GenoCAD: linguistic approaches to synthetic biology

Yizhi Cai

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Genetics, Bioinformatics, and Computational Biology

Jean Peccoud, Chair
David R. Bevan
Madhav V. Marathe
William T. Baumann

April 20, 2010
Blacksburg, Virginia

Keywords: Synthetic Biology, Linguistics, Computer Assisted Design, Bioinformatics

GenoCAD: linguistic approaches to synthetic biology

Yizhi Cai

(ABSTRACT)

Synthetic biology is an emerging interdisciplinary research field, which leverages the maturation of DNA synthesis technologies. By introducing engineering principles to synthetic biological systems design, synthetic biology shows great potential to shed new lights on biology and benefit human beings. Computer assisted design (CAD) tools will play an important role in the rational design of synthetic genetic systems. This dissertation presents the first CAD tool for synthetic biology – GenoCAD, a linguistic-based web application. By viewing DNA sequences as a language, we developed the first syntactic model to design and verify synthetic genetic constructs. Then we conducted a careful curation of the terminal set in the grammar - the first comprehensive analysis of the Registry of standard biological parts. The implementation and major features of GenoCAD are discussed, and in particular we showed how to develop a domain-specific grammar for BioBrick-based construct design and make GenoCAD a useful tool for the iGEM students. Finally, we went beyond the syntactic level to explore the semantics of synthetic DNA sequences: by associating attributes with biological parts and coupling semantic actions with grammar rules, we developed the first semantic models to relate the genotype to the phenotype of synthetic genetic constructs. The theories and techniques presented in this dissertation, along with the informative results presented, will serve as a foundation for the future developments of GenoCAD.

This work was jointly supported by a graduate fellowship from the graduate school at Virginia Tech to Y. Cai, Virginia Bioinformatics Institute and National Science Foundation Award # EF-0850100 to J. Peccoud, PI.

Dedication

$$Dedication \rightarrow Family$$

$$Family \rightarrow Parents|Wife$$

$$Parents \rightarrow Dad|Mom$$

$$Dad \rightarrow Jianshe\,Cai$$

$$Mom \rightarrow Yuzhen\,Huang$$

$$Wife \rightarrow Yijing\,Zheng$$

Acknowledgments

I would like to express my sincere gratitude and appreciation to my PhD committee members: Drs. Jean Peccoud, David R. Bevan, William T. Baumann and Madhav V. Marathe, for their mentorship, patience and encouragement during my time at Virginia Tech. This dissertation would not be in the current form without their insightful input and constructive criticism.

I was fortunate to be in the Genetics, Bioinformatics and Computational Biology (GBCB) interdisciplinary doctoral program, where I met a lot of great faculty members and fellow students. I gratefully thank Drs. Brett M. Tyler and David R. Bevan, and Ms. Dennie Munson, who helped me a great deal during my 3.5 years in the program and showed me the integrity of being a great scientist. Both Sarah and I would like to sincerely thank the vice president of Virginia Tech and the dean of graduate school Dr. Karen P. DePauw, and the director of VBI Dr. Harold "Skip" Garner, for helping us going through a very difficult time one month before my defense.

It has been a pleasure for me to work in Virginia Bioinformatics Institute. The interdisciplinary nature of this institute has played an important role in my scientific development. I thank Drs. João C. Setubal and Shrinivasrao P. Mane for teaching me a lot of Bioinformatics knowledge. I thank Ms. Jodi Lewis for providing me great assistance on traveling to conferences. I thank all the past, and current members in the synthetic biology group. I thank VBI Core Computational Facility (CCF) for assisting me a lot on computational matters. I had the privilege to chair the GenBioOrg student organization, and I thank Drs. Kristy DiVittorio and Ed Smith for providing funding support to GenBioOrg. And I enjoyed a lot working with my fellow GBCB students in promoting interdisciplinary research to the Virginia Tech community.

