sequence Ontology*[6] (SO) and the *Systems Biology Ontology*[7] (SBO) [74].

SBOL also allows one to create a *Model* class to document and link to external models written in standards other than SBOL, like for example the *Systems Biology Markup Language* (SBML) [31], which is discussed in the next subsection.

Most importantly, SBOL allows for the storage of any information not supported by the format in the form of custom and complex user annotations, so that no information is lost when encoding a design in the SBOL format.

---

[3] `http://sbolstandard.org/`

[4] `http://www.insdc.org/documents/feature-table`

[5] `https://github.com/The-Sequence-Ontology/Specifications/blob/master/gff3.md`

[6] `http://www.sequenceontology.org/`

[7] `http://www.ebi.ac.uk/sbo/main/`

2.4.2 Systems Biology Markup Language (SBML)

The *Systems Biology Markup Language* (SBML) is another standard under the COMBINE initiative [32]. It is a free and open format for computer models of biological processes. SBML is useful for models of metabolism, cell signaling, genetic devices, and more. It is supported by an international community with many packages being developed from the users that expand its capabilities. It is supported and used by more than two hundred software tools,[8] which makes it an excellent format for the exchange and reuse of mathematical models between different areas of synthetic biology. This enables researchers to create, simulate, and annotate biological models that can be shared through different databases, like the BioModels database [75]. Furthermore, some parser libraries offer model checking, validation, and verification, as well as support for SBO [76].

SBOL can be used to describe structural and qualitative behavior, whereas SBML is used to specify a mathematical model that describes the quantitative behavior of the system. Both are amply used standards, so naturally there exists converters from SBML to SBOL and vice-versa [77, 78]. This work is essentially a new type of SBOL to SBML converter that creates dynamic models from genetic designs encoded in SBOL using the parametrization from the Cello project. While SBML is a widely accepted and used format for describing model structure, it does not cover the description of analysis or simulation performed to obtain predictions from the mathematical model. Therefore, this work uses another standard format to do so, which is described in the next subsection.

2.4.3 Simulation Experiment Description Markup Language (SED-ML)

The *Simulation Experiment Description Markup Language* (SED-ML) is an XML-based format developed for the encoding of simulation and analysis experiments performed on a mathematical model [79]. It is a core standard of COMBINE, and it is used primarily to specify which models to use in an experiment, modifications to apply on the models before using them, which simulation procedures to run on each model, what analysis results to output, and how the results should be presented [79].

To enable shareable and reproducible analysis, authors should provide SED-ML files

[8] `http://sbml.org/SBML_Software_Guide`

along with the SBML files with their publications, so other researches can reproduce the presented simulation results obtained. This would allow other users to analyze and study under which conditions the simulation was carried out and test the results themselves.

2.5 Online Repositories

Data standards allow for standardization of ideas and information from a diversity of sources in synthetic biology. However, designs, models, and simulation results specified in these standards need to be stored somewhere in order to enable this data to be exchanged between different laboratories or researchers. *Online repositories* have been created for this purpose of storing and sharing data and are used by many researchers and tools. Some of them, like the iGEM Registry of Standard Parts[9] or JBEI-ICE [80], have been developed specifically for the storing and sharing of engineered biological deigns. However, many of these online repositories can only store sequences but not other information such as proteins, interactions, metabolites, biological chassis, or models that describe the function of a circuit, which is very important for genetic design. With this in mind, two online repositories have been developed to fulfill this gap which are discussed in the following sections.

2.5.1 SynBioHub

SynBioHub [81] is a repository for genetic designs encoded in SBOL. SynBioHub is designed to store parts and designs in a linked format so that data can be findable, accessible, interoperable, and reusable (FAIR) [81]. This linked data and the use of sequence ontologies permits for a powerful querying capacities that facilitates searchability from tools and users alike. Designs can be shared with users or other applications, like, for example, Benchling.[10] Sequences can be shared with other applications that specialize in sequence editing and/or copying, and then transferred back into SynBioHub. Moreover, any custom annotations made by a researcher in the SBOL file are queryable, which makes the retrieval of information more straightforward.

These functionalities make SynBioHub an excellent choice to upload genetic designs,

[9] `http://parts.igem.org/Main_Page`

[10] `https://www.benchling.com/`

models, and simulations, to search for existing designs, and to share or export for publication or collaboration.

2.5.2 BioModels

The *BioModels Database* is a public online resource that allows storing and sharing of published, peer-reviewed quantitative, dynamic models of biological processes [75]. Models uploaded to the BioModels Database are manually curated to ensure reliability and correspondence with the original publication's results. All models are annotated with controlled vocabulary terms and linked to external data, which facilitates model reuse and interoperability. Models are stored in the SBML format and are available to download in several other formats.

The submission of models to the BioModels repository has increased rapidly [75]. This allows modelers not only to share their models, but also to reuse models uploaded by other researchers to modify them and implement their own analysis and publish articles. This is a great resource to address the reproducibility crisis in synthetic biology [28, 29].

2.6 Genetic Circuit Design and Modeling

Despite all this potential, genetic circuit design remains one of the most challenging aspects of genetic engineering [45, 82]. Due to the inherent complexity of biological systems, engineering complex genetic circuits is a bigger challenge than was anticipated [83, 84]. As the requirements of developing novel synthetic biological systems have become more complex, the need for models and software design tools has become more acute [24]. Several approaches have been implemented for the development of computational tools for synthetic biology [24, 61, 83, 85, 86]. However, there has been an increased focus on tackling this complexity of genetic circuit design and frame these recent computational tools by developing *genetic design automation* (GDA) tools [86].

GDA tools rely on well characterized, modular genetic parts, in particular the development of orthogonal transcription factors [86]. This presents a challenge for the synthetic biology community since many genetic parts or gates have unbalanced regulator expressions, they behave differently when combined in a genetic system, and they have complicated states depending on the inputs [16]. However, one such project that has

overcome some of these difficulties and developed a library of characterized modular parts to use to automatically design genetic circuits is described in the following subsection.

2.6.1 Cello

The design environment, referred to as *Cello* [16], is a GDA tool created to automatically design genetic circuits with user-defined behavioral response over a set of inputs changes. It was developed to accelerate circuit design, to enable nonexperts to incorporate synthetic genetic circuits into their genetic engineering projects, and to enable one to specify a user-defined computational operation behavior of such a circuit. This design environment implements algorithms that derive a physical design (sequence of parts) from a textual specification in which the user specifies inputs, outputs, and an expected computational Boolean logic in the form of a truth table that the user wants the circuit to perform.

Cello needs three inputs in order to work. First, there is the DNA sequences of the sensor gates for the circuit, and their ON/OFF RPU output. The second is a *user constraint file* (UCF) that contains information such as the functional information (transfer functions in RPU) of the library of gates, the layout of the genetic system, organism, strain, operating conditions, toxicities, promoter road-blocking, and other constraints to be taken into account by the algorithm. And lastly, there is a Verilog code that captures the desired behavior (as a Boolean computational operation) of the genetic circuit to be designed [16].

Cello utilizes this information to automatically design a genetic circuit that connects to cell-based sensors and cellular actuators. It does so in three steps - first, the textual command is converted to a circuit diagram; second, Cello assigns specific regulators to each gate or node in the circuit diagram; the third and final step creates a linear DNA sequence based on the circuit diagram and gate assignment [16]. The output circuit is described using SBOL and it contains the DNA sequences of all the parts of the circuit. The actuator of such circuit can then be connected to any cellular process by directing the output of the circuit as a stimulator or repressor of a metabolic pathway or other genes. Similarly, the sensor gates can be engineered to sense different cellular inputs or experimentally controlled variables such as temperature, pH, etc. The circuit performs Boolean logic computation based on the presence/absence of the sensors and produces the corresponding output from the implemented behavior.

The work that Nielsen *et al.* did in 2016 [16] used a library of gates based on prokaryotic repressors. Nonetheless, the Cello design environment can work with any gate that is repressible in different levels other than RNAP flux regulation, such as RNA-based regulation, protein-protein interactions, CRISPR/Cas-based regulations, or recombinases, as well as in different organisms other than bacteria.

This GDA tool requires genetic logic gates that are sufficiently modular and reliable, such that their interconnected behavior can be predicted, in order to work. For this, the Cello project has developed a set of insulated NOT and NOR gates based on prokaryotic repressors [16, 87]. The following subsection describes the gates used and their parametrization.

2.6.1.1 Cello Gates and Parameters

As mentioned before, each gate in Cello behaves as a NOR or NOT gate, which is composed of an *engineered region* or *expression cassette* preceded by two (NOR gate) repressible promoters or one (NOT gate) repressible promoter, as shown in **Figure 2.1**. When the simulation environment selects different gates for each node in the circuit, it chooses from a library of these engineered regions or expression cassettes, instead of choosing the *Ribosome Binding site* (RBS), *Coding Sequence* (CDS), and terminators individually. Composing RBS + CDS + terminator into a functional component this way reduces variability, but at the same time, it is simpler to model and to combine during the simulated annealing process of Cello. Additionally, composed engineered regions or expression cassettes are easier to characterize experimentally, requiring far less experiments. It is a form of abstraction that reduces complexity and saves time [3].

Characterization of this library of composed parts was obtained experimentally [16, 88] as depicted in **Figure 2.2** and **Figure 2.3**.

Figure 2.2 shows how the **sensor promoters** are parameterized. First, a constitutive promoter is added before a sensor gate, which produces a sensor protein continuously. This sensor protein can repress the sensor promoter that is being characterized, unless an experimenter adds an input molecule that represses this repressor. In the same plasmid, the sensor promoter is placed before a "YFP RPU cassette," which is a functional component that produces *YFP* (Yellow Fluorescent Protein). YFP production is measured, using

RPUs, under two different conditions: adding an excessive amount of input molecule, in which case the sensor promoter is not be repressed and the YFP production is maximum; and without any input molecule, in which the sensor promoter is maximally repressed and only basal production of YFP occurs. With these experiments two parameters are obtained for sensor promoters: y_{max} and y_{min}. The parameter y_{max} depicts the maximum promoter activity for the sensor promoter, and y_{min} the minimum, or basal promoter activity in RPU. Likewise, **Figure 2.3** depicts the parametrization for all other gates that are not sensor gates or promoters. It is similar in fashion to the sensor promoter parametrization, but there is an extra step. In this case, the "YFP RPU cassette" is preceded by the gate promoter. On the same plasmid, the gate that produces the TF that represses this gate promoter is preceded by a sensor promoter. Finally, on a second plasmid, a sensor gate constitutively produces a sensor protein. In this way, without a input molecule that represses the sensor protein, the gate production is minimum, and the gate promoter is not being repressed (producing YFP). Conversely, when the input molecule is present in large amounts, the sensor protein is repressed, the gate being characterized produces maximum amounts of TF and the YFP production is reduced to a minimum. To characterize these gates, an experimenter introduces different concentrations of input molecule, and the YFP production is measured in RPU. A response function for each gate is formulated, and after fitting it to a Hill equation, the parameters y_{max}, y_{min}, n, and κ are obtained. As with the sensor promoter characterization, y_{max} and y_{min} depict the maximum and minimum promoter activity in RPU, respectively. For the other two parameters, n is equivalent to the Hill coefficient, and κ is equivalent to the dissociation constant [88].

The parameters y_{max}, y_{min}, n, and κ were measured for each gate in isolation of other gates, as shown in **Figure 2.3**, and are stored in a UCF, which is then fed to Cello when designing a circuit. Once Cello designs the circuit and assigns gates to each individual node of the circuit, it stores all that information in a SBOL file. The next section describes this output.

2.6.1.2 SBOL Specification

Cello was used to design a large set of circuits (52) based on the insulated gates described earlier [16, 88]. The output of Cello can be encoded in an SBOL file, as well as a

netlist (JSON file), cytometry plot (PNG file), transcription values in RPU (CSV file), truth table (CSN file), or others. Each circuit is composed of multiple NOR and NOT gates, sensor gates, and the output gate. A NOR gate is composed of two repressible promoters and an expression cassette; an example is shown in **Figure 2.4**.

A collection of the Cello insulated gates (expression cassettes and promoters) encoded in SBOL used for this work is uploaded in a SynBioHub repository.[11] With this, one can design a circuit using other design environments other than Cello and use these parts. These parts not only contain parts and sequence information, but they also store the parameters y_{max}, y_{min}, n, and κ. These parameters are stored as SBOL annotations in each expression cassette for the case of Cello gates, and in the repression interaction of sensor proteins to sensor promoters for the case of sensor promoters. The automated model generator of this work searches for these parameters to generate the model that describes the dynamic behavior of a circuit.

2.6.1.3 Cello's Circuit Performance Prediction

Qualitative predictions of circuit performance (output distributions) are obtained computing the combination for each individual gate's output distribution [88]. This is the last step performed by Cello after gate assignment and produces a prediction of the circuit's output as a distribution. To perform this prediction, there has to be experimental data to fit a response function for each gate. Thus, each gate in the circuit must have experimental cytometry distributions added to the UCF, with the fluorescence values reported in RPU [88].

Once all the gate distribution response functions are calculated, the qualitative predictions for the outcome product can be computed. For a particular input combination, the sensor values (concentrations) are fed to the first layer of the circuit (sensor gates). Each sensor gate has a distribution response function, so with the concentration of input molecule a vertical "slice" is obtained from the distribution response function to create an output histogram for the gate. Next, these gate output histograms become the input histograms for the second layer of gates. This is done for all the different layers, composing output histograms for each layer and feeding it as an input for the next layer, until the final

[11] `https://synbiohub.programmingbiology.org/public/Eco1C1G1T1/Eco1C1G1T1_collection/1`

circuit's output histogram is calculated. Then, finally, for each input signal combination, a histogram of output for each individual gate is estimated, and the signal is propagated throughout the entire circuit until the last one (circuit's output) is calculated to produce the truth table predictions of the work [16, 88].

The composition of response functions to obtain a predicted output histogram is based on steady-state experimental results and is a steady-state outcome prediction. This means that any dynamic behavior the circuit undergoes before reaching steady-state is missed by this analysis. However, in the original science paper work [16], researchers did a time-course experiment to obtain output production in RPU every hour for a particular circuit, until the circuit reached steady-state, shown in **Figure 2.5**. In this experiment they observed that the circuit, for some of the input combinations, behaved in an unpredicted manner: the output would vary in unexpected ways before reaching the correct predicted steady-state output of the circuit. This type of behavior cannot be predicted with steady-state modeling and simulation, and it is why it is so important to have a dynamic model that can do so. The ability to dynamically model genetic circuits using Cello gates and parametrization to be able to predict this dynamic behavior before and after reaching steady-states is what inspired the work of this thesis. Dynamic modeling would not only predict this kind of behavior, but also allow for a finer analysis of circuit dynamics to detect failures.

The Cello simulation environment is used mainly for the design and implementation of genetic circuits. However, there are other GDA tools that not only allow for the design, but also the modeling and simulation of genetic circuits. One such tool, iBioSim, is discussed in the next section.

2.6.2 iBioSim

iBioSim is a *genetic design automation* (GDA) tool for the design, modeling, and analysis of genetic circuits that is being actively developed at the University of Utah [89–91]. This tool has been developed to promote model-based design of genetic circuits using community-developed data standards such as SBOL, SBML, and SED-ML. While Cello is a GDA tool to automatically design genetic circuits, iBioSim is not restricted to genetic logic circuits. iBioSim allows for a wider range of genetic parts and metabolic species, modeling,

and simulation using data standards, automatic uploading to online repositories among other things. The following is a high-level description of the key features of iBioSim:

- **Genetic Circuit Design**

    1. Incorporated sequence editor tool SBOLDesigner [92]. Genetic designs can be viewed, edited, and create hierarchical levels of design.
    2. Front-end connection to SynBioHub for the uploading, downloading, and sharing of genetic circuits.

- **Model Generation**

    1. The *Virtual Parts Repository* (VPR) model generator is used to obtain and enrich SBOL files with interaction data, small molecules, and more for designed circuits.
    2. Integrated SBOL to SBML converter [78] that can be used to translate structural and functional information to create a quantitative model expressed in SBML using generic or user-defined parameters.
    3. User interface to edit and refine the model using the model editor GUI.

- **Analysis**

    1. Variety of simulation methods to analyze SBML models such as *ordinary differential equations* (ODEs) and stochastic simulation, and many others using SED-ML.
    2. Perform *flux balance analysis* (FBA) on SBML models.
    3. Perform stochastic model checking, and simulation of grid-based, hierarchical models of dynamic cellular populations.
    4. View simulation results plotted in a graph.

- **Synthesis**

    1. Automated methods for part selection using a process known as technology mapping [93].

2. Technology mapping for asynchronous sequential genetic circuits [94].

iBioSim provides automatic SBOL to SBML converter, though not one that can use Cello's parts and parametrization to generate a dynamic model. This thesis work implements an automatic dynamic model generator for genetic parts that uses Cello parametrization in iBioSim. In the next subsections, a more detailed description of the VPR, the SBOL to SBML converter, and dynamic modeler of iBioSim is provided.

2.6.2.1 VPR

Modular genetic parts for synthetic biology not only can be reused for different projects, but also provides modular and reusable models and information to be shared. Modular models facilitates the process of model-centered design and the availability of databases of modular models is essential to support automated model generation tools like iBioSim. The *Virtual Parts Repository* (VPR) has been developed with this emphasis on mind. VPR is a repository of *standard virtual parts* (SVPs), which are reusable, modular, composable, and shareable models of physical biological parts for synthetic biology [95]. The computational models and interactions are available as SBML documents and in SBOL format for standardization purposes. The repository was populated with data mined from an ontology representation of the BacillOndex dataset [96, 97], which includes around 3000 virtual parts and 700 models of interactions between them.

An *application programming interface* (API) is also available to enable programs to access VPR via a Web service. This can be used to retrieve SVPs, a list of interactions for a part or SBML models of parts and interactions to construct models of biological systems [95].

These features are used by iBioSim to obtain interaction data, and add functional information to the SBOL description of a genetic design. It can add proteins as well as coding sequences in the same SBOL document in which the design is specified [89]. The use of VPR for automated processes like the automated generation of models for GRNs is particularly suitable for these reasons [98].

2.6.2.2 SBOL/SBML Converter

iBioSim also comes with an integrated SBOL to SBML converter [78]. This utility is used to convert qualitative models and structure encoded in SBOL to quantitative models

expressed in SBML [89, 98]. During the construction, the species and reactions generated for the mathematical model encoded in SBML are also annotated with elements from the SBOL document in order to preserve provenance of the model and the molecular identities of the species [78]. This converter adds default parameter values to the interactions [98], but these parameter values can be later modified within the iBioSim model editor. The derivation of these rate laws is based on the law of mass action and some model abstraction techniques like the operator site reduction or quasi-steady-state approximation [78]. For a more detailed review of these abstractions can be found in the literature [19]. SBML models constructed this way can then be simulated in a variety of methods.

The work presented in this thesis is a new SBOL to SBML converter that uses functional and structural information of a genetic circuit encoded in SBOL to produce a mathematical model described in SBML, using parameters and characterization as in the Cello project to create a dynamic model of GRNs.

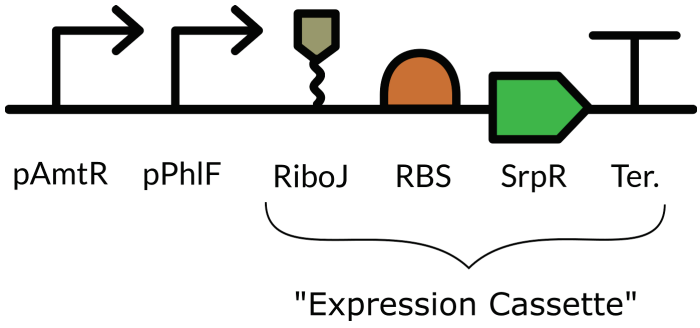


Figure 2.1. A Cello NOR gate [16]. Each gate in Cello consists of a genetic "expression cassette" (in this case the gate is "S4_SrpR") or engineered region that interacts with a downstream promoter. In this example, it is preceded by two repressible promoters, ("pAmtR" and "pPhlF"), which cause the gate to behave as a NOR gate. In this figure: **pAmtR** (promoter repressed by AmtR), **pPhlF** (promoter repressed by PhlF), **RiboJ** (insulator), **RBS** (Ribosome Binding Site), **SrpR** (SrpR coding sequence), and **Ter.** (terminator).

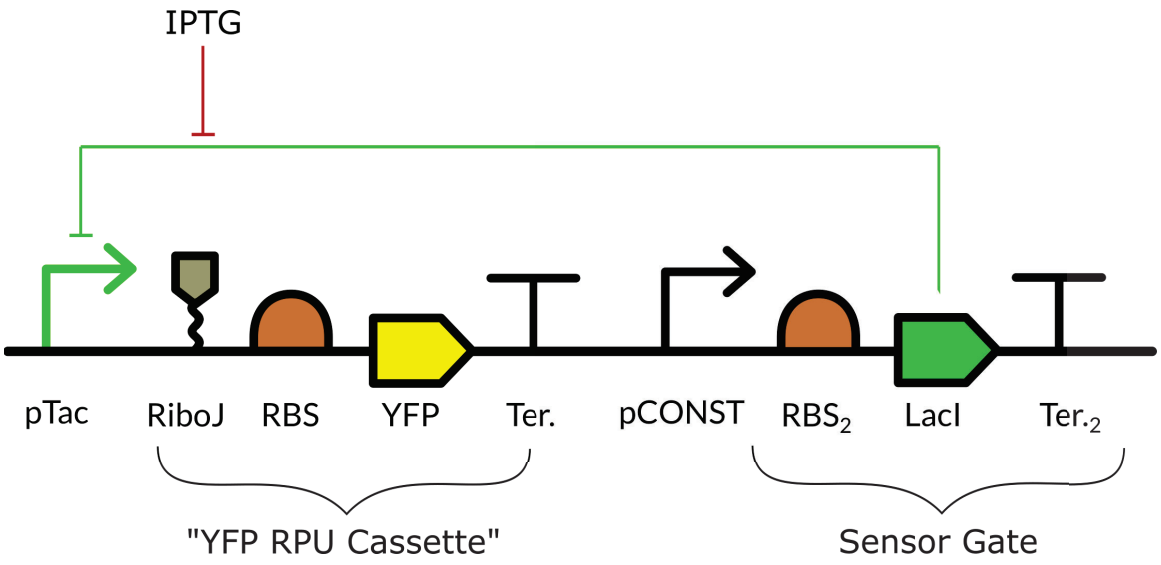


Figure 2.2. Sensor gate parametrization in Cello [16]. Each gate in Cello consists of a genetic "expression cassette" (in this case, a sensor gate) or engineered region that interacts with a promoter (in this case "pTac"). To characterize the RPU activity of a sensor promoter, the sensor promoter ("pTac", green) is positioned in front of an YFP "expression cassette" (or the "YFP RPU cassette") on a plasmid. On the same plasmid, a constitutive promoter ("pCONST") is placed in front of a sensor gate producing the TF, which represses the sensor promoter. In the case of sensor gate characterization, the YFP production is measured in RPU units at different concentrations of inducer (in this case "IPTG"). These data are then fit to obtain the values of y_{max} and y_{min} for the promoter pTac. In this figure: **pTac** (promoter repressed by LacI), **RiboJ** (insulator), **RBS** (Ribosome Binding Site), **YFP** (Yellow Fluorescent Protein coding sequence), **Ter.** (terminator), **pConst** (Constitutive promoter), **LacI** (LacI coding sequence), and **IPTG** ($Isopropyl\beta - D - 1 - thiogalactopyranoside$).

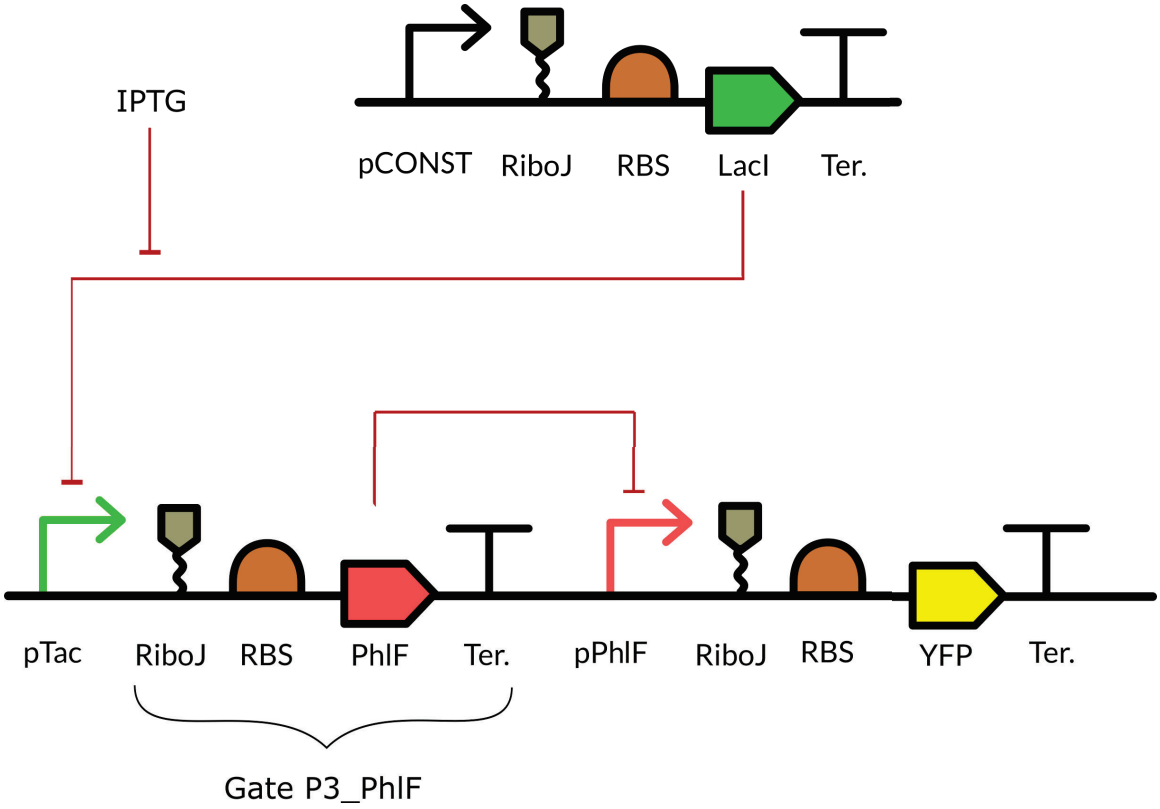


Figure 2.3. Genetic gate parametrization in Cello [16]. Each gate consists of a genetic "expression cassette" (in this case "Gate P3_PhlF") or engineered region that interacts with a promoter (in this case "pPhlF", red). An inducer (in this case IPTG) is added at different concentrations, and the YFP production is measured in RPU units to create a response function (not shown). A Hill equation is fit to the response curve to obtain the values of y_{max}, y_{min}, n, and κ. In this figure: **pTac** (promoter repressed by LacI), **RiboJ** (insulator), **RBS** (Ribosome Binding Site), **YFP** (Yellow Fluorescent Protein coding sequence), **Ter.** (terminator), **pConst** (Constitutive promoter), **LacI** (LacI coding sequence), and **IPTG** ($Isopropyl\beta - D - 1 - thiogalactopyranoside$).

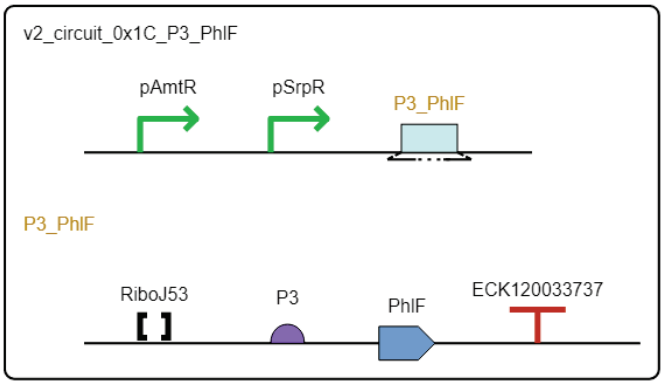


Figure 2.4. SBOL Visual [1] representation of a genetic gate. This gates corresponds to circuit 0x1C in [16]. This gate consists of two repressible promoters (pAmtR and pSrpR) followed by an engineered region or expression cassette (P3_PhlF). This expression cassette is composed of a ribozyme-based insulator (RiboJ), a ribosome binding site (P3), a coding sequence (PhlF), and a terminator (ECK120033737).

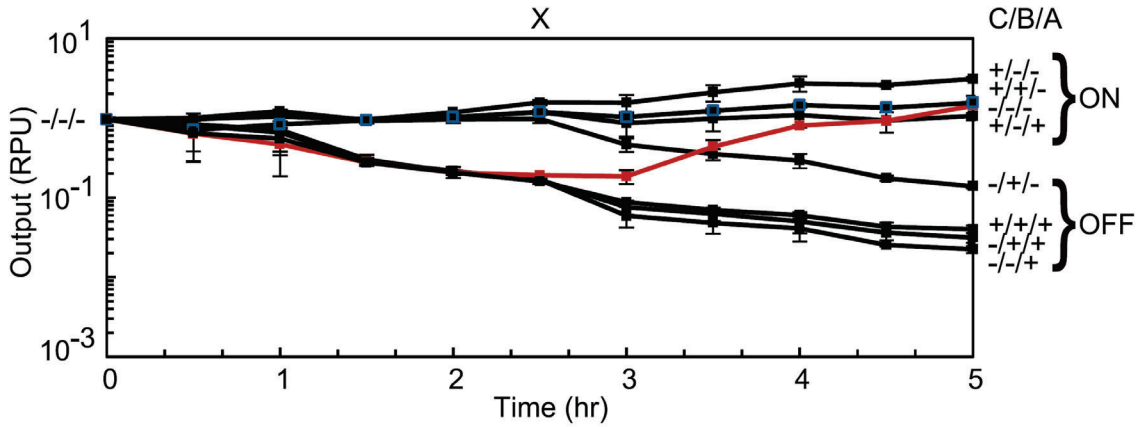


Figure 2.5. Time-course data for circuit 0x8E (courtesy of [16]). Each line represents the output YFP production (in RPUs) over time (in hours) for the circuit 0x8E for each combination of input molecules. This circuit senses three input molecules: Arabinose (Ara), anhydrotetracycline (aTc), and $Isopropyl\beta - D - 1 - thiogalactopyranoside$ (IPTG). In the image, +/+/+ (Ara/aTc/IPTG) represents all input molecules are present and -/-/- represents no input molecules present. Also the *ON* and *OFF* states represent the predicted outcome at steady-state. All outputs behave as expected, except for the +/-/+ state, which experiences an undesirable decay before rising to the *ON* state (red line).

CHAPTER 3

PARAMETERIZED MODEL GENERATOR

This chapter describes the automatic model generator implemented in this work. Section 3.1 starts with the mathematical model used to describe a GRN, which parameters it uses, and where to obtain them. Section 3.2 are the model abstractions used by this model generator to create the production and degradation reaction for each species in a GRN. Section 3.5 continues detailing the different parameters and their units used by the model generator. Finally, the chapter ends in Section 3.6 with a general overview of the algorithm used to generate a mathematical model from an SBOL document automatically, and a simple genetic gate is analyzed and modeled to exemplify the model generator and its outputs.

3.1 Kinetic-Based Mathematical Model

This section describes the mathematical model proposed by Hamid Doosthosseini [27], which is used by the automatic model generator reported in this work to produce the degradation and production reactions described in Sections 3.3 and 3.4. This mathematical model is suitable to model not only genetic circuits generated using the Cello tool but any circuit as long as the appropriate parameters are available. The model stems from the Michaelis-Menten scheme and basic equilibrium kinetics to develop an empirical model that describes the dynamical function of a synthetic genetic circuit, which produces mRNA as well as protein predictions. The model is composed of a set of *Ordinary Differential Equations* (ODEs) that describe the rate of production of mRNA and protein and are rearranged to create variables that can accommodate specific parameters; these equations are solved using an ODE solver using quasi-steady-state assumptions as explained in Section 2.3.5.

A kinetic mass-action model can be used to derive the necessary ODEs that describe the dynamics of a GRN [19, 23, 25, 63]. A simple mass-action kinetic model for the reactions of the transcription and translation processes of a gene can be summarized in a simple

scheme, as shown in **Figure 3.1**. This figure shows a simple diagram of a transcriptional unit (TU) as a section of a DNA strand where there is a promoter (green), followed by an operator (red), and ending with a coding region (purple) of a transcription factor (TF_i). The two most significant steps in the production of a protein (or transcription factor) in a cell are the *transcription* (**Figure 3.1** b) and *translation* (**Figure 3.1** c) processes, which are the production of mRNA and a protein respectively out of a DNA template. We can see that this gene can be repressed by a transcription factor (TF_j) with a binding rate of reaction $k_{1_{repression}}$ and, conversely, it can be unrepressed with a rate of reaction $k_{-1_{repression}}$, as shown in **Figure 3.1** (a). Similarly, **Figure 3.1** (b) shows that there is a rate of binding $k_{1_{transcription}}$ and un-binding $k_{-1_{transcription}}$ for the RNA polymerase (RNAP) onto the promoter segment and a rate of transcription $k_{2_{transcription}}$ by the RNAP, which produces mRNA molecules. For the translation process (**Figure 3.1** c), a ribosome links and unlinks from a strand of mRNA with different rates ($k_{1_{translation}}$ and $k_{-1_{translation}}$, respectively), and eventually, the ribosome starts the translation process with a rate $k_{2_{translation}}$ and produces a protein, or in this model a TF (TF_i), from the mRNA.

If we consider **Figure 3.1** (a) where a transcription-repressor factor binds to the regulatory domain (operator) of a gene, we obtain a mass-action kinetic model for this repression:

$$P_{free} + nTF_j \underset{k_{-1_{repression}}}{\overset{k_{1_{repression}}}{\rightleftharpoons}} P \cdot nTF_j \,, \tag{3.1}$$

in which P_{free} is a promoter that does not have a TF or RNAP bound to it, TF_j is the j^{th} transcription factor of the system that binds to the operator site of this gene and inhibits transcription, and n is the number of TF molecules that bind to an operator site in order to inhibit the transcription, also called binding cooperativity. Similarly, we can produce similar equations for **Figure 3.1** b (3.2) and **Figure 3.1** c (3.3), as shown in the following equations:

$$RNAP_{free} + P_{free} \underset{k_{-1_{transcription}}}{\overset{k_{1_{transcription}}}{\rightleftharpoons}} RNAP \cdot P \xrightarrow{k_{2_{transcription}}} RNAP_{free} + P_{free} + mRNA_i \,, \tag{3.2}$$

$$Rib_{free} + mRNA_i \underset{k_{-1_{translation}}}{\overset{k_{1_{translation}}}{\rightleftharpoons}} Rib \cdot mRNA_i \xrightarrow{k_{2_{translation}}} Rib_{free} + mRNA_i + TF_i \,, \tag{3.3}$$

where $RNAP_{free}$ is unbound RNAP molecules, $mRNA_i$ is the mRNA produced by this gene, Rib_{free} is a ribosome without a linked mRNA, and TF_i is the transcription factor

(protein) produced by this gene, which will, in turn, be a transcription factor for another gene of the circuit. Background production of mRNA, otherwise termed *promoter leakage*, is an important factor to take into account in our model [99]. This would be the production of mRNAs when the promoter is inhibited (for a repressible promoter) or when it is not activated (for a inducible promoter), which will contribute to the y_{min} parameter described later in this chapter. Promoter leakage is described in the following equation as:

$$\emptyset \xrightarrow{k_{i_{background}}} mRNA_i \ . \tag{3.4}$$

There are also multiple and different factors by which $mRNA_i$ and TF_i diminish in concentration. The parameters $k_{mRNA_{i_{dim}}}$ and $k_{TF_{i_{dim}}}$ are a combination of different effects, such as degradation or dilution, that lower the number of mRNA and TF molecules in the cell. They can be modeled as a single rate, as shown in (3.5) and (3.6) that follow:

$$mRNA_i \xrightarrow{k_{mRNA_{i_{dim}}}} \emptyset \ , \tag{3.5}$$

$$TF_i \xrightarrow{k_{TF_{i_{dim}}}} \emptyset \ . \tag{3.6}$$

This set of equations, (3.1) to (3.6), compose the simplified mass-action kinetic model for the production and degradation of mRNA and TF (protein) for a gene with repressible promoters. A pseudo-steady-state approximation (PSSA) can be used to simplify the complex enzymatic reaction descriptions from (3.1), (3.2), and (3.3), which are appropriate for general enzyme systems [23, 100]. This assumption considers that some, or all, of the intermediate enzyme-substrate complexes ($\left[P \cdot nTF_j\right]$, $[RNAP \cdot P]$, and $[Rib \cdot mRNA_i]$), tend to reach a stable-state quickly, meaning that their concentration is constant and their rate of change over time equals 0. Using the PSSA for this system and rearranging (3.1), (3.2), and (3.3) we obtain the following equations:

$$\left[P \cdot nTF_j\right] = \frac{k_{1_{repression}}}{k_{-1_{repression}}} \left[P_{free}\right] \left[TF_j\right]^n \ , \tag{3.7}$$

$$[RNAP \cdot P] = \frac{k_{1_{transcription}}}{k_{-1_{transcription}} + k_{2_{transcription}}} \left[P_{free}\right] \left[RNAP_{free}\right] \ , \tag{3.8}$$

$$[Rib \cdot mRNA_i] = \frac{k_{1_{translation}}}{k_{-1_{translation}} + k_{2_{translation}}} \left[Rib_{free}\right] [mRNA_i] \ . \tag{3.9}$$

We consider that the amount or concentration of total promoters, P_{tot}, is conserved during an experiment (or simulation). The total amount of a promoter in a system would be equal to the amount of unbound promoter (P_{free}), and bound promoter to a transcription factor ($P \cdot nTF_j$) or RNAP ($RNAP \cdot P$) as the next equation describes:

$$\begin{aligned}[P_{tot}] &= [P_{free}] + [P \cdot nTF_j] + [RNAP \cdot P] \\ &= [P_{free}] \left(1 + \frac{k_{1_{repression}}}{k_{-1_{repression}}} [TF_j]^n + \frac{k_{1_{transcription}}}{k_{-1_{transcription}} + k_{2_{transcription}}} [RNAP_{free}]\right) . \end{aligned} \tag{3.10}$$

Now we can define the rate of change in concentration for both mRNAs and TFs using the law of mass action, which states that the rate (or velocity) of a reaction is given by the product of the reactant concentrations to the power of their stoichiometry times the reaction rate constant. Thus, the rate change of mRNA and TF over time, using (3.7), (3.8), (3.9), and (3.10) to replace where appropriate, we obtain:

$$\begin{aligned}\frac{\mathbf{d}[mRNA_i]}{\mathbf{d}t} &= k_{2_{transcription}} [RNAP \cdot P] + k_{i_{background}} - k_{mRNA_{i_{dim}}} [mRNA_i] \\ &= k_{2_{transcription}} \cdot \frac{k_{1_{transcription}}}{k_{-1_{transcription}} + k_{2_{transcription}}} [P_{free}] [RNAP_{free}] + k_{i_{background}} - k_{mRNA_{i_{dim}}} [mRNA_i] \\ &= \frac{k_{2_{transcription}} \cdot \frac{k_{1_{transcription}}}{k_{-1_{transcription}} + k_{2_{transcription}}} [P_{tot}] [RNAP_{free}]}{1 + \frac{k_{1_{repression}}}{k_{-1_{repression}}} [TF_j]^n + \frac{k_{1_{transcription}}}{k_{-1_{transcription}} + k_{2_{transcription}}} [RNAP_{free}]} + k_{i_{background}} - k_{mRNA_{i_{dim}}} [mRNA_i] \end{aligned} \tag{3.11}$$

$$\begin{aligned}\frac{\mathbf{d}[TF_i]}{\mathbf{d}t} &= k_{2_{translation}} [Rib \cdot mRNA_i] - k_{TF_{i_{dim}}} [TF_i] \\ &= k_{2_{translation}} \frac{k_{1_{transcription}}}{k_{-1_{transcription}} + k_{2_{transcription}}} [Rib_{free}] [mRNA_i] - k_{TF_{i_{dim}}} [TF_i] . \end{aligned} \tag{3.12}$$

Equations 3.11 and 3.12 are ODEs that describe the rate of change for mRNA and TF species in the system. Solving these ODEs using an ODE solver method would describe the dynamics of mRNA and TF change in concentration for this system. Note the difference between TF_j and TF_i in this model: the former is the TF that represses this system, the latter is the product of the translation process of this gene.

To be able to accommodate Cello parameters and simplify the equations, we define a new set of parameters as follows:

$$k_{mRNA_{i_{max}}} = \frac{k_{2_{transc}} \cdot \frac{k_{1_{transc}}}{k_{-1_{transc}} + k_{2_{transc}}} [P_{tot}][RNAP_{free}]}{1 + \frac{k_{1_{transc}}}{k_{-1_{transc}} + k_{2_{transc}}} [RNAP_{free}]} , \tag{3.13}$$

$$\kappa^n = \frac{1 + \frac{k_{1_{transc}}}{k_{-1_{transc}} + k_{2_{transc}}} [RNAP_{free}]}{\frac{k_{1_{rep}}}{k_{-1_{rep}}}} , \tag{3.14}$$

$$\alpha_i = \frac{k_{mRNA_{i_{max}}}}{k_{mRNA_{i_{dim}}}} , \tag{3.15}$$

$$\beta_i = \frac{k_{i_{background}}}{k_{mRNA_{i_{dim}}}} , \tag{3.16}$$

$$\gamma_i = \frac{k_{2_{transl}} k_{1_{transl}}}{k_{-1_{transl}} + k_{2_{transl}}} \frac{[Rib_{free}]}{k_{TF_{i_{dim}}}} . \tag{3.17}$$

Thus, (3.11) and (3.12) become

$$\begin{aligned} \frac{\mathbf{d}\,[mRNA_i]}{\mathbf{d}t} &= k_{mRNA_{i_{max}}} \cdot \frac{1}{1 + \left(\frac{[TF_j]}{\kappa}\right)^n} + k_{i_{background}} - k_{mRNA_{i_{dim}}} [mRNA_i] \\ &= k_{mRNA_{i_{dim}}} \left(\frac{\alpha_i}{1 + \left(\frac{[TF_j]}{\kappa}\right)^n} + \beta_i - [mRNA_i] \right) , \end{aligned} \tag{3.18}$$

$$\frac{\mathbf{d}\,[TF_i]}{\mathbf{d}t} = k_{TF_{i_{dim}}} \left(\gamma_i [mRNA_i] - [TF_i] \right) . \tag{3.19}$$

Therefore, at steady-state

$$[mRNA_i] = \frac{\alpha_i}{1 + \left(\frac{[TF_j]}{\kappa}\right)^n} + \beta_i , \tag{3.20}$$

$$\begin{aligned} [TF_i] &= \gamma_i [mRNA_i] \\ &= \frac{\alpha_i \gamma_i}{1 + \left(\frac{[TF_j]}{\kappa}\right)^n} + \beta_i \gamma_i . \end{aligned} \tag{3.21}$$

This form is similar to that of the Hill equation and the model presented in the original Cello paper [16], in where y_{max} and y_{min} are

$$\alpha_i \gamma_i = y_{max} - y_{min} , \tag{3.22}$$

$$\beta_i \gamma_i = y_{min} . \tag{3.23}$$

To replace the values of κ, n, y_{max}, and y_{min} and to obtain a set of first-order ODEs shown below (3.24) and (3.25), $[TF_i]$ must be expressed in units of Relative Promoter Units

(RPU) [60]. However, the units of $[mRNA_i]$ are arbitrary and can be adjusted to simplify notation and simplify the model. We can modify the value of γ_i so that $\gamma_i [mRNA_i] = \left[\widehat{mRNA_i}\right]$, and this can be accomplished by setting $\gamma_i = 1$ and therefore the final form for the set of ODEs for this model would be:

$$\frac{\mathbf{d}\left[\widehat{mRNA_i}\right]}{\mathbf{d}t} = k_{mRNA_{i_{dim}}} \left(\frac{y_{max} - y_{min}}{1 + \left(\frac{[TF_j]}{\kappa}\right)^n} + y_{min} - \left[\widehat{mRNA_i}\right] \right) , \tag{3.24}$$

$$\frac{\mathbf{d}\,[TF_i]}{\mathbf{d}t} = k_{TF_{i_{dim}}} \left(\left[\widehat{mRNA_i}\right] - [TF_i] \right) . \tag{3.25}$$

We can apply the same procedure for a system in which a TF activates, instead of represses, the expression of a gene that would lead to an equation (3.26) of the following form:

$$\frac{\mathbf{d}\left[\widehat{mRNA_i}\right]}{\mathbf{d}t} = k_{mRNA_{i_{dim}}} \left(\frac{y_{max} - y_{min}}{1 + \left(\frac{\kappa}{[TF_j]}\right)^n} + y_{min} - \left[\widehat{mRNA_i}\right] \right) . \tag{3.26}$$

Thus, (3.24), (3.25), and (3.26) are the production rates of mRNA and TF of a system, where y_{max} and y_{min} are the production rates of a promoter when it is *on* or *off*, respectively (in relative promoter units [60]), κ and n are obtained from the affinity and cooperativity of binding, and finally, $k_{mRNA_{i_{dim}}}$ and $k_{TF_{i_{dim}}}$ are the composed rates of degradation, diffusion, or any other diminishing phenomenon that decreases the concentration of the mRNA and protein, respectively. These last parameters can be obtained from literature or fitted to experimental data. This is a mathematical model for repressions, activations, transcription, and translation, which is used in the automatic model generator implemented in this work, as shown below.

3.2 Model Abstractions

Applying model abstractions simplifies the original mathematical model and, therefore, speeds up simulation time [19]. Nonetheless, this reduction of complexity requires parametrization of these simplified reactions [101]. The following two subsections explain the abstractions that are considered in this work and how the accompanying parts characterizations can be used to parameterize these abstracted reactions.