My interests in synthetic biology stemmed from my participation in the international Genetically Engineered Machines (iGEM) competition when I was in Edinburgh. I thank Dr. Christopher French and all my teammates for the nice summer we spent together in 2006, when we built an arsenic biosensor together. Through my PhD, I continuously received

advice and encouragement from my master advisor Dr. Gordon Plotkin from the University of Edinburgh.

I had several opportunities to present my work to different groups of audience at different universities. I sincerely thank my hosts: Dr. Roger Levy from the University of California at San Diego, Dr. Gábor Balázsi from MD Anderson Cancer Research Center and Dr. Jef Boeke from the Johns Hopkins University.

I am grateful to Dr. Matt Dyer for sharing his LaTeX dissertation template with me, and also Ms. Janice Austin for the help on ETD format. I thank Dr. David Ball for carefully proof-reading this dissertation. Without their help, I would not be able to finish this dissertation in less than a month, and I am sure it would not be as enjoyable.

I thank my best friends Drs. David Ball and Revonda Pokrzywa, Rebecca Shelton, John Cumbers, Yisha Luo, Kim Siung, Judith Nicholson, Matthew Lux, Laura Adam, Rong Song, Tian Hong, Pinyi Lu for their friendship, encouragement and help through out my PhD.

Finally, this dissertation owes a lot to my wife, Yijing "Sarah" Zheng. She turned down a 5-year fellowship from the University of Pittsburg and came to Blacksburg to support my endeavors of obtaining this PhD. I truly understand it is not easy to be the family of a "workaholic". Thank you for your love, faith, patience and understanding. I am also indebted to my parents, Yuzhen Huang and Jianshe Cai, for their unconditional love and support.

I would like to note that the research presented in Chapter 2 was published by Cai *et al.*, 2007 [29], and has been filed for a US patent application [156]; the research presented in Chapter 3 was published by Peccoud *et al.*, 2008 [155]; the research presented in Chapter 4 was published by Czar *et al.*, 2009 [48]; the research presented in Chapter 5 was published by Cai *et al.*, 2010 [31]; and the research presented in Chapter 6 was published by Cai *et al.*, 2009 [30]. It should be noted that all the five articles were distributed under the terms of the Creative Commons Attribution Non-Commercial License which permits unrestricted non-commercial use, distribution, and reproduction in any medium, provided the original work is properly cited. I sincerely thank all the co-authors for their valuable contributions to these publications.

I gratefully acknowledge the financial support provided by the graduate school at Virginia Tech, Virginia Bioinformatics Institute and National Science Foundation Award #EF-0850100.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"What I cannot create, I do not understand." - Richard Feynman (1918-1988)

1.1 Synthetic biology

In 1970, it took 20 man-years of labor to synthesize a 75 base pair (bp) DNA sequence [2]. Nowadays, one can easily order custom DNA sequences of thousands of base pairs from commercial gene foundries, with affordable cost and reasonable turnaround time. Owing to the rapid development of *de novo* DNA synthesis technology [47, 147, 211], the landscape of life science is undergoing a revolution. Synthetic biology (*aka* SynBio) is one of emerging interdisciplinary research fields, which leverage the power of DNA synthesis. Synthetic biology aims at designing artificial biological systems or modifying natural biological systems to carry out novel functionalities, using engineering principles [6, 11, 12, 16, 17, 33, 62, 66, 69, 85, 90, 101, 108, 112, 140, 193, 217, 225]. Synthetic biology distinguishes itself from traditional genetic engineering in several ways: traditional genetic engineering usually starts with natural DNA fragments (called "templates") and creates variant sequences by using techniques such as site-directed mutagenesis, in contrast synthetic biologists use *de novo* DNA synthesis technology to design DNA sequences without a pre-existing template; traditional genetic engineering is usually (if not always) a laborious "trial and error" process, while synthetic biology emphasizes the "rational design" approach with the introduction of engineering principles; finally synthetic biology aims at constructing artificial biological systems with novel functions, which in turn provides a great venue to re-visit some of the fundamental questions of traditional genetic engineering.