3.2.1 Sensor Gates

A *sensor gate* is a TU that produces a sensor protein. Such a protein is called a *sensor* protein because it can bind to small inducer molecules, called *inputs*, "sensing" their presence/absence. These small inducer molecules are called inputs since they are externally controlled by the experimenter and can transfer into the cytoplasm of a cell. These inputs can bind to sensor proteins to form complexes that have no function and are later degraded by the cell (see **Figure 3.2** (a)), thus inhibiting (or repressing) these sensor proteins. Alternatively, some sensor proteins are inactivated unless they bind to an input molecule (see **Figure 3.2** (b)), in which case they undergo a conformational change and can then induce positively or negatively a promoter.

Inducible promoters that are repressed/activated by these sensor proteins are thus called *sensor promoters* because their activation/repression (*on* and *off* states) can be controlled by the inputs that the experimenter can control.

Cello uses sensor gates with constitutive promoters [16], as shown in **Figure 3.3**. A constitutive promoter is not inducible and is always *on*, meaning there is always going to be production of sensor molecules when it is included on a designed system. It is therefore safe to assume there is always presence of sensor molecules on the system if a sensor gate is included, and thus the modeling for the sensor molecule production can be avoided to speed up model generation and simulation time. This is why the model generator implemented in this work does not produce a mathematical model for the production of sensor mRNAs or proteins, and no species are created for the sensor gate. We can further abstract this model by avoiding simulating the complex formation between sensor proteins and inputs, as explained in the following section.

3.2.2 Circuit Inputs and Sensor Promoters

As explained previously, sensor proteins can activate/repress sensor promoters, and small inducer molecules (inputs) can activate/repress these sensor proteins. Assuming there is always a presence of a sensor molecule, as explained in the previous section, the model can be further abstracted avoiding the modeling of complex formation between sensor and small molecules to directly representing the small molecules as activating or repressing the sensor promoters, as shown in **Figure 3.4**.

We can simplify the circuit because the repression of a repression has analogous functionality as an activation. The designed circuit could have, for example, a sensor protein that represses a sensor promoter like in **Figure 3.4** (a). If this sensor protein can bind to an input molecule that renders it inactive and tags it for degradation (like in **Figure 3.2** (a)), then it could be modeled as if the input molecule *represses* the sensor protein (see **Figure 3.4** (b)), thus circumventing the need to model the complex formation between the sensor protein and input molecule. It can be noted that a repression of a repression is equivalent to an activation further simplifying the model (**Figure 3.4** (c)). This reduction works similarly if the input binds to a sensor protein to activate it, as shown in **Figure 3.2** (b); therefore it can be simplified as the input molecule directly activating the sensor promoter. Hence the model generator implemented in this work simplifies inducibility of a sensor promoter and complex formation by a sensor protein directly as an input molecule inducing a sensor promoter. It is possible to make this abstraction since sensor proteins are continually produced, so they are always present in the system, and we consider that the external input molecules, when present, will significantly surpass in quantity the number of sensor proteins, thus binding and activating/inhibiting all of the sensor proteins present. For this model generator, the concentration of input molecules is not essential since we consider that a sufficient amount is introduced into the system by the experimenter, and, as a consequence, we are only going to model presence/absence of input molecules that will activate the sensor promoters when present (see Section 3.4 for more information).

There are four types of sensor proteins in the original Cello paper [16]: LacI, TetR, AraC, and LuxR. LacI and TetR repress their respective sensor promoters unless they bind to an input molecule (IPTG and aTc, respectively), which renders them inactive. On the other hand, LuxR and AraC can only positively induce their respective sensor promoters if they bind to an input molecule (HSL and Ara, respectively). In all of the cases stated, we can reduce and simplify the model as if the input molecules directly activate the sensor promoters. Thus, the presence of any of the input molecules (IPTG, aTC, HSL, and Ara) will promote transcription of TU with sensor promoters (pTac, pTet, pLuxStar and pBAD respectively), as shown in Table 3.1.

Consequently, the model generator implemented in this work does:

- Not produce a SBML species for any sensor TU or sensor protein,

- Not generate a sensor mRNA and protein production and degradation reaction,
- Avoid generating complex formation reactions between sensor proteins and input molecules, and
- Summarize or abstract all interactions with sensor promoters as activations from input molecules.

These abstractions help speed up the model generating process, as well as the simulation time in an effort to abstract the model without losing predictability.

3.3 New Degradation Reaction

The reactions generated by the automatic model generator of this work are presented in this section and the following one. This reactions stem from the dynamic model presented earlier in Section 3.1 and are automatically generated without the intervention of the user. The model generator implemented in this work generates a production and degradation reaction for each SBML species in the model, except for the TUs and input molecules, since neither of those is going to be produced or degraded during simulation. That means that only TFs and mRNAs will have a production and degradation reaction associated with them. The Equations 3.24 and 3.25, which represent the *change* in concentration for both these species, have factors that diminish and increment species concentrations. Discriminating between these factors will create a production and degradation reaction for both TFs and mRNA. Unfactorizing both equations we obtain

$$\frac{\mathbf{d}\left[\widehat{mRNA_i}\right]}{\mathbf{d}t} = k_{mRNA_{i_{dim}}}\left(\frac{y_{max}-y_{min}}{1+\left(\frac{[TF_j]}{\kappa}\right)^n}+y_{min}\right) - k_{mRNA_{i_{dim}}}\cdot\left[\widehat{mRNA_i}\right] \ , \tag{3.27}$$

$$\frac{\mathbf{d}\left[TF_i\right]}{\mathbf{d}t} = k_{TF_{i_{dim}}}\cdot\left[\widehat{mRNA_i}\right] - k_{TF_{i_{dim}}}\cdot\left[TF_i\right] \ . \tag{3.28}$$

Since y_{max} and y_{min} are always positive numbers, and $y_{max} > y_{min}$, it can be deduced that the only terms that diminish concentration are the negative terms. Therefore, the degradation reaction produced for each mRNA and TF in this model generator will be

- *mRNA degradation reaction*:

$$\textbf{mRNA Degradation Reaction} = -k_{mRNA_{i_{dim}}} \cdot \left[\widehat{mRNA}_i\right] \ , \tag{3.29}$$

and

- *TF degradation reaction*:

$$\textbf{TF Degradation Reaction} = -k_{TF_{i_{dim}}} \cdot [TF_i] \ . \tag{3.30}$$

The model generator creates only one degradation reaction for each SBML species that degrades (TFs, mRNAs), and it will depend only on the diminishing constants $k_{mRNA_{i_{dim}}}$ and $k_{TF_{i_{dim}}}$ and the concentration of the species itself.

3.4 New Production Reaction

Grouping the positive terms will produce the production reactions for the different SBML species that are produced in this model (TFs and mRNAs) as follows:

- *mRNA production reaction*:

$$\textbf{mRNA Production Reaction} = k_{mRNA_{i_{dim}}} \left(\frac{y_{max} - y_{min}}{1 + \left(\frac{[TF_j]}{\kappa}\right)^n} + y_{min} \right) \ , \tag{3.31}$$

and

- *TF production reaction*:

$$\textbf{TF Production Reaction} = k_{TF_{i_{dim}}} \cdot \left[\widehat{mRNA}_i\right] \ . \tag{3.32}$$

Cello's algorithms produce gates with tandem promoters (see **Figure 2.4**), choosing from a library of parts and avoiding those with known roadblocking issues. The model carried out in this work assumes the promoters are completely orthogonal. This means that each promoter will be independent of which other promoter is in tandem with and that the effect over the rate of production of mRNA for this gate is *additive*. This is a safe assumption since Cello only chooses tandem promoters that do not display roadblocking behavior.

Figure 2.4 shows an example of a gate produced for the circuit 0xC1 in the original Cello paper [16]. In this case, the production reaction for $mRNA_{PhlF}$ would be

$$\frac{\mathbf{d}\left[\widehat{mRNA_{PhlF}}\right]}{\mathbf{d}t} = k_{mRNA_{i_{dim}}}\left(\frac{y_{max_{pAmtR}} - y_{min_{pAmtR}}}{1+\left(\frac{[AmtR]}{\kappa_{AmtR}}\right)^{n_{AmtR}}} + y_{min_{pAmtR}}\right) + k_{mRNA_{i_{dim}}}\left(\frac{y_{max_{pSrpR}} - y_{min_{pSrpR}}}{1+\left(\frac{[SrpR]}{\kappa_{SrpR}}\right)^{n_{SrpR}}} + y_{min_{SrpR}}\right) .$$

It should be noted that all gates with sensor promoters will have a similar form, but since sensor promoters are considered only to be *on* or *off* if there is a presence/absence of input molecules, the production reaction for them is a piece-wise function like the following:

$$\frac{\mathbf{d}\left[\widehat{mRNA_{PhlF}}\right]}{\mathbf{d}t} = k_{mRNA_{i_{dim}}} \cdot \left(piecewise(y_{min_{pTet}}, TetR == 0, y_{max_{pTet}})\right) .$$

This means that the promoter is at minimum production rate (y_{min}) when there is no input molecule ($[TetR] == 0$), and at maximum production rate (y_{max}) when there is presence of input molecule ($[TetR] \neq 0$). As explained in Section 3.2.1, the model does not take into account input molecule concentration. It should be noted that for this work all the values of $k_{mRNA_{i_{dim}}}$ and $k_{TF_{i_{dim}}}$ are the same for all the different genetic parts since there is no information available of their real magnitudes, but future work using RNAseq and Riboseq data can provide information on these constants for each gate as explained in Section 3.5.2.

3.5 Parameters and Units

The mathematical model described requires six different parameters: y_{max}, y_{min}, n, κ, $k_{mRNA_{i_{dim}}}$, and $k_{TF_{i_{dim}}}$. Since the units for y_{max} and y_{min} are in *Relative Promoter Units* (RPU) [60], and the units for $k_{mRNA_{i_{dim}}}$ and $k_{TF_{i_{dim}}}$ are per seconds (s^{-1}), the output rate of the production and degradation reactions are in RPU per second. It is useful to report rate of changes in RPUs since it is a standardized unit for measuring promoter activity and has an associated standardized protocol for measurement that is compatible with different laboratories with different equipment.

Described in the following subsection is how the model generator obtains the first four parameters (y_{max}, y_{min}, κ, and n) from any online repository that uses the same parametrization as the Cello tool does. The next subsection explains what the values are for $k_{mRNA_{i_{dim}}}$

and $k_{TF_{i_{dim}}}$ chosen for this work, and how to improve these in the future to better fit the model predictions to experimental data.

3.5.1 y_{max}, y_{min}, κ, and n

The values for y_{max}, y_{min}, n, and κ were obtained experimentally in the Cello project [16] and can be retrieved from the appropriate repository where they are stored. The model generator of this work does so by opening a front-end in SynBioHub and retrieving this information from the appropriate repository.

In this model, y_{max} is the maximum RPU produced for a gene, when the promoter is *active* or *on*; y_{min} would be the basal or background transcription in RPU of a gene or gate with an *inhibited* or *off* promoter; n is equivalent to the Hill coefficient, and κ is equivalent to the dissociation constant.

Genes and proteins behave differently depending on the genetic and cellular context [102]; thus, there is an assumption that they will behave similarly once the genetic circuit is assembled, and hence, it is an assumption of the model developed in this work. Nonetheless, these parameters can be fitted to experimental data (RNA-seq and Ribo-seq data) to estimate more accurately what is the value when implemented on a specific circuit and strain. There is an inherent risk when fitting parameters to experimental data, and that is over-fitting, which can result in significant errors if the model is applied to other circuits. However, if the interest is on a specific genetic circuit implementation, then it would be favorable to fit the parameters to experimental data to have a better model prediction and finer debugging capabilities when analyzing the results.

3.5.2 $k_{mRNA_{i_{dim}}}$ and $k_{TF_{i_{dim}}}$

The parameters $k_{mRNA_{i_{dim}}}$ and $k_{TF_{i_{dim}}}$ represent the decay or degradation of mRNA and protein, respectively, and their experimentally acquired values pool a variety of different phenomenon such as degradation from cell processes to dilution due to cell growth. Nonetheless, since there is no time-series data available for the expression profiles of the different genes, the values for $k_{mRNA_{i_{dim}}}$ and $k_{TF_{i_{dim}}}$ were chosen from literature [64, 103–105] as $\frac{1}{300s^{-1}}$ and $\frac{1}{3600s^{-1}}$, respectively. These values were chosen from studies of mRNA and protein decay rates for *Escherichia coli* and represent an average across multiple experimental conditions as well as types of mRNA and proteins. However, each mRNA

and protein have different values for $k_{mRNA_{i_{dim}}}$ and $k_{TF_{i_{dim}}}$, and it would be interesting to obtain them experimentally for each genetic gate, in order to enrich the library of genetic parts and their parametrization. RNAseq and Ribo-seq time-series data can also be used to fit these values to the observed data and obtain more precise values for $k_{mRNA_{i_{dim}}}$ and $k_{TF_{i_{dim}}}$, as well as for y_{max}, y_{min}, n, and κ.

Even though it is valuable to derive parametrization of $k_{mRNA_{i_{dim}}}$ and $k_{TF_{i_{dim}}}$ for each mRNA and protein in the system to attain more precise models, it is important to note that the values of these two parameters do not affect the steady-state outcome of each species. If we group the production and degradation reaction ((3.29) and (3.31) for mRNAs and (3.30) and (3.32) for proteins) into the same equation, we obtain the total rate of change for mRNA or protein as follows:

$$\frac{\mathbf{d}\left[\widehat{mRNA_i}\right]}{\mathbf{d}t} = k_{mRNA_{i_{dim}}}\left(\frac{y_{max}-y_{min}}{1+\left(\frac{[TF_j]}{\kappa}\right)^n}+y_{min}\right) - k_{mRNA_{i_{dim}}}\cdot\left[\widehat{mRNA_i}\right] \ ,$$

$$\frac{\mathbf{d}\left[TF_i\right]}{\mathbf{d}t} = k_{TF_{i_{dim}}}\cdot\left[\widehat{mRNA_i}\right] - k_{TF_{i_{dim}}}\cdot\left[TF_i\right] \ .$$

These equations describe the rate of change for any mRNA and protein for the system. At steady-state $\frac{\mathbf{d}\left[\widehat{mRNA_i}\right]}{\mathbf{d}t} = 0$ and $\frac{\mathbf{d}[TF_i]}{\mathbf{d}t} = 0$. Therefore,

$$0 = \frac{y_{max}-y_{min}}{1+\left(\frac{[TF_j]}{\kappa}\right)^n}+y_{min} - \left[\widehat{mRNA_i}\right] \ , \tag{3.33}$$

$$0 = \left[\widehat{mRNA_i}\right] - [TF_i] \tag{3.34}$$

means that the steady-state of these species does not depend on the values of $k_{mRNA_{i_{dim}}}$ and $k_{TF_{i_{dim}}}$. Nonetheless, these two parameters can affect the response function of the genetic gates. Modifying these rates can account for slower or delayed response, as shown in **Figure 3.5**.

It would prove valuable to obtain time-series data for the response functions of the different gates used in this work, in order to fit the values of $k_{mRNA_{i_{dim}}}$ and $k_{TF_{i_{dim}}}$ and have more accurate dynamic response predictions. However, for the purpose of this thesis work, the generic values of $k_{mRNA_{i_{dim}}}$ and $k_{TF_{i_{dim}}}$ are adequate to provide a general dynamic model prediction and analysis of behavior.

3.6 Automated Model Generator

Taking into consideration all previous sections, we can now illustrate the high-level description of the dynamic model generator implemented in this work and a mock example of its output. In it, a general description of how the algorithm creates a model, species, and reactions from an enriched SBOL document that describes the GRN to be modeled is detailed. The complete algorithm of the dynamic model generator can be accessed at `https://github.com/MyersResearchGroup/iBioSim/tree/ModelingGenerator`.

As a demonstration, this section shows the output of the model generator for a simple genetic gate (see **Figure 2.4**) in iBioSim. This demonstration is meant to show what the user would see when generating a model for a genetic circuit using the model generator derived in this work. One would start with a design of the circuit, either by creating it within iBioSim with SBOLDesigner, or importing the SBOL file with the design specification, as shown in **Figure 3.6**. Once the design is created/uploaded, the user would have to double-click on the panel on the left and then click on the "*Generate Model*" button, as shown in **Figure 3.7**. A pop-up window appears so that the user can select the appropriate online repository, as shown in **Figure 3.8**, and the automated model generator produces the dynamic model, as shown in **Figure 3.9**. In the process of generating the model, the algorithm will:

- Use VPR to enrich the SBOLDocument with interactions and their participants from the chosen repository and create a Top Module that encompasses all the different sub-modules.
- Extract the parameter information from the parts.
- Create an SBML document and an empty model.
- Create all the species and reactions in the model.
- Generate the mathematical equations that compose the dynamic model for the chosen parts.

Once it is done, the user can look for the mathematical formulas of each reaction under the "tabular" tab. The equations generated for this example are shown in (3.35).

$$mRNA_{PhlF}\ \textbf{production reaction} = k_{mRNA_{i_{dim}}}\left(\frac{y_{max_{pAmtR}} - y_{min_{pAmtR}}}{1+\left(\frac{[AmtR]}{\kappa_{AmtR}}\right)^{n_{AmtR}}} + y_{min_{pAmtR}}\right) +$$
$$k_{mRNA_{i_{dim}}}\left(\frac{y_{max_{pSrpR}} - y_{min_{pSrpR}}}{1+\left(\frac{[SrpR]}{\kappa_{SrpR}}\right)^{n_{SrpR}}} + y_{min_{SrpR}}\right)\ ,$$
$$PhlF\ \textbf{production reaction} = k_{TF_{i_{dim}}} \cdot [mRNA_{PhlF}]\ ,$$
$$mRNA_{PhlF}\ \textbf{degradation reaction} = k_{mRNA_{i_{dim}}} \cdot [mRNA_{PhlF}]\ ,$$
$$PhlF\ \textbf{production reaction} = k_{TF_{i_{dim}}} \cdot [PhlF]\ . \quad (3.35)$$

The mathematical model describes how the different species change depending on other chemical species and inputs, but it does not describe in which context will this circuit be implemented. To specify the environment and sequence of inputs this circuit will be subjected to, a simulation environment has to be created where the user can designate input value changes over time. One such simulation environment could be subjecting the circuit to all the different combinations of input molecules, as shown in **Figure 3.10**. In the simulation environment shown in this example, the input values alternate to all possible combinations of presence/absence of input molecules, in the order shown in [16]. Simulation of this environment would produce the expected dynamic behavior of the genetic gate or circuit. For this example, simulation would be uninformative since it is only one gate of a genetic circuit which does not respond to changes in input molecules. However, Chapter 4 demonstrates the dynamic simulation produced by the model generator and this environment, for all the circuits depicted in the original Cello paper, and the analysis we can do on such simulations.

Table 3.1. Model abstractions for input molecules and sensor promoters

Sensor Promoter	Sensor Protein	Input Molecule	Abstracted Interaction
pTac	LacI	IPTG	*activates*
pTet	TetR	aTc	*activates*
pLuxStar	LuxR	HSL	*activates*
pBAD	AraC	Ara	*activates*

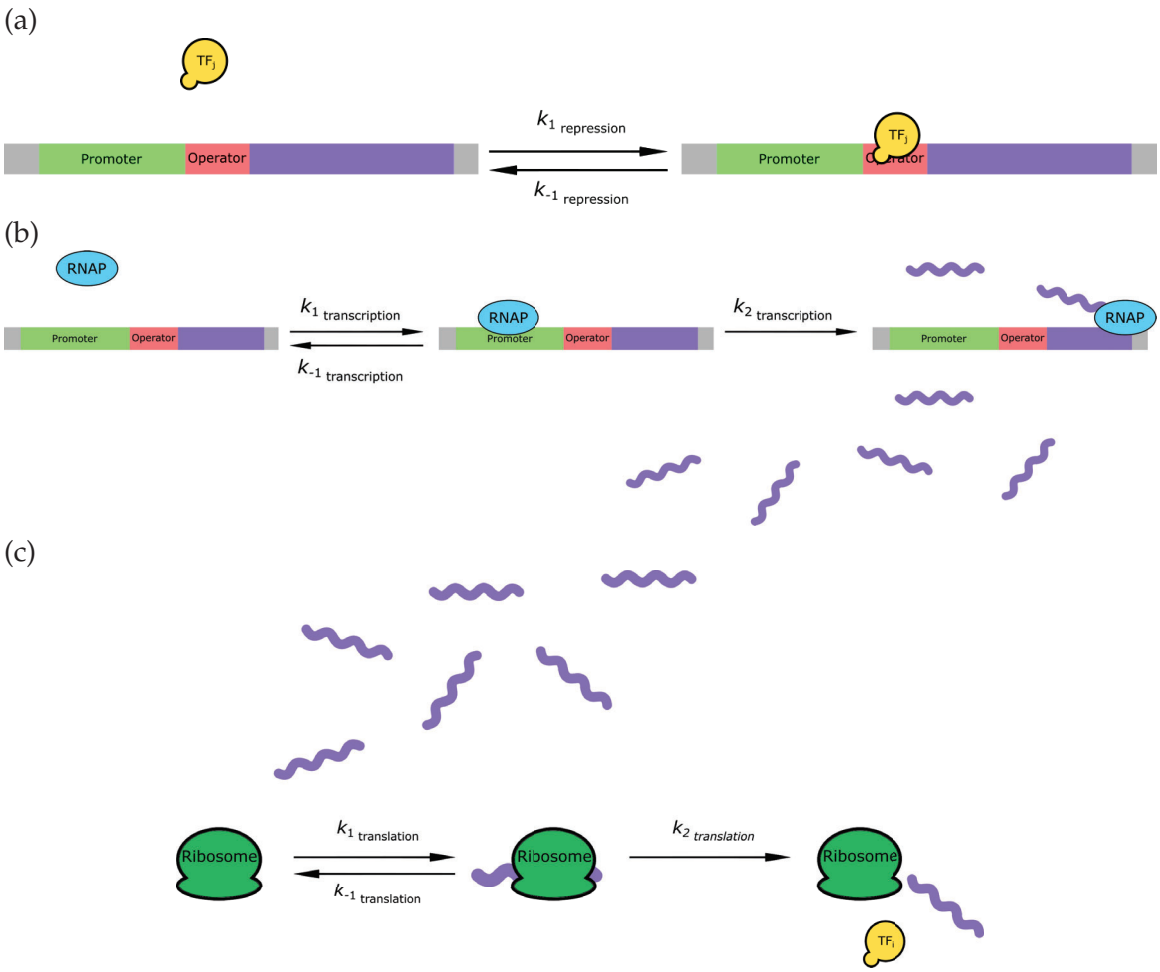


Figure 3.1. Schematic of kinetic model for repression, transcription, and translation. (a) A simple kinetic model of a repression interaction of when a *transcription factor* (TF) attaches to a operator site, blocking transcription. (b) A simple kinetic model of transcription, where a RNAP binds to a promoter site and starts transcribing a mRNA molecule from the DNA template. (c) A simple kinetic model of a translation process, where a ribosome links to a mRNA and produces a protein (or TF) specified in the codon sequence of the mRNA.

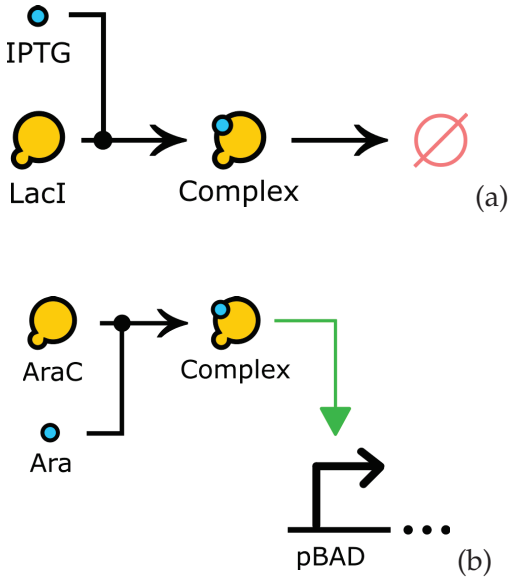


Figure 3.2. Complex formation between an input molecule and a sensor protein. (a) A sensor protein (LacI) can bind with an input molecule (IPTG) to form a complex that does not induce any promoter and is tagged for degradation. (b) The sensor protein (AraC) is inactive unless it binds to an input molecule (Ara) to form a complex that can positively induce a promoter (pBAD).

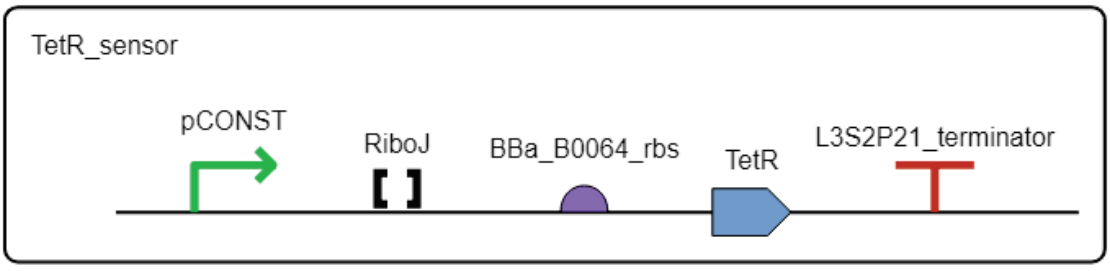


Figure 3.3. Representation of a sensor gate used by Cello. Sensor molecule TetR is being constantly produced since this TU has a constitutive promoter (pCONST).

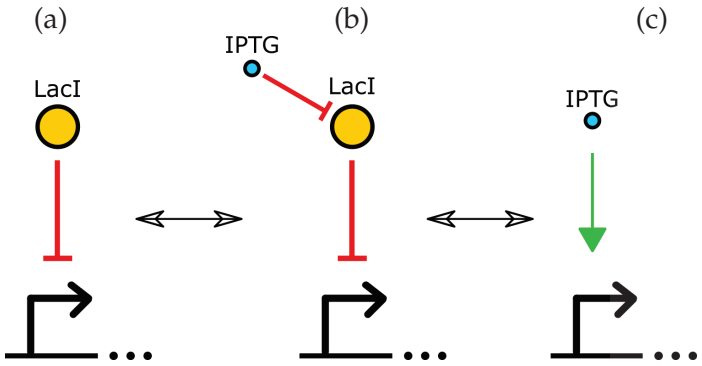


Figure 3.4. Model abstractions permitted when sensor proteins are available in abundance. (a) Schematic showing a sensor protein that represses a sensor promoter. (b) A small molecule (input) that represses the sensor protein. (c) Model abstraction where the small molecule can be considered to activate the sensor promoter.

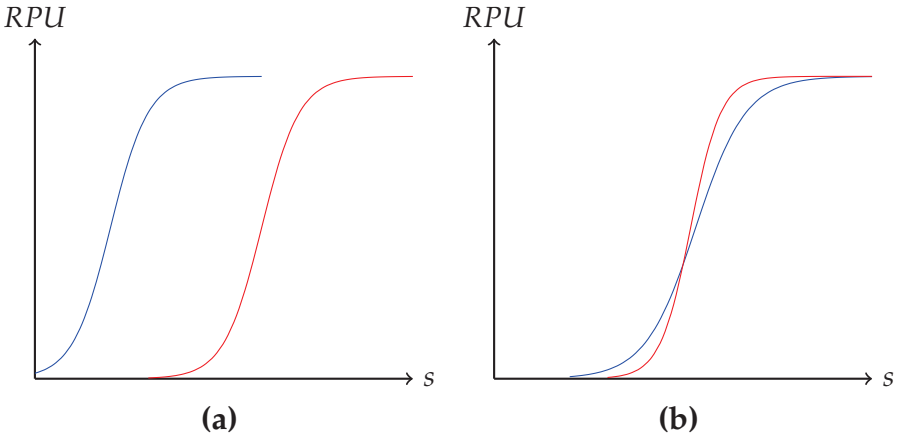


Figure 3.5. Changing dynamic response with $k_{mRNA_{i_{dim}}}$ and $k_{TF_{i_{dim}}}$. Changing the values of $k_{mRNA_{i_{dim}}}$ or $k_{TF_{i_{dim}}}$ can cause a delayed response (a), a faster response (b), or a combination of both.

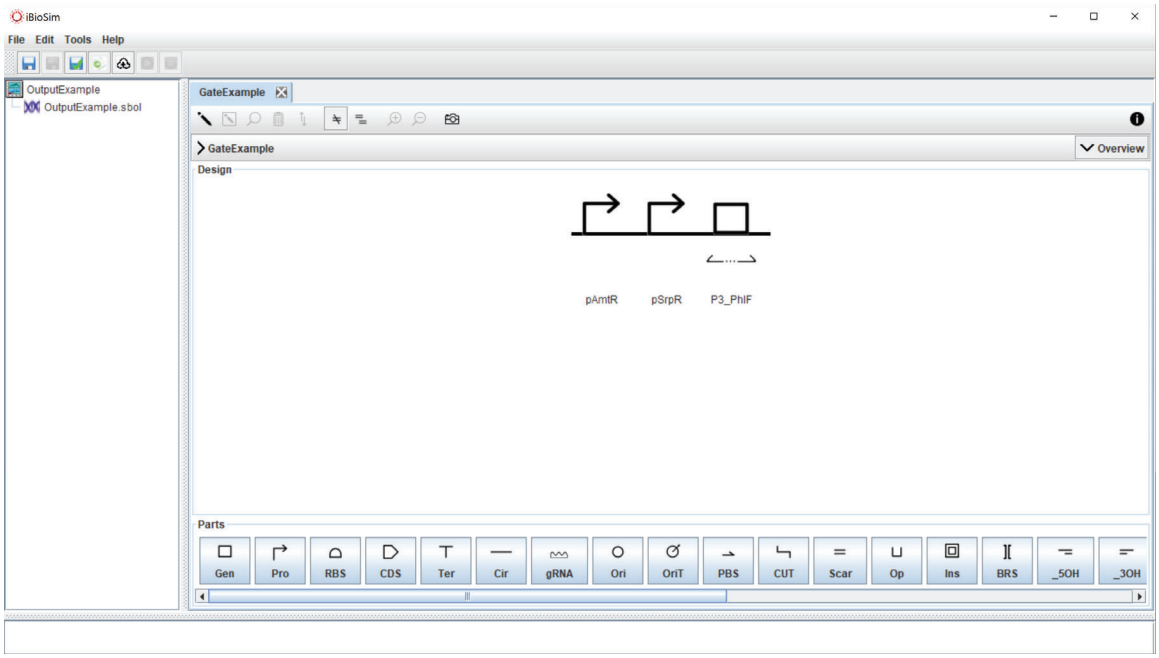


Figure 3.6. Designing a genetic NOR gate in iBioSim using SBOLDesigner.

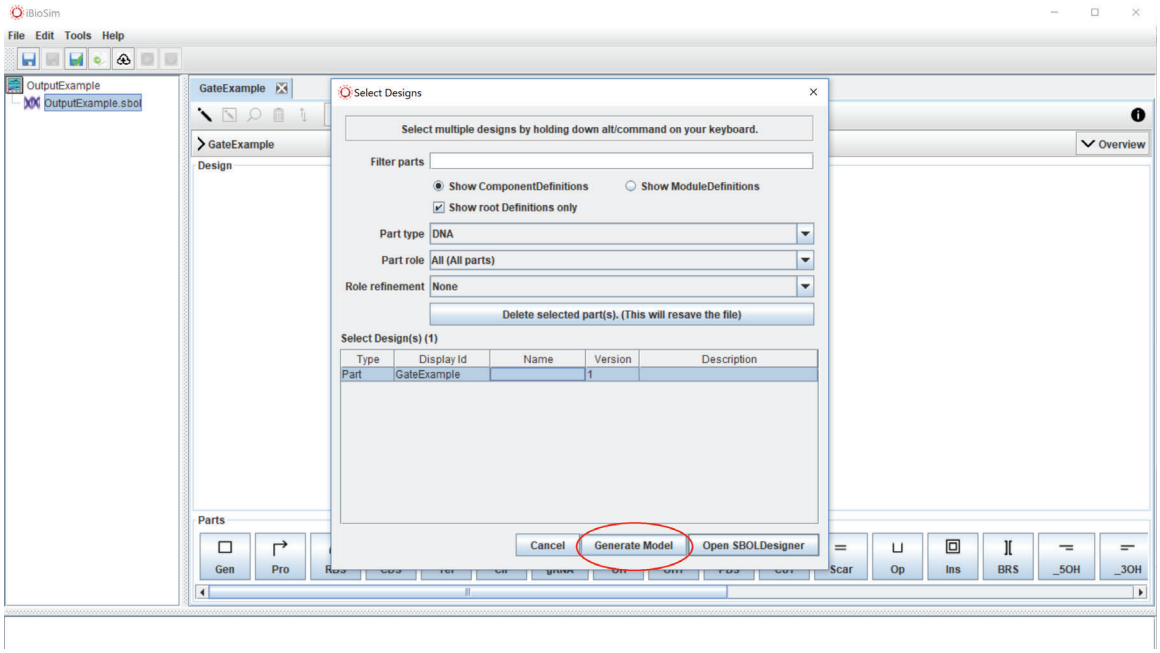


Figure 3.7. Automatically generating a model for a circuit design. Once the user clicks the circled button in iBioSim, the algorithm implemented in this work starts to automatically generate the model for the selected gates/circuits.

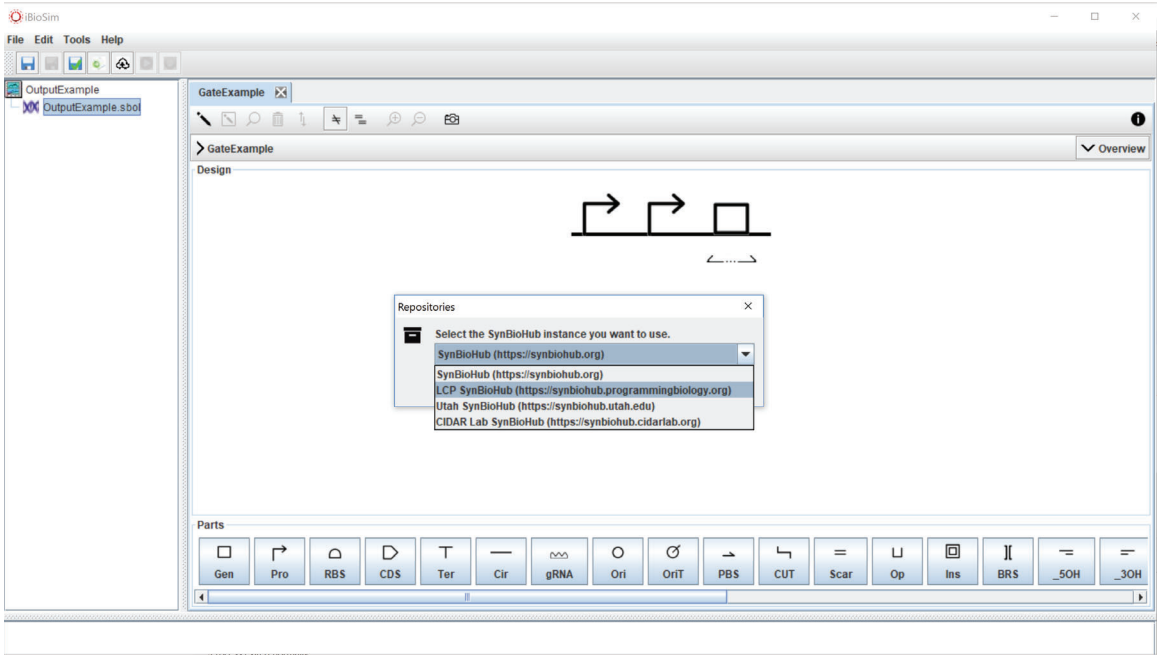


Figure 3.8. Selecting online repository in iBioSim. A pop-up window appears in iBioSim after selecting "Generate Model" in which it asks for the user to select an online repository to fetch information about the parts used in the genetic design.

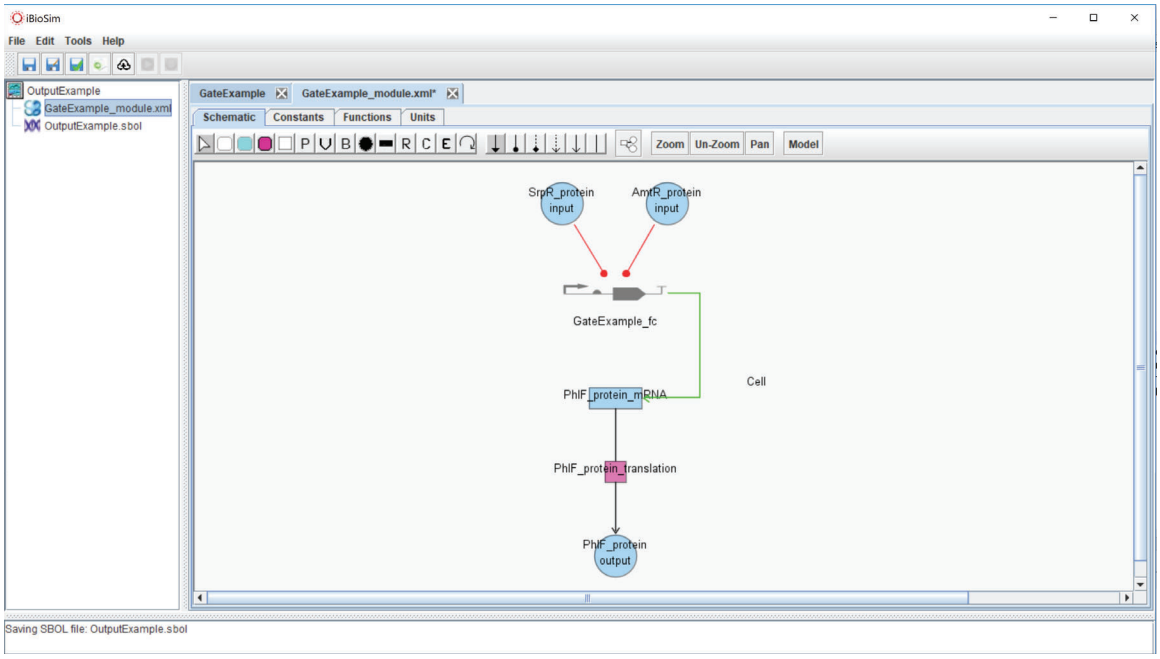


Figure 3.9. Mathematical model output in iBioSim. Graphical exposition of the mathematical model generated for the selected genetic gate/circuit in iBioSim.

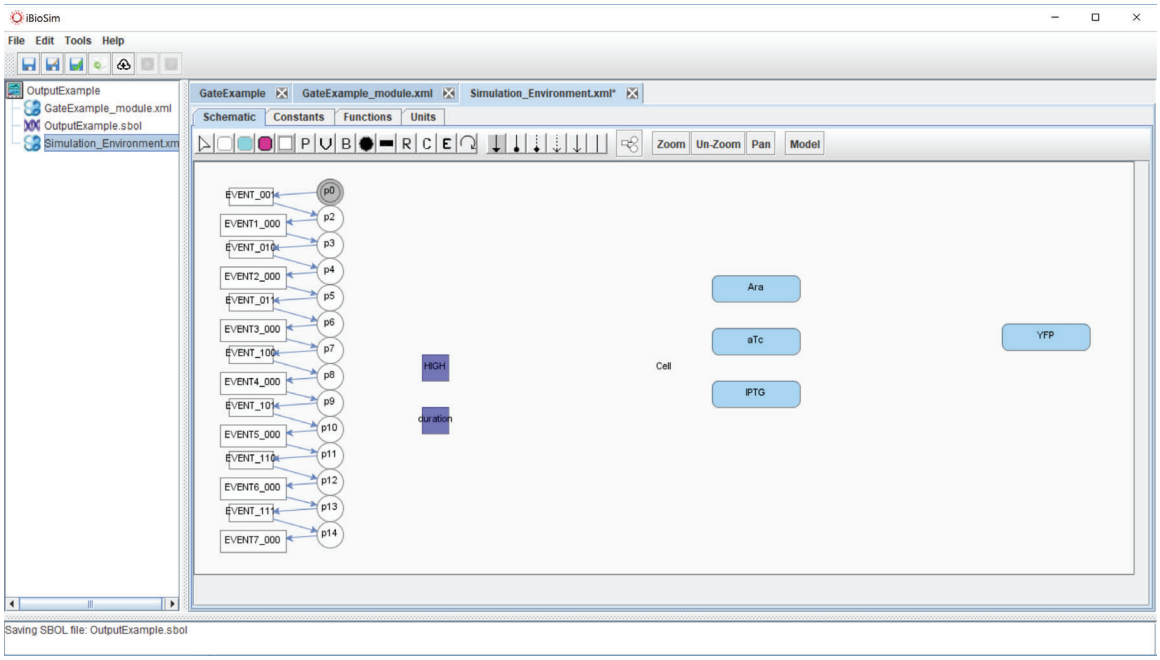


Figure 3.10. Simulation environment in iBioSim. Simulation Environment created in iBioSim to simulate the different input combinations the circuit can be subjected to. The order of these input combinations follows the one shown in the Cello paper [16].

CHAPTER 4

CASE STUDIES

This chapter provides the results obtained from using the automated model generator on preexisting genetic circuits designed with Cello [16]. A full workflow, from importing the genetic circuits encoded in SBOL files, to uploading the finished models and their simulation to an online repository is performed during the process. However, this chapter is not limited to only showcasing these results. In further sections, we use the simulation results to analyze these genetic circuits and show how the dynamic behavior of a circuit can help understand the underlying phenomenon to some of the undesired or unexpected behavior predicted.

4.1 Cello Circuits

The Cello project [16] designed 52 different circuits and simulated them using steady-state modeling. Using the dynamic model generator of this work, we set out to generate dynamic models and simulate these same circuits. However, none of the models would be interesting by themselves if not instantiated in a simulation environment. This simulation environment should describe the input changes the circuit is subjected to, and therefore would provide information on how these circuits respond to them. In the next section, a description of the simulation environment used in this project ensues.

4.1.1 Simulation Environment

Figure 4.1 shows a schematic of the simulation environment, which was created using SBML editor in iBioSim. For each genetic circuit, an iBioSim project was created, and the automatic model generator implemented in this work was used to create a dynamic model for each. Then, the simulation environment model shown in **Figure 4.1** was imported for each project, and the corresponding replacements were done. This simulation environment goes through all the different states the circuits in the Cello paper were subjected to,

both in experimentation and simulation environments.

Once the simulation environment was imported, the dynamic model of the circuit was instantiated, and the input molecules and YFP replacements from these two hierarchical models were done, the simulation proceeded. All cases were simulated for 750000 seconds (208.3 hours) to allow the circuit to reach steady-state for each input combination, before changing to the next. To solve the ODEs that were created by the automatic model generator of this work, the Runge–Kutta–Fehlberg (4, 5) method, offered by the iBioSim tool, was chosen. The simulation results are shown in the next section.

4.1.2 Results

Using the SBOL files that describe each of these circuits generated by Cello, and the parameters stored at `https://synbiohub.programmingbiology.org/public/Eco1C1G1T1/Eco1C1G1T1_collection/1`, we generated models for each of the circuits of the Cello work. For each of these models, the simulation environment described earlier was used to simulate the same input changes as in the Cello project [16]. After simulation, we uploaded all of these SBOL files, models, and simulations as collections in a SynBioHub repository. [1]

An example of one such simulation result is shown in **Figure 4.2**. This figure shows the YFP production (in RPU) over time (seconds) predicted for circuit 0x4D [16] using the automatic model generator of this work.

The simulation alternates through all the states the circuit was subjected to in the Cello project. The output of the circuit (YFP production, yellow line) is considered to be *ON* for the same states as in the Cello paper [16] shown in **Figure 4.3**.

The same has been observed for the other circuits simulated, showing that the model generator implemented in this work successfully predicts the same steady-states as the steady-state modeling in the Cello project. However, the model generator of this work also predicts the dynamic behavior between these steady-states, not available in the Cello paper [16].

In the dynamic prediction shown in **Figure 4.2**, some spikes in YFP production can be observed before the system reaches steady-state. This behavior would not have been

[1] `https://synbiohub.utah.edu/public/DynamicModelGenerator/DynamicModelGenerator_collection/1`

possible to predict using a steady-state modeling, as in the Cello project, and it is the reason why implementing a dynamic modeler, like the one on this work, is so important. Other circuits simulated presented other unexpected behavior, and is discussed further in the next section.

4.2 Unexpected Behavior

The YFP production spikes of Circuit 0x4D is just one example of unexpected dynamical behavior when designing these genetic circuits [16]. Many other circuits simulated present some spikes like the one shown in **Figure 4.2**, but also other unexpected behaviors like an undesirable decay on YFP production before rising to the *ON* state.

One such circuit is 0x8E [16], which its dynamic simulation produced using the dynamic modeler of this work is shown in **Figure 4.4**. When transitioning from the *no inputs* state 0/0/0 (Ara = 0 / IPTG = 0 / aTc = 0), to the state where IPTG and Ara are present (1/1/0), there is an unexpected decay in YFP production before reaching the *ON* steady-state. This was observed experimentally in the original Cello project [16], as shown in **Figure 4.5** and explained in Section 2.6.1.3. This simulation shows that the dynamic model generator of this work was able to predict this decay of YFP production when moving through the two states, observed by experimental results and which was not predicted by the steady-state modeling of Cello. It also predicts another YFP decay not predicted or observed experimentally in the Cello project, because the YFP production of the circuit was never measured when removing input molecules from the system. The other YFP decay can be observed when moving from a state where Ara and aTc are present (1/0/1) to a state where there is no inputs (0/0/0), which produces the other decay in YFP production (**Figure 4.4**).

However, Nielsen *et al.* [16] did not offer any explanation as to why this switching dynamics was happening. In the following sections, we use the dynamic model generator of this work to better understand the switching behavior of these genetic circuits.

4.2.1 Circuit Analysis

To demonstrate our analysis procedure, this work analyzes circuit 0x8E (see **Figure 4.6**) from the Cello paper [16], since this paper offers experimental time course data that in-

cluded glitching behavior (**Figure 4.5**). The behavior of this circuit, which is predicted using our automatically generated model, is shown on **Figure 4.4**, in which *Yellow Fluorescent Protein* (YFP) production in *Relative Promoter Units* (RPU) [60] is shown over time for each possible input value. The simulation shows glitches on YFP production in several places including the condition observed experimentally (namely when Ara and IPTG are provided simultaneously).