Despite its early stage, synthetic biology has already shown great potential to make significant scientific breakthroughs, which will improve the living conditions of human beings. The compelling examples of synthetic biology include, but are not limited to: engineer-

ing metabolic pathways of yeast to produce the antimalarial drug precursor artemisinic acid [173]; using engineered microorganisms to convert biomass to biofuel as a replacement of fossil fuels [113, 125, 160]; utilizing engineered bacteriophage as adjuvants for antibiotic therapy [134]; and detecting arsenic of drinking water with engineered *E.Coli* [4].

The genomes of such virus as polio [34] and $\phi X174$ virus [200] have been re-synthesized in an effort to better understand viral mechanisms to aid in the systematic design of new vaccines. Coleman *et al.* took a further step to synthesize polio virus capsid protein using underrepresented synonymous codon pairs, and the result showed a significant decrease in the protein translation rate of polio virus (so called "virus attenuation") [42]. Due to the evolutionary pressure, virus genomes are usually very compact and have many overlapping regions (a virus can encode two proteins within the same stretch of DNA fragment), which make it difficult to manipulate the virus genome from an engineering perspective. Chan *et al.* re-factored one quarter of the bacteriophage genome by eliminating the overlapping open reading frames, and demonstrated that the overlapping genetic elements are non-essential for T7 viability [35]. Lartigue *et al.* invented a technique called "genome transplantation" to use the genome from a donor bacterium *Mycoplasma mycoides* to "boot up" a recipient bacterium *Mycoplasma capricolum* [123]. At the moment of this dissertation being written, Gibson *et al.* hold the record for synthesizing the largest prokaryotic genome: they completely chemically synthesized the 582,970-base pair *Mycoplasma genitalium* genome from scratch [75]. It is also worthy of mention that at Johns Hopkins University, an interdisciplinary class (the Build-a-Genome course) has been set up for undergraduate students to learn synthetic biology and genetics by building an artificial eukaryotic genome, *Saccharomyces cerevisiae* chromosomes [56].

The above examples all use a top-down approach to the creation of synthetic biological systems, however there is another camp of synthetic biologists that has adapted a bottom-up approach to understand the principles of life by incrementally designing small-scale artificial gene circuits [40, 135, 209, 217, 222]. The pioneering work includes: building switches [51, 72, 120, 133], oscillators [61, 71, 205, 212], pulse generator [13], light detector [127], counter [70], synchronized clocks via quorum sensing [49], and mathematical problem solver [89]. Mathematical models play an important role in guiding the construction of these artificial gene circuits [60]. Although the complexity of these artificial genetic circuits is close to that of the first integrated circuit built by Kilby of Texas Instruments in 1958 [141], they shed new light on the mechanisms of genetics and molecular biology.

Like the rapid maturation of DNA synthesis technologies, the engineering principles brought by engineers are also instrumental to the progress of synthetic biology [62]. The first key principle is to standardize components, which can facilitate the assembly of a system. In mechanical engineering, imposing the dimensional standard for bolts, nuts and rivets makes it much easier to quickly build a system [100]. Similarly, the BioBrick Foundation (BBF,

bbf.openwetware.org) brought the "part" concept to synthetic biology. A "part" is a DNA fragment that carries definable biological functions, such as a promoter that initiates transcription, and a ribosome binding site (RBS) that controls translation [216]. A "standard part" is a part that conforms to a defined structure, such as being flanked by certain restriction sites. Chapter 5 elaborates the concept and usage of BioBricks. The Registry of Standard Biological Parts at MIT (the Registry, www.partsregistry.org) is the first and largest online catalog of BioBrick parts, and it has been widely used in the community especially by the students enrolled in the international Genetically Engineered Machines competition (iGEM, www.igem.org). The recent NSF-funded BioFab (www.biofab.org) is an ambitious project to design, fabricate and calibrate BioBrick parts. Standardizing biological components makes it possible to re-use parts in a "plug and play" manner, and also facilitates the fabrication process. After defining the basic biological components as parts, one can combine some parts to build a "device" (or "module") which performs a more complicated task, such as a logic NOT gate. Finally, by combining different modules one can scale up the complexity of the design to build a "system" [161]. The hierarchy of abstraction based on the complexity of the target problem helps designers to pay more attention to the detail of their working layer. The third engineering principle is to decouple design from fabrication, which is borrowed from the VLSI electronics engineering. The decoupling not only leads to the division of labour to increase the productivity, but also promotes the rational design of synthetic biology projects. Computer assisted design (CAD) is the use of computer technology to aid in the design of a product, and it has been widely used in many areas of engineering such as architectural engineering, mechanical engineering and electronic and electrical engineering [124]. Computer scientists have been developing CAD tools for synthetic biology [138, 207], with the hope that one day a synthetic biologist will be able to design custom DNA sequences *in-silico*, test their functions by running computer simulations, and finally send the DNA sequences for fabrication.

BioJADE was one of the earliest examples of CAD software for synthetic biology design [79]. The program adapted ideas from electronic design and implemented abstract representations of genetic components and designs. As a proof-of-concept program and because the necessary connection between the Registry was shut off, BioJade has never been publicly released. Asmparts is a command line based Linux application to perform the assembly of biological parts [175]. Without a graphical user interface, the use of Asmparts is challenging for users lacking experience with Linux. To date, Asmparts can only deal with promoter, RBS, coding sequence and terminator, which limits its design capability. Gene Designer is a software tool developed by gene synthesis company DNA2.0 to construct synthetic DNA segments [215]. Gene Designer has a nice user interface which allows users to easily design a construct by "drag and drop" parts. Gene Designer provides many useful features, *e.g.*, codon optimization, real-time Tm calculation for oligonucleotides design, and seamless connection with the gene synthesis pipeline of DNA2.0. GeneDesign is a Perl-based web platform for large-scale synthetic gene designs [168]. GeneDesign integrates multiple handy functionalities, including reverse translation, codon juggling, silent restriction site removal,

oligonucleotide design and sequence analysis modules. ProMoT is another drag-and-drop tool for designing and simulating synthetic circuits [137]. ProMoT introduced quantifiable signal carriers to describe the interactions between parts and the concept of pools to collect free signal carriers. ProMoT requires each part to be in the format of modeling description language (MDL), which impedes the creation of new parts by the typical user. Designing a genetic circuit in ProMoT not only requires users to put parts together in the right order, but also requires users to connect the correct ports of adjacent parts with wires, which increases the possibility of design errors. SynBioSS [92] allows users to run a computational simulation even on a supercomputer, however it requires extensive user inputs to construct a system.

1.2 Formal language

A language is a set of (possibly infinite) strings derived from an alphabet Σ, and it encodes information for communication purpose [208]. There are several kinds of languages, including natural languages (*e.g.*, English and Chinese), computer languages (*e.g.*, C and Perl), and mathematical languages (*e.g.*, first-order logic). However not all the strings over the alphabet belong to a language, only those which follow certain rules are part of a language. A grammar is a finite set of rules that specifies the syntax (permissible structure) of a language. A grammar G contains four components:

- A finite set N of non-terminal symbols.
- A finite set Σ of terminal symbols that is disjoint from N.
- A finite set P of rewriting rules, each rule is in the form of $\alpha \to \beta$, where α and β are both strings of symbols, and α contains at least one symbol from N. More formally put, a rewriting rule can be represented as $(\Sigma \cup N)^* N (\Sigma \cup N)^* \to (\Sigma \cup N)^*$, where * is the Kleene star operation and $\cup$ is the set union operation.
- A distinguish symbol $S \in N$ that is the start symbol.

In the 1950s, Chomsky classified grammatical models into four classes based on the forms of their production rules, which reflect their expressive power [41]. In the following definitions of these four classes of formal grammars, $A, B \in N$, $a, b \in \Sigma$, and $\alpha, \beta \in (N \cup \Sigma)^*$.