To better understand the cause of this glitch, let us consider the truth table for this circuit shown in **Table 4.1**. The values in the truth table indicate the steady-state YFP production (1: *High*, 0: *Low*) for different combinations of input molecules Ara, IPTG, and aTc (1: *Present*, 0: *Not Present*). This truth table summarizes the steady-state behavior of the circuit for different input combinations. This truth table can be represented in a more compact form, known as a Karnaugh map.

The Karnaugh map for this circuit, shown in **Table 4.2**, also shows the same steady-states from the truth table for each input combination. The first column and row of this map show all the possible input combinations, and the values indicate the steady-state YFP production (1: *High*, 0: *Low*) for different combinations of input molecules. For example, when (Ara, IPTG, aTc) = (0, 0, 0), then YFP production = 1; and when (Ara, IPTG, aTc) = (1, 1, 1), YFP production = 0. Using the Karnaugh map (**Table 4.2**) and the dynamic simulation (**Figure 4.4**), we can analyze what is producing these glitches.

Let us first consider when the environment changes from no input molecules to where IPTG and aTc are *high*. In the Karnaugh map, this moves the circuit from the states where (Ara, IPTG, aTc) = (0, 0, 0) to (0, 1, 1). Even though circuit 0x8E is experiencing two input changes *simultaneously*, it may *"sense"* one input change before the other, depending on the speed that the effect of this input change propagates through the circuit. Therefore, when there are two input changes to a system occurring *simultaneously*, there are two different paths from the initial-state to the end-state (**Table 4.3** a), depending on which input change the circuit *senses* first. If the circuit senses the aTc change first, the circuit will momentarily pass through state (0, 0, 1) (**Table 4.3** a, green line), where it momentarily evaluates to a *low* output, before reaching the final state (0,1,1), where it also evaluates to *low*. Likewise, if the circuit senses IPTG change first, the circuit momentarily passes through state (0, 1, 0) (**Table 4.3** a, blue line), which evaluates to a *low* output as does the end-state. In both of

these transient states and the final state, the circuit evaluates to *low*, so the circuit makes a monotonic change from *high* to *low*, no matter which input change the circuit senses first. There is no possibility of a glitch, and as the simulation shows (**Figure 4.4**), there is no predicted glitch behavior.

Now, let us consider a transition from no input molecules (0, 0, 0) to where Ara and IPTG are *high* (1, 1, 0). If the circuit senses first the Ara input molecule, it passes through state (1, 0, 0) before reaching state (1, 1, 0), and it evaluates to *high* in all these states (**Table 4.3** b, green line). However, if the circuit senses IPTG before it senses Ara, it will momentarily pass through state (0, 1, 0), which evaluates to a *low* output, before reaching the end-state where the output is *high* (**Table 4.3** b, blue line). This would produce the glitch that is observed both in the simulation (**Figure 4.4**) and the experimental results (**Figure 4.5**) from [16]. This potential for a glitch is known in the asynchronous logic community as a *function hazard* [106]. The existence of a function hazard means that regardless of how the circuit is designed, the possibility of a glitch remains. In other words, a function hazard is a property of the *function* and not of the circuit implementation.

However, it is not always the case that the existence of a function hazard means that a glitch occurs. The transition from no inputs (0, 0, 0) to Ara and aTc *high* (1, 0, 1) also implies two input changes, thus two different paths from the initial-state to the end-state (**Table 4.3** c). In this case, the order that these input changes are sensed also affects the output of the circuit; therefore a function hazard exists. Yet, the glitching behavior is not observed in the simulation shown in **Figure 4.4**. So even if there is a function hazard and a possibility of a glitch behavior, it does not necessarily mean that the glitch will occur.

Once the issue of function hazards and glitches was recognized, we proceeded to identify all the two and three input change function hazards of circuit 0x8E. **Figure 4.7** shows the dynamic simulation of these function hazards using the dynamic model generator of this work. The figure shows many occurrences of glitching behavior, especially for the two-input change hazard simulation. This glitching behavior could not be predicted with the steady-state modeling and was not measured experimentally in the Cello project [16]. In the following section, a more detailed explanation on these function hazards and glitches is provided.

4.2.2 Hazards and Glitches

A *hazard* is the possibility of an unwanted or unexpected variation of the output of a combinational logic network before it reaches steady-state. A *glitch* is the actual observance of such a problem. These terms are used mostly for electronic circuits, though glitches have been observed for *genetic regulatory networks* (GRNs) as well [107].

A glitch is a transient behavior that corrects itself as the system reaches a steady-state. If the output of the circuit is only sampled when the circuit is allowed to reach steady-state, or if only the average output behavior is of importance, then hazards and glitches should not be a major problem. Nonetheless, this glitching behavior can have drastic consequences if this transient output of the GRN causes an irreversible change such as a cascade of responses within or with other cells, if it induces apoptosis, or if it releases a toxic pharmaceutical where/when it should not, etc. Therefore, avoiding glitching behavior can be crucial for safe operation of a genetic circuit.

Glitches manifest when multiple inputs pass through multiple paths with different delays, due to the propagational delays of each path. The circuit diagram for circuit 0x8E, shown in **Figure 4.6**, illustrates the difference in these path ”lengths” from sensing an input molecule, to the production of YFP. The reason we are seeing the glitch is because there are two inputs that are changing: one is going through the ”shorter” path and one is going through the ”longer” path. This delay can cause a faster response for some input changes and longer for others, which produces unwanted switching variations in the output. Therefore, a possible solution to remove the glitch could be to add more delay to the shorter path. This could be done by adding some redundant logic to the circuit (like two successive NOT gates), as shown **Figure 4.8** (a). Using iBioSim and the dynamic model generator, we created the specification and dynamic model for this modified circuit for simulation. **Figure 4.8** (b) and (c) show the simulation results for both two and three input changes for the original and modified circuits. This indicates how adding the redundant logic avoids some glitches but not all of them, and, moreover, the response is generally slower (**Figure 4.8** (b) and (c), red line).

The only way to avoid function hazards is to restrict the allowed input changes to the system. As an example of this, limiting the input changes to only *single* input changes produces a very smooth and glitch-free function, as shown by the simulation in **Figure 4.9**.

Since function hazards occur when there are multiple input changes to a system, restricting the input changes to single-input changes avoids these hazards. Therefore, if the output of a circuit has some intra or intercellular irreversible effects that should be avoided, one might need to restrict the input changes to a system. This does not necessarily mean that only single-input changes are needed to avoid hazards. As shown previously, there are multiple input changes that do not contain function hazards. Therefore, understanding which combination of input changes do not have function hazards would be valuable information for a synthetic biologist.

4.2.3 Important Considerations

A system with a function hazard always has the potential to glitch for the specified input change, and modifying the implementation can only change its likelihood of occurring, but not eliminate the possibility of it occurring. For example, we demonstrated how adding some extra redundant logic to add delay to the short path of the circuit may reduce the likelihood of some glitches while increasing it for others. Function hazards are inherently unavoidable [106] unless one restricts the allowed input changes to the system to include only single-input changes and a restricted set of multiple input changes.

Simulation of a dynamic model allows for the prediction of glitches that cannot be observed with steady-state analysis. The analysis done here was using a ODE numerical solver, which predicts the *average* expected behavior. However, biology is stochastic in nature [19, 22, 25, 62, 63, 66], and it would be necessary to perform stochastic analysis of the system to reach more meaningful results. Additionally, stochastic simulation allows one to calculate the *probability* that a glitch might occur. In the future, we plan to map the RPU units to actual molecule count to enable stochastic analysis for prediction of the probability of glitches.

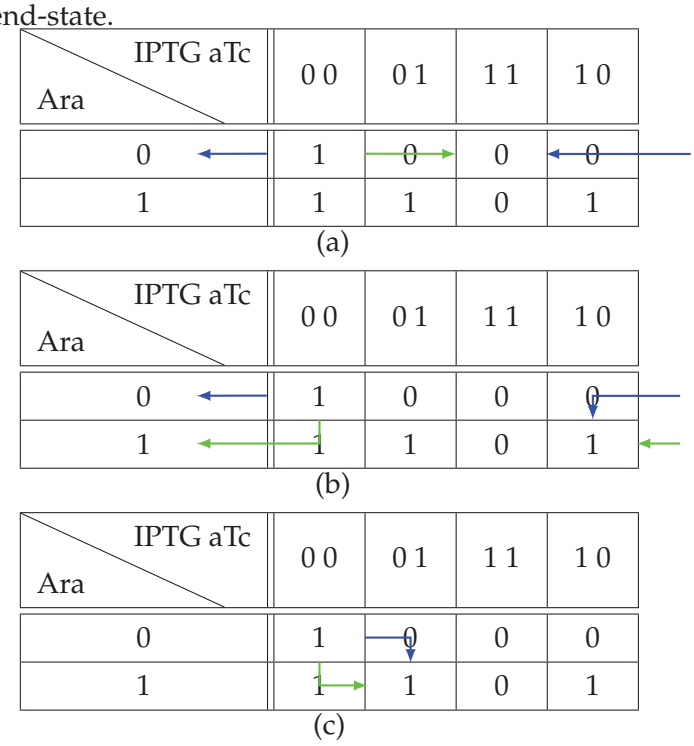
Table 4.1. Truth table for circuit 0x8E. *Low* input/output is represented with a 0, and *high* input/output is represented with a 1. The steady-state YFP production outcome (4th column) is matched to each input combination (columns 1-3).

Ara	IPTG	aTc	YFP
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Table 4.2. Karnaugh map for circuit 0x8E. *Low* input/output is represented with a 0, and *high* input/output is represented with a 1. The header row and first column represent the different combination of input molecules, and the values of the table represent the steady-state YFP production outcome of the circuit.

IPTG aTc / Ara	0 0	0 1	1 1	1 0
0	1	0	0	0
1	1	1	0	1

Table 4.3. Function hazard analysis using Karnaugh map for circuit 0x8E. Green and blue arrows represent the different paths a system can transition when moving from the initial-state to the end-state.



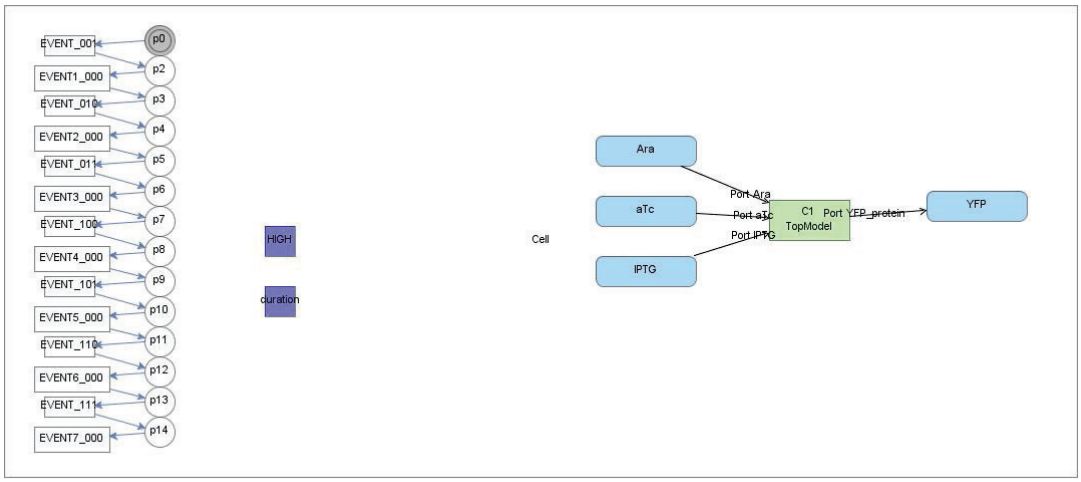


Figure 4.1. Graphical schematic of a simulation environment. This model was created using SBML graphical editor in iBioSim and imported to each project for each genetic circuit simulated. In this model a series of Petri net places (p0, p1, etc.) and transitions (EVENT_000, EVENT_001, etc.) are instantiated to simulate all the input changes as in the Cello project [16]. Global variables *HIGH* and *duration* are also created to control the concentrations of input molecules and duration of the events, respectively. Finally, the top-level module definition (model) of the circuit being simulated is instantiated and all the replacements (arrows) for the top-level input molecules are created.

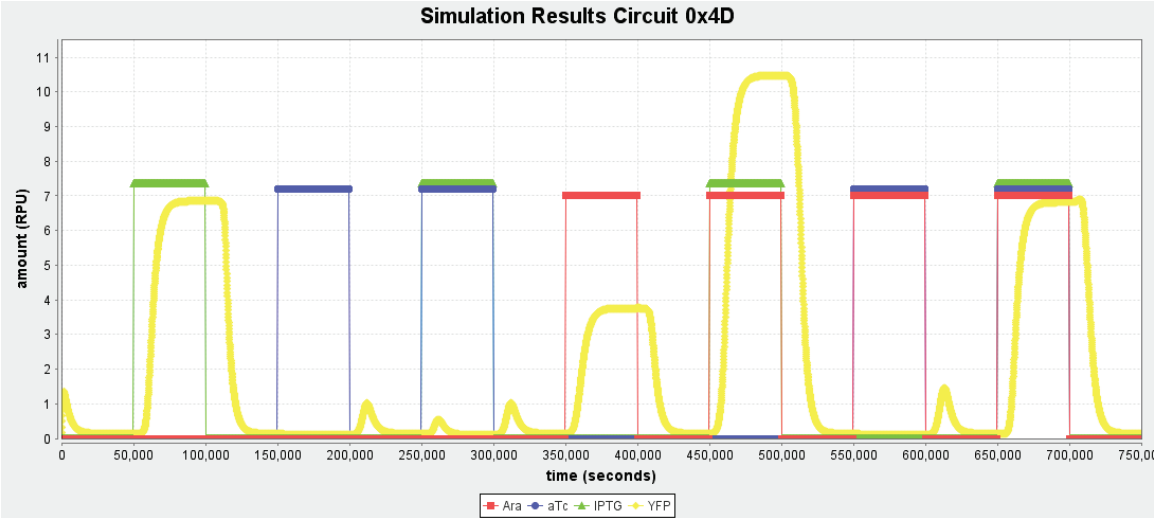


Figure 4.2. Simulation results for circuit 0x4D. This simulation shows the YFP production (in RPU) per unit of time (in seconds) predicted for the circuit using the model generator of this work and the simulation suite of iBioSim [89]. The mathematical model was analyzed using the Runge–Kutta–Fehlberg (4, 5) method and simulated over 750000 seconds. Ara, aTc, and IPTG are the input molecules that the circuit is subjected to.

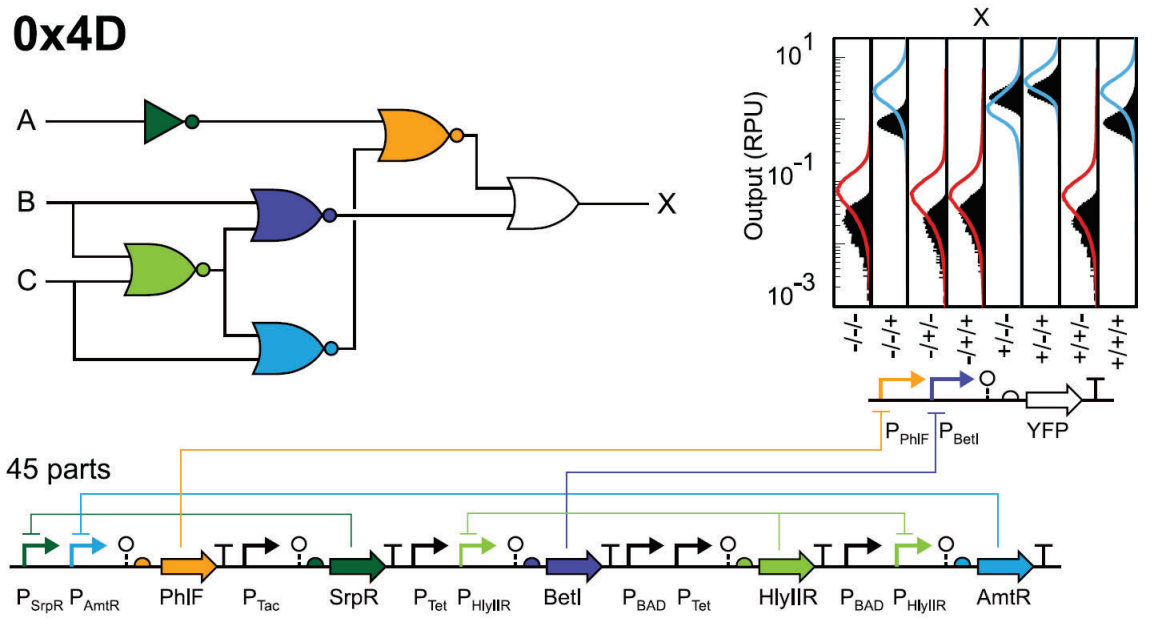


Figure 4.3. Circuit 0x4D from the Cello paper (courtesy of [16]). This image was obtained from [16], and it shows the circuit diagram of Circuit 0x4D, a representation of its genetic parts and interactions, as well as the output predictions (blue and red line distributions) and experimental output measurements (solid black distributions). The inputs A, B, and C correspond to LacI, TetR, and AraC.

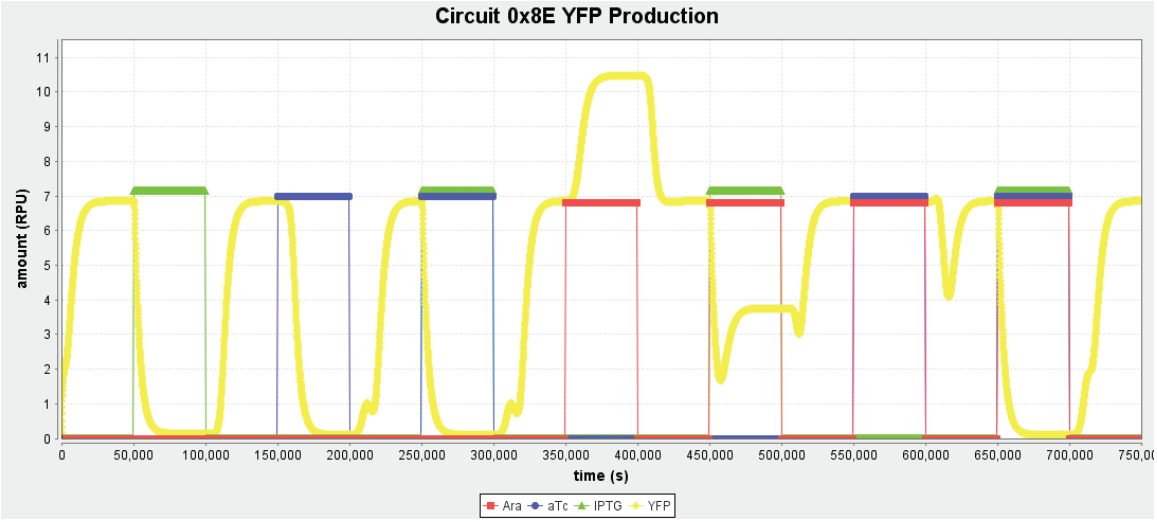


Figure 4.4. YFP production (in *Relative Promoter Units*) over time (in *seconds*) for circuit 0x8E for each combination of input molecules (IPTG, aTc, Ara). Simulation obtained using the automatic model generator of this work, in iBioSim [98].

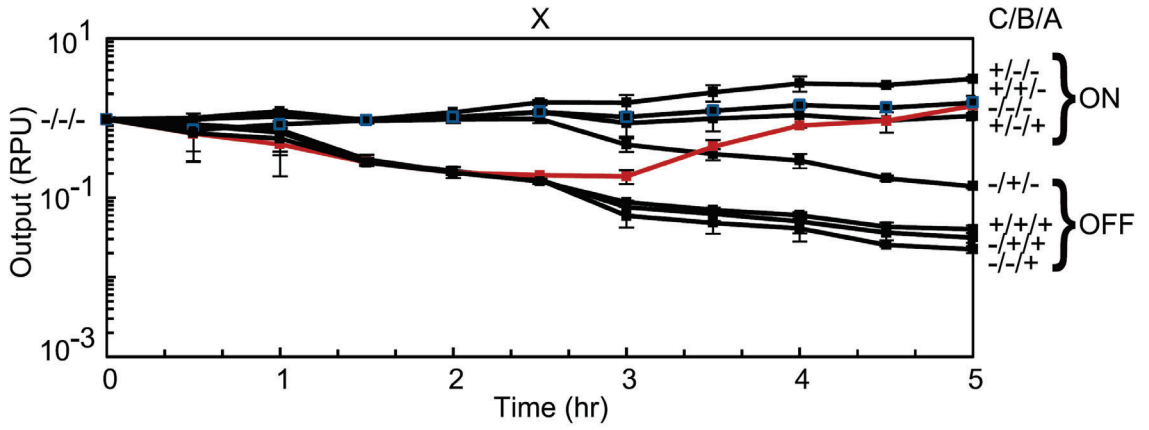


Figure 4.5. Time-course data for circuit 0x8E (courtesy of [16]). Each line represents the output YFP production (in RPUs) over time (in hours) for the circuit 0x8E for each combination of input molecules. This circuit senses three input molecules: Arabinose (Ara), anhydrotetracycline (aTc), and $Isopropyl\beta - D - 1 - thiogalactopyranoside$ (IPTG). In the image, +/+/+ (Ara/aTc/IPTG) represents all input molecules that are present and -/-/- represents no input molecule present. Also the *ON* and *OFF* states represent the predicted outcome at steady state. All outputs behave as expected, except for the +/-/+ state, which experiences an undesirable decay before rising to the *ON* state (red line).

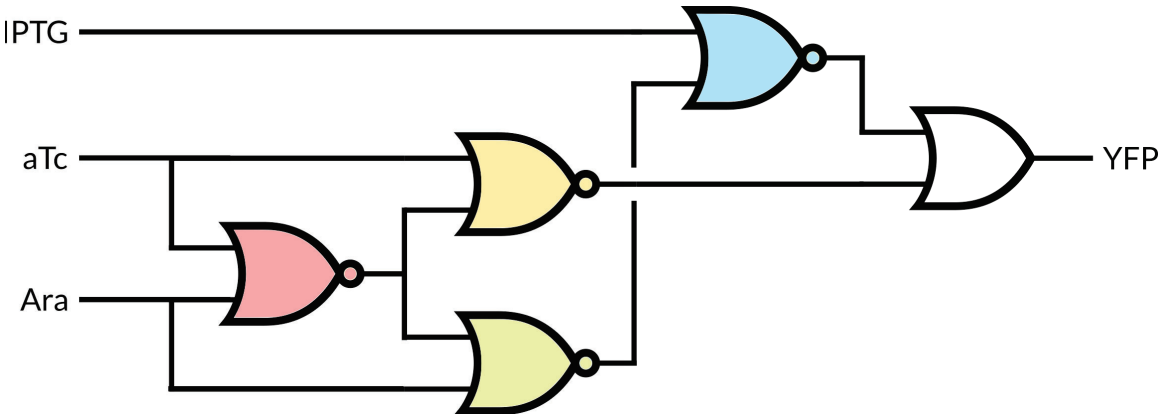
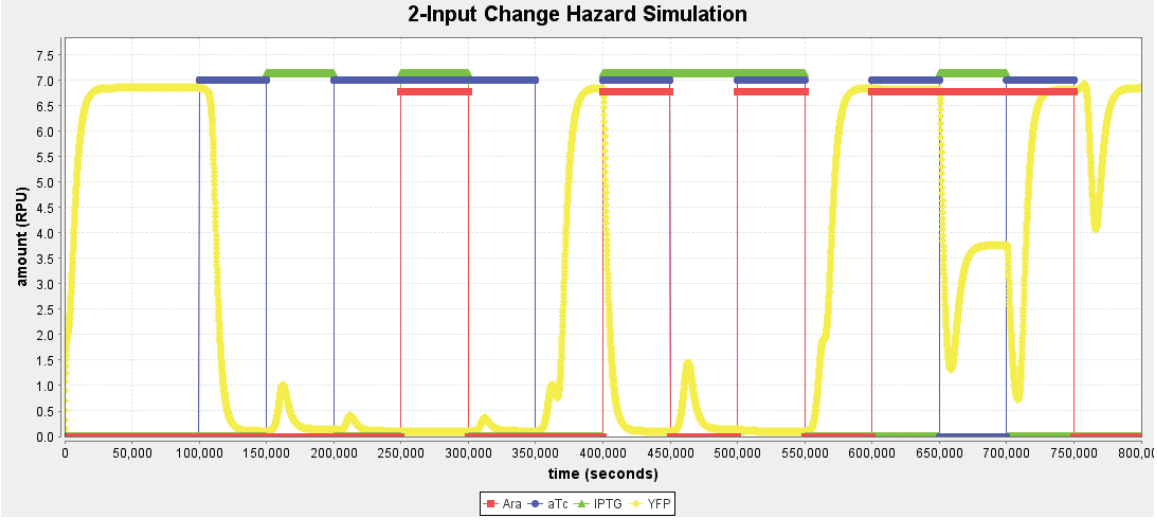
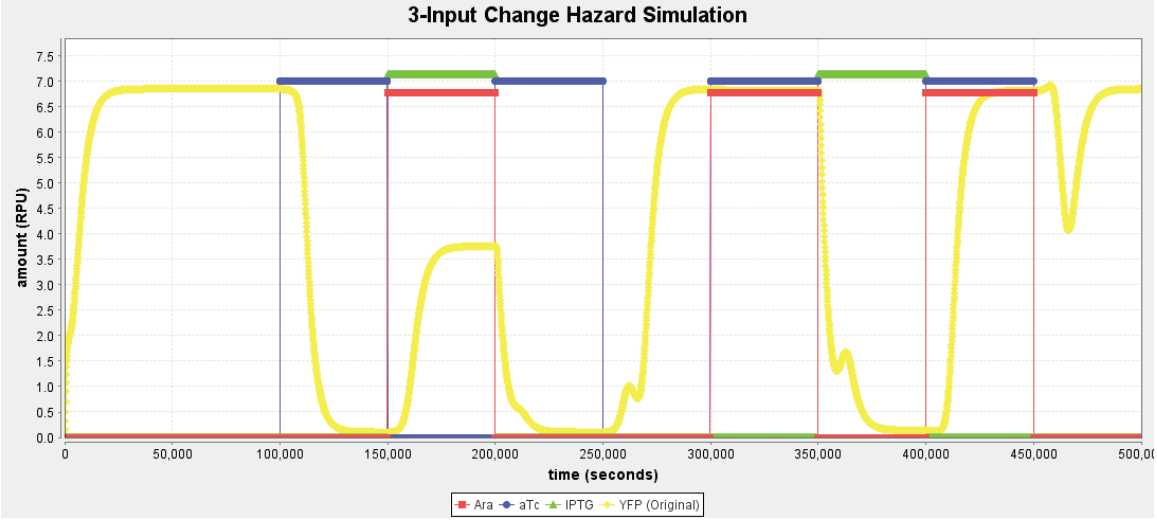


Figure 4.6. Circuit diagram for circuit 0x8E from the Cello paper [16]. In this image, ⊽ represents a NOR gate and ⊃ represents a OR gate. *IPTG*, *aTc*, and *Ara* are the three input molecules for this circuit.

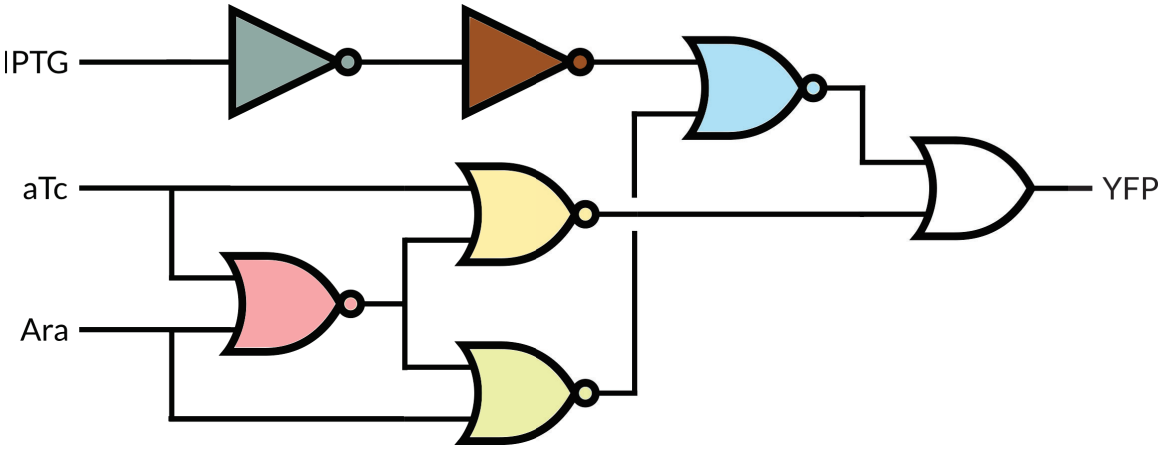


(a)

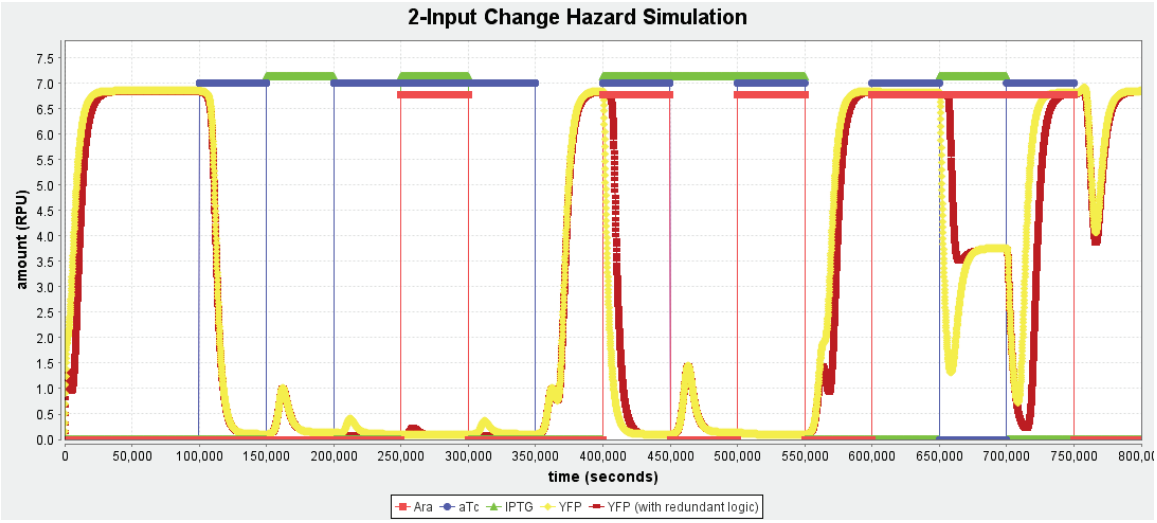


(b)

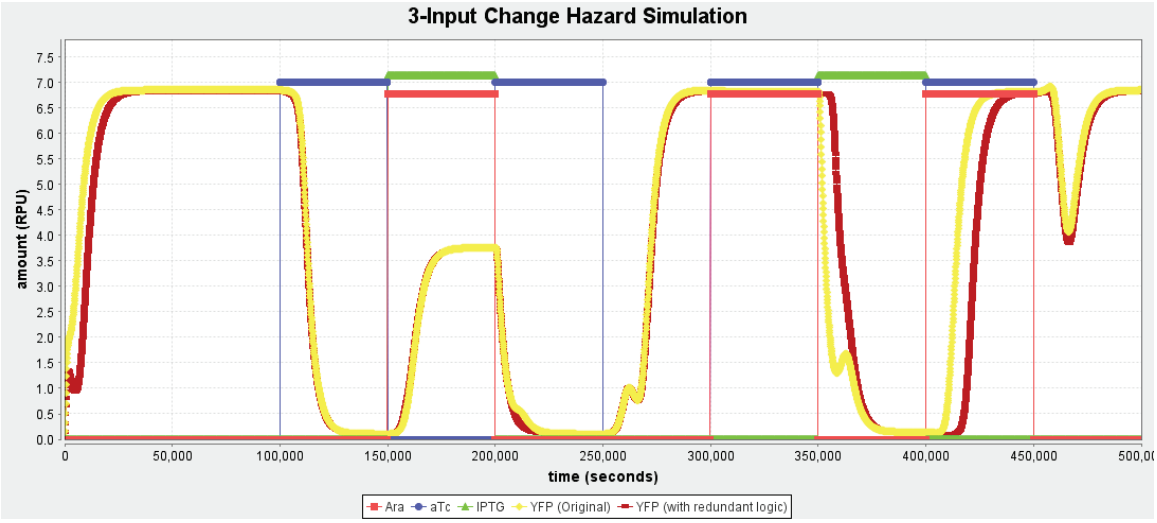
Figure 4.7. Two and three input change hazard simulation. (a) 2-input change hazard simulation for circuit 0x8E. (b) 3-input change hazard simulation for circuit 0x8E.



(a)



(b)



(c)

Figure 4.8. Two and three input-changes hazard simulation for circuit 0x8E with redundant logic. (a) Circuit diagram for circuit 0x8E with redundant logic. (b) Simulation for all 2-input change function hazards of the circuit. (c) Simulation for all 3-input change function hazards of the circuit.

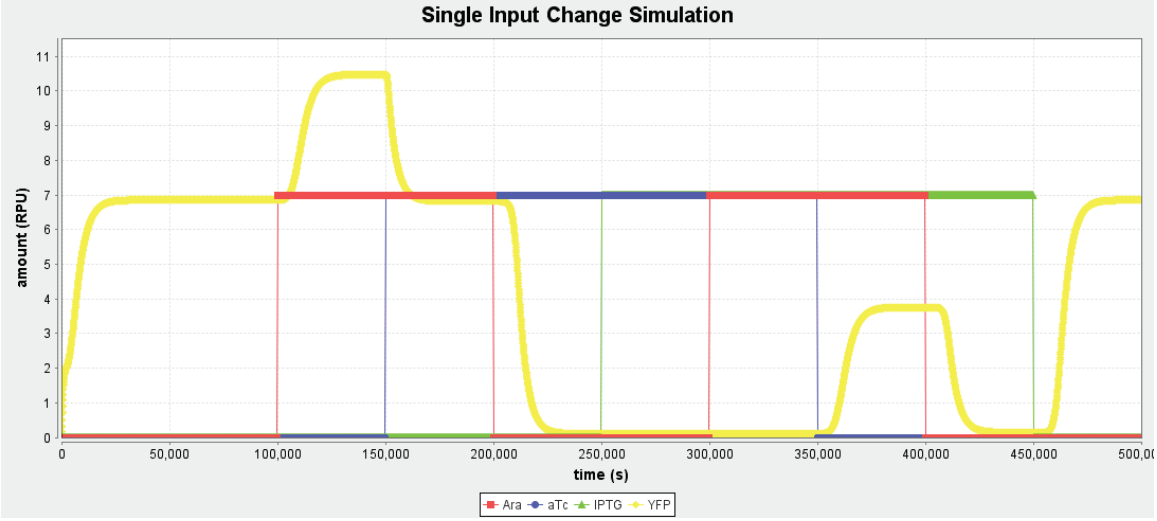


Figure 4.9. Single-input change simulation for circuit 0x8E. For this simulation, the circuit is subjected to different states while only changing one input molecule at a time.

CHAPTER 5

CONCLUSIONS

Synthetic biology witnessed a surge of development over the past two decades [2, 101]. Nonetheless, genetic circuit complexity has not developed on par with genetic engineering technologies [21, 108]. This is not only due to the inherent complexity of biological systems and highly interconnected genetic parts, but also due to a lack of software that helps to cope with this intricacy.

Model-driven design is of paramount importance when designing ever more complex genetic circuits. Modeling is instrumental to show faults in the genetic design, our understanding of underlying biological processes, and the dynamical transition stages of a genetic circuit and potential glitches in the system. However, devising a model for genetic circuits can be a tedious and complex endeavor. Additionally, parametrization is usually lacking for different models, thus making a model inaccurate.

In this work, we have successfully implemented a model generator that automatically creates a dynamic model that can obtain parameters from a repository to produce a two-step model of the transcription and translation processes. This produces mRNA and protein predictions that can be used to analyze the dynamic behavior of a circuit. The dynamic behavior of a circuit can be used to filter faulty circuits, as well as compare with experimental data to further debug the design or understand underlying biological phenomena. This model generator has shown how dynamic modeling can expose unwanted transition states before the circuit reaches stable-states or unexpected time to reach said stable-state, which is not available for steady-state models.

We expect that this new model generation method will bridge the gap between experimentalists and designers as it will help both sides with the results obtained. Designers can use data to better fit the model to produce more accurate predictions, and experimentalists can use these predictions to debug their genetic circuits and predict the behavior of circuits

before constructing them, saving much time and effort. Moreover, the automated model generator has great potential to be further developed to dynamically produce genetic circuits in a wide range of environments, and with more automated processes that are described in detail in the next sections. However, to exploit the potential of this model generator and expand its usability, we propose a workflow and standards to be adopted by the synthetic biology community, which will not only aid model-driven design but also experimentalists when constructing large and complex genetic systems.

5.1 Proposed Workflow

Synthetic biology projects typically rely on iterative workflows composed of different tasks [83]. As mentioned in Section 1.4, this work is anticipated to be part of a larger workflow in the design/build/test pipeline, shown in greater detail in **Figure 5.1**. This expanded Design/Build/Test workflow into a Design/Model/Build/Test/Learn pipeline provides an increased automation and modularization process for designing genetic circuits. There have already been contributions with different software developers that would fit in this workflow, such as Puppeteer [1] and, of course, Cello.[2] Other projects are aiming at parameterizing more genetic gates using Cello parametrization, and debugging a genetic circuit using RNA-seq [13]. The automatic model generator of this work would not only help with model predictions of genetic circuits before the building stage, but also in recognizing circuit failures, function hazards, and glitches of a circuit to either go back to the drawing board, or building the circuit with known restrictions on the circuit implementation. In all, the automated model generation of this work would help filter Cello's copious output designs for function and circuit anomalies using data standards such as SBOL, SBML, and SED-ML for reproducibility and reuse in the community.

5.2 Proposed Standards

Standardization is essential to enable a predictable, top-down engineering discipline [50]. Efforts to promote data standardization have produced widely-used community standards (for a more detailed explanation, refer to Section 2.4). However, the community

[1] http://cidarlab.org/puppeteer/

[2] https://github.com/CIDARLAB/cello

has yet to develop standards for most classes of *biological* parts such as gate composition, gate functions, experimental measurements, and system operation [3]. Therefore, this work uses, and promotes, the use of gate composition and characterization used by the Cello project [16].

5.2.1 Cello Gates

The modular design used in the Cello project [16] (Section 2.6.1.1), if adopted by other research groups, would help in creating a richer and more diverse library of genetic parts to use with the automated model generator of this work. This can help expand the variety of genetic circuits designed, but also open a new standard for other CAD tools to implement when designing genetic circuits. The gate composition used in the Cello project also provides a simpler, more abstract way to parameterize gates and simplify experiments to do so, as it requires less parameters than other characterizations.

5.2.2 Cello Parametrization

Limited availability of reaction rate constants and other parameters are a major hindrance for the development of accurate models, and therefore, for model-driven design [22]. In this respect, lower model resolution or abstraction of different parameters into more generalized parameters is an advantage as it requires less characterization in the laboratory and less detailed understanding of the regulatory mechanisms that underlie a GRN [3, 16, 22, 64, 88]. Cello parametrization, as well as gate composition, provides an abstracted design to reduce part characterization and facilitate modeling. Therefore, if it would be accepted as a standard for genetic gates, part reusability across different laboratories would grow and cooperation would then be facilitated.

5.3 Future Work

The automatic modeler presented in this work has opened lines of communication with researchers from different domains of the synthetic biology community. There are plans in motion to further develop, and integrate, the automatic model generator of this work into genetic design workflows of automatic genetic design. One can only hypothesize what would the evolution of this tool be in the future. Nonetheless, in the following sections, several lines of potential research objectives are described that would promote

further expansion of the automatic model generator developed in this work.

5.3.1 Experimental Data

Hypothesis or model driven genetic design is dependent upon experimental data [61, 62, 109]. Experimental data provides parameter values and validation data, and both modeling and experiments can profit from each other in an iterative learning process.

In the scope of this thesis work, more detailed characterization of genetic parts allows for more accurate and precise model predictions. The automated model generator of this work uses parameter values for $k_{mRNA_{i_{dim}}}$ and $k_{TF_{i_{dim}}}$ taken from literature. Characterization of these two parameter values for each mRNA and protein (TF) from the library of genetic parts would greatly improve the model predictive capabilities and produce more accurate results. Furthermore, high-throughput characterization using RNAseq data and the model predictions of the mathematical model can help with the debugging and comparison of genetic circuits performance and underlying biological phenomena (like gate-toxicity) [13].

Additionally, characterization of new genetic parts not only allows for modeling of genetic circuits in other organisms besides *Escherichia coli* (*E. coli*), but also for different contexts (like genomic vs. plasmid genetic circuits). In the following sections, a more detailed explanation of possible characterization efforts that can be employed to expand the automatic model generator of this work follows.

5.3.1.1 Estimation of Parameters

Systems biology has developed new *-omics* tools that offer the potential to take a "snapshot" of the inner workings of a circuit in a single experiment [13]. One very popular transcriptomic methods of these, *RNA sequencing* (RNA-seq), enables one to quantify the mRNA levels of each gate of a circuit with nucleotide resolution [110]. This high-throughput experiment allows for a complete analysis of a genetic circuit and its impact on the organism, the transfer functions of each genetic gate or part when not in isolation, maximum and minimum levels of transcription, and many other transcriptome analyses [111].

In this thesis work, genetic gates characterized in *E. coli* were used. However, the development of a model that allows for the estimation of biochemical parameters from

RNA-seq, Ribo-seq and proteomics profiling data could be implemented that would allow to use the automated model generator of this work with genetic gates characterized in different environments. This means that the user could feed the automated model generator RNA-seq and Ribo-seq data, and the model generator could estimate the Cello parametrization of each gate, and therefore automatically generate a dynamic model for the circuit with the correct parameter values. The following sections describe in more details how this automated estimation of parameters can help extend the use of the automated model generator of this work.

5.3.1.2 Adjusting Parameters

The ideal modular, orthogonal genetic part, or gate would have the same response function in different genetic and biochemical contexts. However, this is rarely true for most genetic parts. Most genetic gates, when composed into a genetic circuit, have divergent behavior from when they are characterized in isolation. Furthermore, each organism has a unique biochemical and genetic environment that greatly influences the dynamic behavior of genetic gates.

The automated estimation of parameters using RNA-seq and Ribo-seq could be used to estimate the Cello parameters of each gate when combined as a genetic circuit for a particular environment. This can help to calibrate already-known parameters of genetic gates when in different genetic or biochemical environments and account for the context-dependent variability of them. Furthermore, it can also be used to estimate the $k_{mRNA_{i_{dim}}}$ and $k_{TF_{i_{dim}}}$ values for each mRNA and TF of a system, which are missing from the Cello genetic gate library, as well as adjust y_{max}, y_{min}, n, and κ for each gate.

Also, if we want to have a predicted behavior of a circuit for an organism we do not have characterization for, we can adjust the model using Ribo and RNA-seq data of the most similar organism available, and have a better estimate of how a circuit would work on the novel chassis.

5.3.1.3 Parameters for Novel Chassis

The automatic model generator of this work was used with gates characterized in *E. coli*. However, if parametrization were to exist for other organisms that can assimilate the genetic gates used in this work or any homologues of the TetR-family repressors, the auto-

matic model generator could be used to dynamically model their behavior. This automated model generator does not necessarily need to work only with TetR-family homologues. As long as a genetic circuit uses promoters that can be repressed or activated, gates follow the gate composition proposed in the Cello project, and characterization follows the proposed Cello parametrization, the automated model generator can generate a dynamic model for it. Therefore, the only missing information needed for this automatic model generator to work in many other organisms other than *E. Coli* is the same parametrization scheme, since gates that follow these conditions already exist. Whether these parameters are obtained only through experimentation, or through the automated estimation of parameters described in previous sections, is irrelevant.

There are already TetR-homologues that work in yeast [46, 87, 112] and genetic gates using these promoters and that follow the gate composition of the Cello project have been created. However, parametrization as the one used in this work and the Cello project is still underway. Once these parameter values are available, modeling, simulation, and analysis of the results like the ones presented here could be done for genetic circuits in yeast.

5.3.1.4 Plasmid Versus Genomic Parameters

As the synthetic biology field progresses from a proof-of-concept principle to practical applications, it is important to develop single-copy synthetic genetic circuits that are integrated to the genome of the host organism. This would minimize cell-resource consumption and reduce performance variation from the synthetic circuits that are introduced to a host organism.

The automated model generator can generate dynamic models for these circuits if the parameter values for these are available. There are experiments currently underway for the characterization of genomic-integrated genetic gates using the proposed Cello parametrization, which would allow the dynamic model generator of this work to generate dynamic models of them. The dynamic model generator of this work could then be used to generate dynamic models for a same circuit when it is plasmid or genomic integrated and analyze the differences between them. The differences found could help elucidate underlying biological phenomena that differentiates performance between them like gate toxicity, resource allocation, and many other effects. The automated model generator could also have

the option for the user to specify if they want the synthetic genetic circuit they designed to be modeled as a plasmid or genomic integrated circuit.

5.3.1.5 CRISPR/dCas9 Circuits

Recently, programmable and orthogonal CRISPR/dCas9 transcription factors have been developed [113–117]. These transcription factors can be used to build genetic circuits with dCas9-mediated repression. The CRISPR/dCas9 transcription factor system can yield orthogonal genetic gates with low variability and that show minimal retroactivity or effects on cell growth [113]. This makes the gates relatively easy to combine into Boolean logic circuits since the different guide-RNA's have very high specificity, and therefore there is no gate interference and/or leakage. Genetic circuits constructed this way are amongst the largest ever built in any organism [113].