Regular grammar is the most restricted class. Only rewriting rules of the form $A \to a$ or $A \to aB$ are allowed. The left-hand side only contains a single non-terminal symbol A, and the right-hand side contains a terminal symbol a followed by an optional non-terminal symbol B. The equivalent abstract computational device (automaton) is a finite state automaton. The computational complexity to recognize regular grammars is linear.

Context-free grammar allows any production rule of the form $A \rightarrow \alpha$. The left-hand side only consists of a single non-terminal symbol A, and the right hand side can be any string α. The corresponding automaton for a context free grammar is a push-down automaton. The computational complexity to recognize a context free grammar is polynomial.

Context-sensitive grammar allows rewriting rule of the form $\alpha_1 A \alpha_2 \rightarrow \alpha_1 \beta \alpha_2$. The transformation of non-terminal symbol A to a string β depends on its context α_1 and α_2. It also requires that the right-hand side contains at least as many symbols as the left-hand side, *i.e.,* $|A| \leq |\beta|$. The equivalent automaton for context-sensitive grammar is a linear bounded automaton, and the computational complexity is exponential.

Unrestricted grammar uses unrestricted rules in the form of $\alpha \rightarrow \beta$, where $\alpha \neq \epsilon$. Unrestricted grammars allow any non-empty string to be transformed to any strings. The corresponding automaton for unrestricted grammars is a Turing machine, and the complexity of recognition becomes undecidable.

Even though the formal language theory was developed almost at the same time as the discovery of DNA structure in the 1950s [221], it was not until thirty years later that researchers started to look at the biological sequences as a new kind of languages. Specifically, in 1982, Doerfler *et al.* demonstrated the structural similarities between natural languages and the genetic language [53]. The advantages of treating biological sequences and potentially entire genomes as languages are multi-fold: it provides a concise generalization about the information contained in the biological sequences and it opens the possibility for the analysis of biological sequences to take advantages of linguistic methods originally developed for computer science and computational linguistics [183, 187, 188, 189].

Linguistic applications to biological sequences include sequence alignments using finite state automatons [185, 190], pattern recognition and motif finding in biological sequences [23, 95, 170, 176], the representation of various structural features of DNA sequences [39, 77, 126] and RNA pseudoknots which are difficult due to the crossing dependences in the secondary structure [26, 28, 57, 106, 142, 162, 171, 172], and the inference of grammars from naturally existing sequences [148, 177, 178, 179, 180, 181, 202].

1.3 Organization

Chapter 1 sets the stage for the dissertation: we started with an introduction and overview of synthetic biology, and focus on the existing computer assisted designs for synthetic biology; then we will introduce the fundamental concepts of computational linguistics and a brief review of linguistic applications to biological sequences, with the hope to make this

dissertation more accessible to the readers without much background in linguistics.

We begin in Chapter 2 with a syntactic model to design and verify synthetic constructs. The syntactic model captures some basic design principles of synthetic genetic constructs in the form of context-free grammar (CFG). To the best of our knowledge, this is the first linguistic model for designing synthetic constructs, and it lays the theoretical foundation for the following chapters.

Chapter 3 focuses on the curation of the terminals in the grammar – the biological parts. We analyzed the Registry of Standard Biological Parts (Registry) in two steps: first, we computed inclusion relationships between parts based on theoretical sequences to study the usage pattern of the parts and the abstraction hierarchy of the Registry; second, we sequence verified all the clones that generated a single PCR (Polymerase Chain Reaction) fragment greater than 120 bp. We conclude this chapter by discussing the organizational guidelines for developing next generation Registry of parts based on the findings of this study.

Chapter 4 presents GenoCAD™, a web-based computer assisted design for design and verification of synthetic constructs. GenoCAD is built upon the linguistic model described in Chapter 2. In this chapter, we focus on the major features, implementation and data model of GenoCAD.

Chapter 5 connects previous chapters: we formalized various BioBrick assembly schemes in the form of context-free grammar, categorized BioBrick parts based on the inclusion relationships of restriction sites, and finally implemented the grammar in GenoCAD. This chapter shows a concrete example how to transform domain-specific knowledge into a linguistic model. The implementation of this linguistic model in GenoCAD helps the iGEM students to quickly design synthetic genetic systems from a rich library of parts compliant with six popular BioBrick standards.