The automatic model generator of this work can easily be adapted to model CRISPR gates and create automatic models for circuits that implement them. Estimation of the parameters for these gates could be done by analysis of results or the automatic estimation of parameters proposed earlier. This would expand the automated model generator of this work to not only work with Cello composed and parameterized gates, but also CRISPR/dCas9 mediated repression gates, and have another method to generate models and simulations for circuits in yeast.

5.3.2 Stochastic Analysis

The deterministic framework of ODE analysis is appropriate to describe the mean behavior of a system, with underlying assumptions such that the variables vary in a deterministic and continuous fashion. In other words, there is no randomness or stochasticity associated with the model, and the same results are obtained given the same initial conditions [118]. However, the stochastic nature of biochemical reactions, even at the single-gene level [119], generates significant intrinsic genetic noise to a system [82]. Furthermore, the underlying assumption of continuous-deterministic models that the number of molecules is high (or that the volume of the system is infinite) can be invalid for biochemical systems where there are very few transcription factors or only one copy DNA segment [21]. Since transcription factors, enzymes, and DNA copies can exist in systems at a low concentration such as a single molecule per cell, any realistic analysis of these

systems must include stochastic effects, and therefore stochastic modeling and analysis [14].

In order to capture the stochastic behavior, a *stochastic chemical kinetics* approach must be taken. With this approach, reactions are assigned propensities of occurring, rather than a rate of reaction, and molecule numbers can be estimated [6, 19]. Simulation requires a Monte Carlo approach, such as Gillespie's *stochastic simulation algorithm* (SSA), which is already implemented in iBioSim [89]. In a SSA simulation, each simulation step selects a random time for the next reaction and a reaction to perform, and it repeats this process until a preselected time limit is reached [6], and, therefore, each simulation is unique. This generates different simulation for the same initial conditions every time the simulation is done, and propensities and probabilities of certain states or dynamic behaviors occurring can be calculated.

Therefore, to be able to produce a stochastic model and analysis on circuits like the ones modeled and analyzed on this thesis work, a translation from RPU units to molecule count has to be developed. This translation would map RPU units to actual molecule count in a cell, and with it, the propensity of each reaction rate. Both of these mappings are needed to enable stochastic modeling and simulation. A stochastic analysis would not only allow for more precise models, but also for other types of analysis, which are described in the next sections.

5.3.2.1 Parametric Sensitivity Analysis

A stochastic model would also allow to implement a sensitivity analysis of the kinetic model [120–124], which would allow one to systematically study the dependence of the quantities of interest on the parameters that define the model and the initial conditions in which it is simulated. A *parametric sensitivity analysis* (PSA) analyzes how changes in the output of a model can be appropriated to different input parameters or variables with a wide range of applications for systems biology. The reasons of performing a PSA vary from the determination of which parameters require additional research at the stage of model calibration and identification, to analysis of the robustness of a circuit and the model results, to a reduction or abstraction of the model via the identification of the parameters that are not relevant for its dynamics [121]. This is useful not only during the design pro-

cess of a circuit, but also on the iterative build/learning process workflow once a genetic circuit is designed. It can help researches learn which genetic gates need a bigger difference in the ON/OFF promoter activity, or which gates need more isolation/stability from the environment, to identify model predictions that are inconsistent with experimental data, suggesting novel experimentation to either validate or falsify a model and many other uses [125].

The automated model generator of this work could be adapted to have the option to perform a PSA on the automatically generated stochastic model.

5.3.2.2 Glitch Propensity

An automatically generated stochastic model can also allow one to calculate the *probability* of certain states or dynamic behavior, running the analysis multiple times and computing the probability from the results.

In this thesis work, we analyzed function hazards and observed glitch behavior predicted using the deterministic-continuous approach of an ODE model analysis. This represents the average behavior expected for a system. However, with analysis of a stochastic model of the same circuits, we could compute the *probabilities* of each possible glitch behavior and have the automated model generator report these probabilities. In this way, anyone using this model generator can have an idea of the risks of this unwanted switching behavior happening and determine if they need to restrict the allowed input changes to the circuit to make certain that the glitching behavior does not occur.

5.3.3 Circuit Hazard Identification

As of now, to analyze a circuit's function hazards and glitch behavior using the dynamic model generator of this work is not an automatic process, and a certain grasp of the topic is needed in order to perform this analysis. To make this analysis available to any user without the need of any understanding on how to do so, it would prove very useful to implement an algorithm that can automatically detect function hazards on a circuit and report them back to the user. This report can also include a list of input changes that may cause a glitching behavior. If a stochastic model of the circuit is available (Sections 5.3.2 and 5.3.2.2), the probability of a glitch can also be reported. This automated analysis can be integrated in the automated model generator of this work, or as a stand-alone application

that can be exported to other software tools, like Cello [16].

5.3.4 Complexity Score

Circuit complexity is measured by the size and depth of a circuit. Complexity scores for electronic circuits essentially depend only on number of gates and can be easily calculated from the truth table or Karnaugh map of a circuit. However, deriving a complexity score for synthetic biology is not as simple since firstly, genetic circuits for a given truth table differ in both gene numbers and quantity of regulation factors and levels, and secondly, circuit performance is not guaranteed due to nonlinear interactions present when the genetic circuit is introduced into a host cell [126].

Larger genetic circuits have produce a larger metabolic burden or load on the host cell and are harder/more costly to build [111]. Therefore, a complexity score may help a designer to set an upper limit on the complexity of a circuit if the known limit for the organism intended is known.

Deriving or adopting a complexity score suitable for genetic circuits and developing an algorithm that can calculate such a score would be indeed be a valuable asset for the automated model generator.

5.3.5 Circuit Performance

Circuit performance can be thought of as the capacity of a circuit to reproduce faithfully or successfully the truth table. There are two essential factors that determine the performance of a circuit, which are the extent that the *high* and *low* output signal of each genetic gate can be practically distinguished, and the transient dynamics after changes in the inputs that may produce incorrect results [126, 127].

The automated model generator of this work could be expanded to calculate an predicted circuit performance. This metric could be based solely on a statistical analysis of the difference between the predicted steady-state output and experimental results. Marchisio *et al.* [126, 127] propose the difference of the minimal and maximal output at steady-state for each gate as a main parameter to quantify gate and circuit performance. Whichever method is preferred, a report (along with complexity score) could be reported as soon as the automated model generator produces the dynamic model, so as to help the user in the design process of a genetic circuit.

5.3.6 Consortium Simulation

Microbial consortia can perform more complicated tasks and endure more changeable environments than monocultures can [128]. Moreover, if a genetic circuit is too large for a single organism to handle, the circuit can be subdivided into smaller genetic circuits for each strain of the consortium, since the division of labor would make the burden of each circuit smaller. This makes synthetic biology for microbial consortium a new and important frontier for synthetic biology.

As biological research on bacteria consortium progresses, a modeling scheme has to be developed to accompany this progress. Mathematical models that describe a consortium of cells are becoming a necessary tool to drive, rather than supplement, design of these systems. From an engineering standpoint, the special considerations for modeling bacterial consortium has to be taken into account when building mathematical models that describe the interactions not only within each cell, but also between the different cells that compose the consortium. This will especially provide insights on unknown interactions (such as resource competition), when the predictions of the model can be compared to experimental results, and the information revealed can help genetic circuit designers to achieve desired symbiotic behaviors.

There are several attempts in the systems biology research discipline to develop models that describe the dynamic behavior of gut microbia in [129–133], to cite a few. However, modeling and analysis of well-defined synthetic genetic networks are more amenable to detailed mathematical modeling and analysis [134]. A possible future course for this work would be to develop or implement an existing mathematical model for synthetic symbiotic ecosystems into the automated model generator of this work, which would enable to automatically create models for these systems without extensive knowledge on the topic.

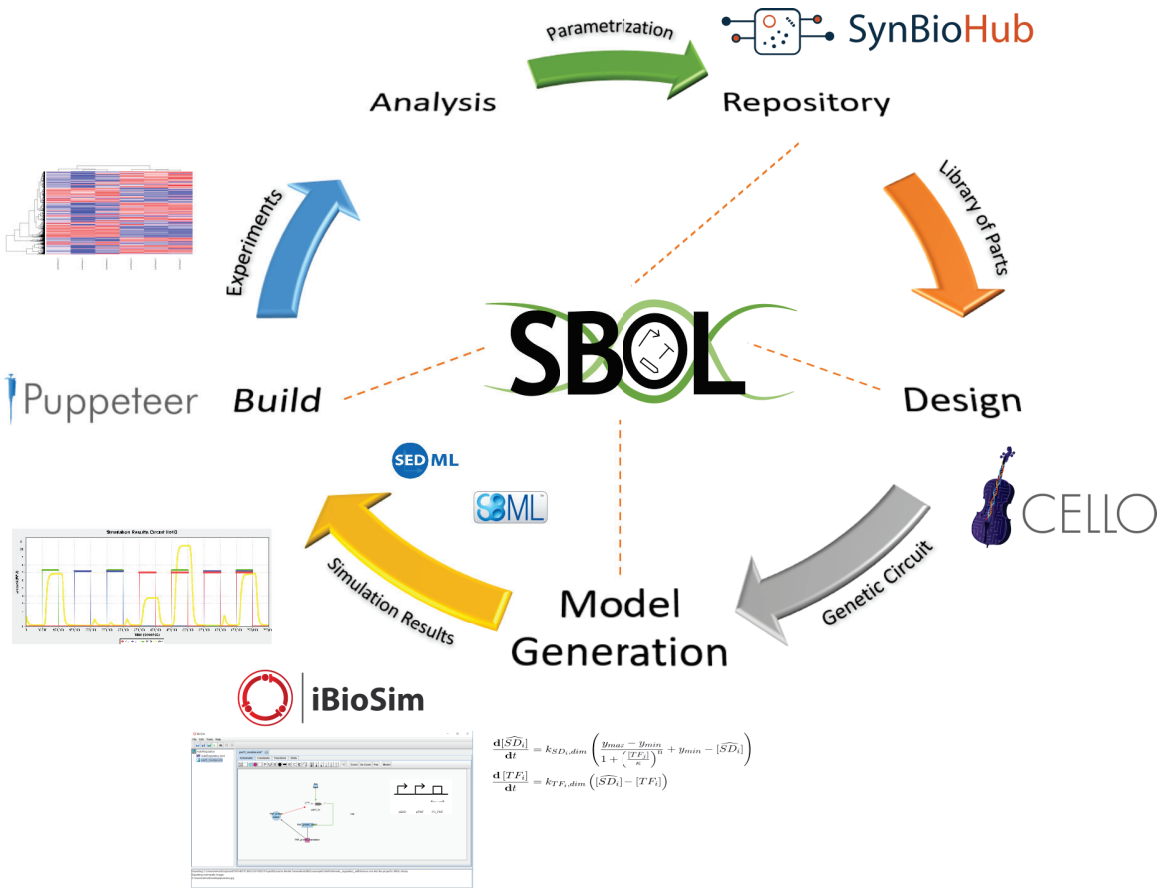


Figure 5.1. Design/Model/Build/Test/Learn workflow. An expanded version of the Design Build Test pipeline [39] and a proposed workflow for model-based design of genetic circuits.

REFERENCES

[1] R. S. Cox, C. Madsen, J. McLaughlin, T. Nguyen, N. Roehner, B. Bartley, S. Bhatia, M. Bissell, K. Clancy, T. Gorochowski, R. Grünberg, A. Luna, N. N. Le, M. Pocock, H. Sauro, J. T. Sexton, G.-B. Stan, J. J. Tabor, C. A. Voigt, Z. Zundel, C. Myers, J. Beal, and A. Wipat, "Synthetic Biology Open Language Visual (SBOL Visual) Version 2.0," *J. Integr. Bioinforma.*, vol. 15, no. 1, 2018.

[2] E. Lukas, R. Nicolas, J. Nick, C. Vijayalakshmi, L. Camille, L. Chen, and L. N. Nicolas, "Designing and encoding models for synthetic biology," *J. R. Soc. Interface*, vol. 6, no. suppl_4, pp. S405–S417, Aug. 2009.

[3] D. Endy, "Foundations for engineering biology," *Nature*, vol. 438, no. 7067, pp. 449–453, Nov. 2005.

[4] T. Knight, "DARPA BioComp plasmid distribution 1.00 of standard Biobrick components," Tech. Rep., Massachusetts inst. of tech Cambridge, Artificial Intelligence lab, May 2002.

[5] S. Rollié, M. Mangold, and K. Sundmacher, "Designing biological systems: Systems Engineering meets Synthetic Biology," *Chem. Eng. Sci.*, vol. 69, no. 1, pp. 1–29, Feb. 2012.

[6] C. J. Myers, "Computational synthetic biology: Progress and the road ahead," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 1, no. 1, pp. 19–32, Jan.-March 2015.

[7] N. Le Novère, M. Hucka, N. Anwar, G. D. Bader, E. Demir, S. Moodie, and A. Sorokin, "Meeting report from the first meetings of the Computational Modeling in Biology Network (COMBINE)," *Stand. Genomic. Sci.*, vol. 5, no. 2, pp. 230–242, Nov. 2011.

[8] C. M. Agapakis, "Designing synthetic biology," *ACS Synth. Biol.*, vol. 3, no. 3, pp. 121–128, Mar. 2014.

[9] T. S. Gardner, C. R. Cantor, and J. J. Collins, "Construction of a genetic toggle switch in *Escherichia coli*," *Nature*, vol. 403, no. 6767, pp. 339–342, Jan. 2000.

[10] M. B. Elowitz and S. Leibler, "A synthetic oscillatory network of transcriptional regulators," *Nature*, vol. 403, no. 6767, pp. 335–338, Jan. 2000.

[11] E. M. Judd, M. T. Laub, and H. H. McAdams, "Toggles and oscillators: New genetic circuit designs," *BioEssays*, vol. 22, no. 6, pp. 507–509, June 2000.

[12] S. Mukherji and A. van Oudenaarden, "Synthetic biology: Understanding biological design from synthetic circuits," *Nat. Rev. Genet.*, vol. 10, no. 12, pp. 859–871, Dec. 2009.

[13] T. E. Gorochowski, A. E. Borujeni, Y. Park, A. A. Nielsen, J. Zhang, B. S. Der, D. B. Gordon, and C. A. Voigt, "Genetic circuit characterization and debugging using RNA-seq," *Mol. Syst. Biol.*, vol. 13, no. 11, pp. 952, Nov. 2017.

[14] N. Crook and H. S. Alper, "Model-based design of synthetic, biological systems," *Chem. Eng. Sci.*, vol. 103, pp. 2–11, Nov. 2013.

[15] M. Heinemann and S. Panke, "Synthetic biology—putting engineering into biology," *Bioinformatics*, vol. 22, no. 22, pp. 2790–2799, Nov. 2006.

[16] A. A. K. Nielsen, B. S. Der, J. Shin, P. Vaidyanathan, V. Paralanov, E. A. Strychalski, D. Ross, D. Densmore, and C. A. Voigt, "Genetic circuit design automation," *Science*, vol. 352, no. 6281, pp. aac7341, Apr. 2016.

[17] J. Peccoud, "Synthetic biology: Fostering the cyber-biological revolution," *Synth. Biol.*, vol. 1, no. 1, Jan. 2016.

[18] N. Davidsohn, J. Beal, S. Kiani, A. Adler, F. Yaman, Y. Li, Z. Xie, and R. Weiss, "Accurate predictions of genetic circuit behavior from part characterization and modular composition," *ACS Synth. Biol.*, vol. 4, no. 6, pp. 673–681, June 2015.

[19] C. J. Myers, *Engineering Genetic Circuits*. Salt Lake City, Utah, USA: Chapman and Hall/CRC, 2016.

[20] H. de Jong, "Modeling and simulation of genetic regulatory systems: A literature review," *J. Comp. Biol.*, vol. 9, no. 1, pp. 67–103, Jan. 2002.

[21] Y. N. Kaznessis, "Models for synthetic biology," *BMC Syst. Biol.*, vol. 1, no. 1, pp. 47, Nov. 2007.

[22] G. Karlebach and R. Shamir, "Modelling and analysis of gene regulatory networks," *Nat. Rev. Mol. Cell Biol.*, vol. 9, no. 10, pp. 770–780, Oct. 2008.

[23] T. Schlitt and A. Brazma, "Current approaches to gene regulatory network modelling," *BMC Bioinformatics*, vol. 8, no. 6, pp. S9, Sept. 2007.

[24] J. T. MacDonald, C. Barnes, R. I. Kitney, P. S. Freemont, and G.-B. V. Stan, "Computational design approaches and tools for synthetic biology," *Integr. Biol.*, vol. 3, no. 2, pp. 97–108, 2011.

[25] M. A. Marchisio, *Introduction in Synthetic Biology: About Modeling, Computation, and Circuit Design*. New York, NY: Springer Berlin Heidelberg, 2018.

[26] U. Alon, *An Introduction to Systems Biology: Design Principles of Biological Circuits*. Chapman & Hall/CRC, July 2006.

[27] H. Doosthosseini, "Deterministic Dynamic Model of Synthetic Genetic Circuit Function," Tech. Rep., MIT, Cambridge, MA, USA, June 2018.

[28] D. Waltemath, R. Henkel, F. Winter, and O. Wolkenhauer, "Reproducibility of model-based results in systems biology," in *Systems Biology: Integrative Biology and Simulation Tools*, A. Prokop and B. Csukás, Eds., pp. 301–320. Netherlands: Springer, 2013.

[29] E. Klipp, W. Liebermeister, A. Helbig, A. Kowald, and J. Schaber, "Systems biology standards—the community speaks," *Nat. Biotechnol.*, vol. 25, pp. 390–391, Apr. 2007.

[30] M. Galdzicki, K. P. Clancy, E. Oberortner, M. Pocock, J. Y. Quinn, C. A. Rodriguez, N. Roehner, M. L. Wilson, L. Adam, J. C. Anderson, B. A. Bartley, J. Beal, D. Chandran, J. Chen, D. Densmore, D. Endy, R. Grünberg, J. Hallinan, N. J. Hillson, J. D. Johnson, A. Kuchinsky, M. Lux, G. Misirli, J. Peccoud, H. A. Plahar, E. Sirin, G.-B. Stan, A. Villalobos, A. Wipat, J. H. Gennari, C. J. Myers, and H. M. Sauro, "The Synthetic Biology Open Language (SBOL) provides a community standard for communicating designs in synthetic biology," *Nat. Biotechnol.*, vol. 32, no. 6, pp. 545–550, June 2014.

[31] C. Chaouiya, D. Bérenguier, S. M. Keating, A. Naldi, M. P. van Iersel, N. Rodriguez, A. Dräger, F. Büchel, T. Cokelaer, B. Kowal, B. Wicks, E. Gonçalves, J. Dorier, M. Page, P. T. Monteiro, A. von Kamp, I. Xenarios, H. de Jong, M. Hucka, S. Klamt, D. Thieffry, N. Le Novère, J. Saez-Rodriguez, and T. Helikar, "SBML qualitative models: A model representation format and infrastructure to foster interactions between qualitative modelling formalisms and tools," *BMC Syst. Biol.*, vol. 7, no. 1, pp. 135, Dec. 2013.

[32] M. Hucka, A. Finney, H. M. Sauro, H. Bolouri, J. C. Doyle, H. Kitano, A. P. Arkin, B. J. Bornstein, D. Bray, A. Cornish-Bowden, A. A. Cuellar, S. Dronov, E. D. Gilles, M. Ginkel, V. Gor, I. I. Goryanin, W. J. Hedley, T. C. Hodgman, J.-H. Hofmeyr, P. J. Hunter, N. S. Juty, J. L. Kasberger, A. Kremling, U. Kummer, N. Le Novère, L. M. Loew, D. Lucio, P. Mendes, E. Minch, E. D. Mjolsness, Y. Nakayama, M. R. Nelson, P. F. Nielsen, T. Sakurada, J. C. Schaff, B. E. Shapiro, T. S. Shimizu, H. D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, and J. Wang, "The systems biology markup language (SBML): A medium for representation and exchange of biochemical network models," *Bioinformatics*, vol. 19, no. 4, pp. 524–531, Mar. 2003.

[33] D. Chandran, F. T. Bergmann, and H. M. Sauro, "TinkerCell: Modular CAD tool for synthetic biology," *J. Biol. Eng.*, vol. 3, no. 1, pp. 19, Oct. 2009.

[34] A. Dräger, N. Hassis, J. Supper, A. Schröder, and A. Zell, "SBMLsqueezer: A CellDesigner plug-in to generate kinetic rate equations for biochemical networks," *BMC Syst. Biol.*, vol. 2, no. 1, pp. 39, Apr. 2008.

[35] A. Funahashi, Y. Matsuoka, A. Jouraku, M. Morohashi, N. Kikuchi, and H. Kitano, "CellDesigner 3.5: A versatile modeling tool for biochemical networks," *Proc. IEEE*, vol. 96, no. 8, pp. 1254–1265, Aug. 2008.

[36] S. Hoops, S. Sahle, R. Gauges, C. Lee, J. Pahle, N. Simus, M. Singhal, L. Xu, P. Mendes, and U. Kummer, "COPASI—a COmplex PAthway SImulator," *Bioinformatics*, vol. 22, no. 24, pp. 3067–3074, Dec. 2006.

[37] S. Mirschel, K. Steinmetz, M. Rempel, M. Ginkel, and E. D. Gilles, "ProMoT: Modular modeling for systems biology," *Bioinformatics*, vol. 25, no. 5, pp. 687–689, Mar. 2009.

[38] C. Wrzodek, A. Dräger, and A. Zell, "KEGGtranslator: Visualizing and converting the KEGG PATHWAY database to various formats," *Bioinformatics*, vol. 27, no. 16, pp. 2314–2315, Aug. 2011.

[39] R. T. Gill, A. L. Halweg-Edwards, A. Clauset, and S. F. Way, "Synthesis aided design: The biological design-build-test engineering paradigm?," *Biotechnol. Bioeng.*, vol. 113, no. 1, pp. 7–10, 2016.

[40] W. A. Lim, "Designing customized cell signalling circuits," *Nat. Rev. Mol. Cell Biol.*, vol. 11, no. 6, pp. 393–403, June 2010.

[41] T. K. Lu, A. S. Khalil, and J. J. Collins, "Next-generation synthetic gene networks," *Nat. Biotechnol.*, vol. 27, no. 12, pp. 1139–1150, Dec. 2009.

[42] N. Nandagopal and M. B. Elowitz, "Synthetic biology: Integrated gene circuits," *Science*, vol. 333, no. 6047, pp. 1244–1248, Sept. 2011.

[43] D. Sprinzak and M. B. Elowitz, "Reconstruction of genetic circuits," *Nature*, vol. 438, no. 7067, pp. 443, Nov. 2005.

[44] C. A. Voigt, "Genetic parts to program bacteria," *Curr. Opin. Biotech.*, vol. 17, no. 5, pp. 548–557, Oct. 2006.

[45] J. A. N. Brophy and C. A. Voigt, "Principles of genetic circuit design," *Nat. Methods*, vol. 11, no. 5, pp. 508–520, May 2014.

[46] B. Wang, R. I. Kitney, N. Joly, and M. Buck, "Engineering modular and orthogonal genetic logic gates for robust digital-like synthetic biology," *Nat. Commun.*, vol. 2, pp. 508, Oct. 2011.

[47] S. Cardinale and A. P. Arkin, "Contextualizing context for synthetic biology – identifying causes of failure of synthetic biological systems," *Biotechnol. J.*, vol. 7, no. 7, pp. 856–866, 2012.

[48] C. R. Boehm and R. Bock, "Recent advances and current challenges in synthetic biology of the plastid genetic system and metabolism," *Plant Physiol.*, vol. 179, no. 3, pp. 794–802, Mar. 2019.

[49] D. A. Drubin, J. C. Way, and P. A. Silver, "Designing biological systems," *Genes Dev.*, vol. 21, no. 3, pp. 242–254, Jan. 2007.

[50] T. Decoene, B. D. Paepe, J. Maertens, P. Coussement, G. Peters, S. L. D. Maeseneire, and M. D. Mey, "Standardization in synthetic biology: An engineering discipline coming of age," *Crit. Rev. Biotechnol.*, vol. 38, no. 5, pp. 647–656, July 2018.

[51] B. Canton, A. Labno, and D. Endy, "Refinement and standardization of synthetic biological parts and devices," *Nat. Biotechnol.*, vol. 26, no. 7, pp. 787–793, July 2008.

[52] Y.-J. Chen, P. Liu, A. A. K. Nielsen, J. A. N. Brophy, K. Clancy, T. Peterson, and C. A. Voigt, "Characterization of 582 natural and synthetic terminators and quantification of their design constraints," *Nat. Methods*, vol. 10, no. 7, pp. 659–664, July 2013.

[53] S. Kosuri, D. B. Goodman, G. Cambray, V. K. Mutalik, Y. Gao, A. P. Arkin, D. Endy, and G. M. Church, "Composability of regulatory sequences controlling transcription and translation in Escherichia coli," *PNAS*, vol. 110, no. 34, pp. 14024–14029, Aug. 2013.

[54] V. K. Mutalik, J. C. Guimaraes, G. Cambray, C. Lam, M. J. Christoffersen, Q.-A. Mai, A. B. Tran, M. Paull, J. D. Keasling, A. P. Arkin, and D. Endy, "Precise and reliable gene expression via standard transcription and translation initiation elements," *Nat. Methods*, vol. 10, no. 4, pp. 354–360, Apr. 2013.

[55] V. K. Mutalik, J. C. Guimaraes, G. Cambray, Q.-A. Mai, M. J. Christoffersen, L. Martin, A. Yu, C. Lam, C. Rodriguez, G. Bennett, J. D. Keasling, D. Endy, and A. P. Arkin, "Quantitative estimation of activity and quality for collections of functional genetic elements," *Nat. Methods*, vol. 10, no. 4, pp. 347–353, Apr. 2013.

[56] V. K. Mutalik, L. Qi, J. C. Guimaraes, J. B. Lucks, and A. P. Arkin, "Rationally designed families of orthogonal RNA regulators of translation," *Nat. Chem. Biol.*, vol. 8, no. 5, pp. 447–454, May 2012.

[57] T. Knight, "Idempotent vector design for standard assembly of biobricks," *MIT Synth. Biol. Work. Group*, 2003.

[58] I. Phillips and P. Silver, "A new biobrick assembly strategy designed for facile protein engineering," *MIT SBWG Tech. Rep.*, Apr. 2006.

[59] R. P. Shetty, D. Endy, and T. F. Knight, "Engineering BioBrick vectors from BioBrick parts," *J. Biol. Eng.*, vol. 2, no. 1, pp. 5, Apr. 2008.

[60] J. R. Kelly, A. J. Rubin, J. H. Davis, C. M. Ajo-Franklin, J. Cumbers, M. J. Czar, K. de Mora, A. L. Glieberman, D. D. Monie, and D. Endy, "Measuring the activity of BioBrick promoters using an in vivo reference standard," *J. Biol. Eng.*, vol. 3, no. 1, pp. 4, Mar. 2009.

[61] M. A. Marchisio, Ed., *Computational Methods in Synthetic Biology*, Number 1244 in Methods in Molecular Biology. New York: Humana Press, 2015.

[62] D. Chandran, W. Copeland, S. Sleight, and H. Sauro, "Mathematical modeling and synthetic biology," *J. Biol. Eng.*, vol. 5, no. 4, pp. 299–309, Dec. 2008.

[63] H. Bolouri, *Computational Modelling Of Gene Regulatory Networks – A Primer*. 57 Shelton Street, London: Imperial College Press, Aug. 2008.

[64] J. A. Bernstein, A. B. Khodursky, P.-H. Lin, S. Lin-Chao, and S. N. Cohen, "Global analysis of mRNA decay and abundance in Escherichia coli at single-gene resolution using two-color fluorescent DNA microarrays," *PNAS*, vol. 99, no. 15, pp. 9697–9702, July 2002.

[65] S. Ghaemmaghami, W.-K. Huh, K. Bower, R. W. Howson, A. Belle, N. Dephoure, E. K. O'Shea, and J. S. Weissman, "Global analysis of protein expression in yeast," *Nature*, vol. 425, no. 6959, pp. 737, Oct. 2003.

[66] T. Schlitt, "Approaches to modeling gene regulatory networks: A gentle introduction," *Methods Mol. Biol.*, vol. 1021, pp. 13–35, 2013.

[67] J. J. Tyson, K. C. Chen, and B. Novak, "Sniffers, buzzers, toggles and blinkers: Dynamics of regulatory and signaling pathways in the cell," *Curr. Opin. Cell Biol.*, vol. 15, no. 2, pp. 221–231, Apr. 2003.

[68] M. R. Bennett, D. Volfson, L. Tsimring, and J. Hasty, "Transient dynamics of genetic regulatory networks," *Biophys. J.*, vol. 92, no. 10, pp. 3501–3512, May 2007.

[69] F. Boulier, M. Lefranc, F. Lemaire, and P.-E. Morant, "Applying a rigorous quasi-steady state approximation method for proving the absence of oscillations in models of genetic circuits," in *Algebraic Biology*, K. Horimoto, G. Regensburger, M. Rosenkranz, and H. Yoshida, Eds. 2008, Lecture Notes in Computer Science, pp. 56–64, Berlin Heidelberg: Springer.

[70] A. Polynikis, S. J. Hogan, and M. di Bernardo, "Comparing different ODE modelling approaches for gene regulatory networks," *J. Theor. Biol.*, vol. 261, no. 4, pp. 511–530, Dec. 2009.

[71] C. J. Myers, J. Beal, T. E. Gorochowski, H. Kuwahara, C. Madsen, J. A. McLaughlin, G. Mısırlı, T. Nguyen, E. Oberortner, M. Samineni, A. Wipat, M. Zhang, and Z. Zundel, "A standard-enabled workflow for synthetic biology," *Biochem. Soc. Trans.*, vol. 45, no. 3, pp. 793–803, June 2017.

[72] E. C. Hayden, "Synthetic biology called to order: Meeting launches effort to develop standards for fast-moving field," *Nature*, vol. 520, no. 7546, pp. 141–143, 2015.

[73] W. R. Pearson and D. J. Lipman, "Improved tools for biological sequence comparison," *PNAS*, vol. 85, no. 8, pp. 2444–2448, Apr. 1988.

[74] N. Roehner, J. Beal, K. Clancy, B. Bartley, G. Misirli, R. Grünberg, E. Oberortner, M. Pocock, M. Bissell, C. Madsen, T. Nguyen, M. Zhang, Z. Zhang, Z. Zundel, D. Densmore, J. H. Gennari, A. Wipat, H. M. Sauro, and C. J. Myers, "Sharing structure and function in biological design with SBOL 2.0," *ACS Synth. Biol.*, vol. 5, no. 6, pp. 498–506, June 2016.

[75] V. Chelliah, C. Laibe, and N. L. Novère, "BioModels database: A repository of mathematical models of biological processes," in *Encyclopedia of Systems Biology*, W. Dubitzky, O. Wolkenhauer, K.-H. Cho, and H. Yokota, Eds., pp. 134–138. ew York, NY: Springer, 2013.

[76] B. J. Bornstein, S. M. Keating, A. Jouraku, and M. Hucka, "LibSBML: An API Library for SBML," *Bioinformatics*, vol. 24, no. 6, pp. 880–881, Mar. 2008.

[77] T. Nguyen, N. Roehner, Z. Zundel, and C. J. Myers, "A converter from the systems biology markup language to the synthetic biology open language," *ACS Synth. Biol.*, vol. 5, no. 6, pp. 479–486, June 2016.

[78] N. Roehner, Z. Zhang, T. Nguyen, and C. J. Myers, "Generating systems biology markup language models from the synthetic biology open language," *ACS Synth. Biol.*, vol. 4, no. 8, pp. 873–879, Aug. 2015.

[79] D. Waltemath, R. Adams, F. T. Bergmann, M. Hucka, F. Kolpakov, A. K. Miller, I. I. Moraru, D. Nickerson, S. Sahle, J. L. Snoep, and N. Le Novère, "Reproducible

computational biology experiments with SED-ML - The simulation experiment description markup language," *BMC Syst. Biol.*, vol. 5, no. 1, pp. 198, Dec. 2011.

[80] T. S. Ham, Z. Dmytriv, H. Plahar, J. Chen, N. J. Hillson, and J. D. Keasling, "Design, implementation and practice of JBEI-ICE: An open source biological part registry platform and tools," *Nucleic Acids Res.*, vol. 40, no. 18, pp. e141–e141, Oct. 2012.

[81] J. A. McLaughlin, C. J. Myers, Z. Zundel, G. Mısırlı, M. Zhang, I. D. Ofiteru, A. Goñi-Moreno, and A. Wipat, "SynBioHub: A standards-enabled design repository for synthetic biology," *ACS Synth. Biol.*, vol. 7, no. 2, pp. 682–688, Feb. 2018.

[82] P. E. M. Purnick and R. Weiss, "The second wave of synthetic biology: From modules to systems," *Nat. Rev. Mol. Cell Biol.*, vol. 10, no. 6, pp. 410–422, June 2009.

[83] M. W. Lux, B. W. Bramlett, D. A. Ball, and J. Peccoud, "Genetic design automation: Engineering fantasy or scientific renewal?," *Trends Biochem. Sci.*, vol. 30, no. 2, pp. 120–126, Feb. 2012.

[84] R. Kwok, "Five hard truths for synthetic biology," *Nature*, vol. 463, no. 7279, pp. 288–290, Jan. 2010.

[85] M. A. Marchisio and J. Stelling, "Computational design of synthetic gene circuits with composable parts," *Bioinformatics*, vol. 24, no. 17, pp. 1903–1910, Sept. 2008.

[86] K. Clancy and C. A. Voigt, "Programming cells: Towards an automated 'Genetic Compiler'," *Curr. Opin. Biotech.*, vol. 21, no. 4, pp. 572–581, Aug. 2010.

[87] B. C. Stanton, A. A. K. Nielsen, A. Tamsir, K. Clancy, T. Peterson, and C. A. Voigt, "Genomic mining of prokaryotic repressors for orthogonal logic gates," *Nat. Chem. Biol.*, vol. 10, no. 2, pp. 99–105, Feb. 2014.

[88] A. A. K. Nielsen, B. S. Der, J. Shin, P. Vaidyanathan, V. Paralanov, E. A. Strychalski, D. Ross, D. Densmore, and C. A. Voigt, "Genetic circuit design automation - Supplementary materials," *Science*, 2019.

[89] L. Watanabe, T. Nguyen, M. Zhang, Z. Zundel, Z. Zhang, C. Madsen, N. Roehner, and C. Myers, "iBioSim 3: A tool for model-based genetic circuit design," *ACS Synth. Biol.*, June 2018.

[90] C. J. Myers, N. Barker, K. Jones, H. Kuwahara, C. Madsen, and N.-P. D. Nguyen, "iBioSim: A tool for the analysis and design of genetic circuits," *Bioinformatics*, vol. 25, no. 21, pp. 2848–2849, Nov. 2009.

[91] C. Madsen, C. J. Myers, T. Patterson, N. Roehner, J. T. Stevens, and C. Winstead, "Design and test of genetic circuits using iBioSim," *IEEE Des. Test Comput.*, vol. 29, no. 3, pp. 32–39, June 2012.

[92] M. Zhang, J. A. McLaughlin, A. Wipat, and C. J. Myers, "SBOLDesigner 2: An intuitive tool for structural genetic design," *ACS Synth. Biol.*, vol. 6, no. 7, pp. 1150–1160, July 2017.

[93] N. Roehner and C. J. Myers, "Directed acyclic graph-based technology mapping of genetic circuit models," *ACS Synth. Biol.*, vol. 3, no. 8, pp. 543–555, Aug. 2014.

[94] T. Nguyen, T. S. Jones, P. Fontanarrosa, J. V. Mante, Z. Zundel, D. Densmore, and C. J. Myers, "Design of asynchronous genetic circuits," *Proc. IEEE*, pp. 1–13, 2019.

[95] G. Misirli, J. Hallinan, and A. Wipat, "Composable modular models for synthetic biology," *J. Emerg. Technol. Comput. Syst.*, vol. 11, no. 3, pp. 22:1–22:19, Dec. 2014.

[96] G. Misirli, A. Wipat, J. Mullen, K. James, M. Pocock, W. Smith, N. Allenby, and J. S. Hallinan, "BacillOndex: An integrated data resource for systems and synthetic biology," *J. Integr. Bioinforma.*, vol. 10, no. 2, June 2013.

[97] G. Mısırlı, J. Hallinan, M. Pocock, P. Lord, J. A. McLaughlin, H. Sauro, and A. Wipat, "Data integration and mining for synthetic biology design," *ACS Synth. Biol.*, vol. 5, no. 10, pp. 1086–1097, Oct. 2016.

[98] G. Misirli, T. Nguyen, J. A. McLaughlin, P. Vaidyanathan, T. S. Jones, D. Densmore, C. Myers, and A. Wipat, "A computational workflow for the automated generation of models of genetic designs," *ACS Synth. Biol.*, vol. 11, no. 2, May 2018.

[99] L. Huang, Z. Yuan, P. Liu, and T. Zhou, "Effects of promoter leakage on dynamics of gene expression," *BMC Syst. Biol.*, vol. 9, Mar. 2015.

[100] F. Battelli and C. Lazzari, "On the pseudo-steady-state approximation and Tikhonov theorem for general enzyme systems," *Math. Biosci.*, vol. 75, no. 2, pp. 229–246, Aug. 1985.

[101] E. Andrianantoandro, S. Basu, D. K. Karig, and R. Weiss, "Synthetic biology: New engineering rules for an emerging discipline," *Mol. Syst. Biol.*, vol. 2, no. 1, pp. 2006.0028, Jan. 2006.

[102] D. Del Vecchio, "Modularity, context-dependence, and insulation in engineered biological circuits," *Trends Biotechnol.*, vol. 33, no. 2, pp. 111–119, Feb. 2015.

[103] H. Chen, K. Shiroguchi, H. Ge, and X. S. Xie, "Genome-wide study of mRNA degradation and transcript elongation in Escherichia coli," *Mol. Syst. Biol.*, vol. 11, no. 1, Jan. 2015.

[104] D. W. Selinger, R. M. Saxena, K. J. Cheung, G. M. Church, and C. Rosenow, "Global RNA half-life analysis in Escherichia coli reveals positional patterns of transcript degradation," *Genome Res.*, vol. 13, no. 2, pp. 216–223, Jan. 2003.

[105] K. Nath and A. L. Koch, "Protein Degradation in Escherichia coli I. Measurement of rapidly and slowly decaying components," *J. Biol. Chem.*, vol. 245, no. 11, pp. 2889–2900, Oct. 1970.

[106] C. J. Myers, *Asynchronous Circuit Design*. Salt Lake City, Utah, USA: John Wiley & Sons, July 2001.

[107] T. S. Moon, C. Lou, A. Tamsir, B. C. Stanton, and C. A. Voigt, "Genetic programs constructed from layered logic gates in single cells," *Nature*, vol. 491, no. 7423, pp. 249–253, Nov. 2012.

[108] A. L. Slusarczyk, A. Lin, and R. Weiss, "Foundations for the design and implementation of synthetic genetic circuits," *Nat. Rev. Genet.*, vol. 13, no. 6, pp. 406–420, June 2012.

[109] P. Li, J. O. Dada, D. Jameson, I. Spasic, N. Swainston, K. Carroll, W. Dunn, F. Khan, N. Malys, H. L. Messiha, E. Simeonidis, D. Weichart, C. Winder, J. Wishart, D. S. Broomhead, C. A. Goble, S. J. Gaskell, D. B. Kell, H. V. Westerhoff, P. Mendes, and N. W. Paton, "Systematic integration of experimental data and models in systems biology," *BMC Bioinformatics*, vol. 11, pp. 582, Nov. 2010.

[110] Z. Wang, M. Gerstein, and M. Snyder, "RNA-Seq: A revolutionary tool for transcriptomics," *Nat. Rev. Genet.*, vol. 10, no. 1, pp. 57–63, Jan. 2009.

[111] Y. Xiang, N. Dalchau, and B. Wang, "Scaling up genetic circuit design for cellular computing: Advances and prospects," *Nat. Comput.*, vol. 17, no. 4, pp. 833–853, 2018.

[112] S. Ikushima and J. D. Boeke, "New orthogonal transcriptional switches derived from Tet repressor homologues for Saccharomyces cerevisiae regulated by 2,4-Diacetylphloroglucinol and other ligands," *ACS Synth. Biol.*, vol. 6, no. 3, pp. 497–506, Mar. 2017.

[113] M. W. Gander, J. D. Vrana, W. E. Voje, J. M. Carothers, and E. Klavins, "Digital logic circuits in yeast with CRISPR-dCas9 NOR gates," *Nat. Commun.*, vol. 8, pp. 15459, May 2017.

[114] A. Didovyk, B. Borek, L. Tsimring, and J. Hasty, "Transcriptional regulation with CRISPR-Cas9: Principles, advances, and applications," *Curr. Opin. Biotech.*, vol. 40, pp. 177–184, Aug. 2016.

[115] L. A. Gilbert, M. H. Larson, L. Morsut, Z. Liu, G. A. Brar, S. E. Torres, N. Stern-Ginossar, O. Brandman, E. H. Whitehead, J. A. Doudna, W. A. Lim, J. S. Weissman, and L. S. Qi, "CRISPR-mediated modular RNA-guided regulation of transcription in Eukaryotes," *Cell*, vol. 154, no. 2, pp. 442–451, July 2013.

[116] A. A. Nielsen and C. A. Voigt, "Multi-input CRISPR/Cas genetic circuits that interface host regulatory networks," *Mol. Syst. Biol.*, vol. 10, no. 11, pp. 763, Nov. 2014.

[117] L. S. Qi, M. H. Larson, L. A. Gilbert, J. A. Doudna, J. S. Weissman, A. P. Arkin, and W. A. Lim, "Repurposing CRISPR as an RNA-guided platform for sequence-specific control of gene expression," *Cell*, vol. 152, no. 5, pp. 1173–1183, Feb. 2013.

[118] G. Baldwin, T. Bayer, R. Dickinson, T. Ellis, P. S. Freemont, R. I. Kitney, K. M. Polizzi, and G.-B. Stan, Eds., *Synthetic Biology: A Primer*. Hackensack, NJ ; London,: Imperial College Press ; World Scientific Publishing Co. Pte. Ltd, revised edition edition, 2016.

[119] M. B. Elowitz, A. J. Levine, E. D. Siggia, and P. S. Swain, "Stochastic gene expression in a single cell," *Science*, vol. 297, no. 5584, pp. 1183–1186, Aug. 2002.

[120] D. K. Dacol and H. Rabitz, "Sensitivity analysis of stochastic kinetic models," *J. Math. Phys.*, vol. 25, no. 9, pp. 2716–2727, Sept. 1984.

[121] C. Damiani, A. Filisetti, A. Graudenzi, and P. Lecca, "Parameter sensitivity analysis of stochastic models: Application to catalytic reaction networks," *Comput. Biol. Chem.*, vol. 42, pp. 5–17, Feb. 2013.

[122] A. Gupta and M. Khammash, "An efficient and unbiased method for sensitivity analysis of stochastic reaction networks," *J. R. Soc. Interface*, vol. 11, no. 101, pp. 20140979, Dec. 2014.

[123] A. D. Irving, "Stochastic sensitivity analysis," *Appl. Math. Model.*, vol. 16, no. 1, pp. 3–15, Jan. 1992.

[124] S. Marino, I. B. Hogue, C. J. Ray, and D. E. Kirschner, "A methodology for performing global uncertainty and sensitivity analysis in systems biology," *J. Theor. Biol.*, vol. 254, no. 1, pp. 178–196, Sept. 2008.

[125] Y.-S. Lee, O. Z. Liu, H. S. Hwang, B. C. Knollmann, and E. A. Sobie, "Parameter sensitivity analysis of stochastic models provides insights into cardiac calcium sparks," *Biophys. J.*, vol. 104, no. 5, pp. 1142–1150, Mar. 2013.

[126] M. A. Marchisio and J. Stelling, "Automatic design of digital synthetic gene circuits," *PLoS Comput. Biol.*, vol. 7, no. 2, pp. e1001083, Feb. 2011.

[127] M. A. Marchisio, "Parts & pools: A framework for modular design of synthetic gene circuits," *Front. Bioeng. Biotechnol.*, vol. 2, 2014.

[128] K. Brenner, L. You, and F. H. Arnold, "Engineering microbial consortia: A new frontier in synthetic biology," *Trends Biotechnol.*, vol. 26, no. 9, pp. 483–489, Sept. 2008.

[129] K. Faust, "Microbial consortium design benefits from metabolic modeling," *Trends Biotechnol.*, vol. 37, no. 2, pp. 123–125, Feb. 2019.

[130] A. L. Gould, V. Zhang, L. Lamberti, E. W. Jones, B. Obadia, N. Korasidis, A. Gavryushkin, J. M. Carlson, N. Beerenwinkel, and W. B. Ludington, "Microbiome interactions shape host fitness," *PNAS*, vol. 115, no. 51, pp. E11951–E11960, Dec. 2018.

[131] F. Pinto, D. A. Medina, J. R. Pérez-Correa, and D. Garrido, "Modeling metabolic interactions in a consortium of the infant gut microbiome," *Front. Microbiol.*, vol. 8, Dec. 2017.

[132] O. S. Venturelli, A. V. Carr, G. Fisher, R. H. Hsu, R. Lau, B. P. Bowen, S. Hromada, T. Northen, and A. P. Arkin, "Deciphering microbial interactions in synthetic human gut microbiome communities," *Mol. Syst. Biol.*, vol. 14, no. 6, pp. e8157, June 2018.

[133] R. J. White, G. C. Y. Peng, and S. S. Demir, "Multiscale modeling of biomedical, biological, and behavioral systems (Part 1) [Introduction to the special issue]," *IEEE Eng. Med. Biol. Mag.*, vol. 28, no. 2, pp. 12–13, Mar. 2009.

[134] P. K. R. Kambam, M. A. Henson, and L. Sun, "Design and mathematical modelling of a synthetic symbiotic ecosystem," *IET Syst. Biol.*, vol. 2, no. 1, pp. 33–38, Jan. 2008.