As previous chapters all focus on the syntactic level of synthetic DNA sequences, Chapter 6 takes a further step to explore the semantics of synthetic DNA sequences. Attribute grammars have been widely used in building compilers to translate human-friendly source codes to object codes that can be executed by microprocessors. By associating attributes to parts and coupling semantic actions with context-free rules, we managed to use attribute grammars as a means to relate genotype to the phenotypes encoded by the DNA sequences. After introducing the development of an attribute grammar for synthetic DNA sequences and the workflow of compiling the mathematical models from DNA sequences, we demonstrated one application of this semantic framework, that is to systematically explore the design space of genetic toggle switches.

Finally, we conclude with an outlook and some perspectives on the future work in this research area in Chapter 7.

Chapter 2

A syntactic model to design and verify synthetic genetic constructs derived from standard biological parts

Published at: Y. Cai, B. Hartnett, C. Gustafsson, and J. Peccoud. A syntactic model to design and verify synthetic genetic constructs derived from standard biological parts. Bioinformatics, 23(20):27607, Oct 2007.

Authors: Yizhi Cai, Brian Hartnett, Claes Gustafsson and Jean Peccoud

ABSTRACT

Motivation: The sequence of artificial genetic constructs is composed of multiple functional fragments, or genetic parts, involved in different molecular steps of gene expression mechanisms. Biologists have deciphered structural rules that the design of genetic constructs needs to follow in order to ensure a successful completion of the gene expression process, but these rules have not been formalized, making it challenging for non-specialists to benefit from the recent progress in gene synthesis.

Results: We show that context-free grammars (CFG) can formalize these design principles. This approach provides a path to organizing libraries of genetic parts according to their biological functions, which correspond to the syntactic categories of the CFG. It also provides a framework for the systematic design of new genetic constructs consistent with the design principles expressed in the CFG. Using parsing algorithms, this syntactic model enables the verification of existing constructs. We illustrate these possibilities by describing a CFG that generates the most common architectures of genetic constructs in *Escherichia coli*.

Availability: A web site allows readers to experiment with the algorithms presented in this article: `www.genocad.org`

Contact: peccoud@vt.edu

Supplementary information: Sequences and models are available at Bioinformatics online.

2.1 Introduction

Gene synthesis technology now enables molecular biologists to assemble long DNA molecules that may include multiple genes and their regulatory sequences. We will refer to these molecules as 'genetic constructs' or just 'constructs'. As the throughput of construct manufacturing increases, the design of complex genetic constructs becomes the bottleneck of the process. It becomes easier to assemble complex DNA molecules than to design them. A natural way of designing complex constructs involves combining basic building blocks also known as 'biological parts' or 'genetic parts' [17, 90, 216]. These parts are small DNA fragments implementing specific biological functions. The mechanisms of gene expression require that certain structural constraints are met in order for a construct to be functional. Parts of different types need to be placed in a particular order and next to each other in order to ensure that coding sequences are properly transcribed and translated. Certain parts are functional only in a specific context whereas other parts have proved functional in organisms other than the one from which they originate. For instance, promoters are often restricted to specific organisms or even cell types [149, 159, 226] whereas genes coding for proteins can often be expressed in multiple species [78]. The design of complex genetic constructs such as artificial gene networks [40, 61, 72, 86, 87, 90, 118, 140] therefore requires an intimate knowledge of gene expression mechanisms. It is interesting to observe that more than 6 years after the description of the first artificial gene networks [61, 72], this technology has yet to find biomedical applications. It is likely that most biologists who could use sophisticated genetic constructs to control the expression of their gene of interest do not have the expertise to design the construct they need. One way to lower the barrier to entry into synthetic biology is to formalize the structural constraints associated with the use of standardized biological parts in a construct. Such formalism can be used to build software wizards to guide users in the design of their constructs. It can also provide a foundation to the development of parsers capable of verifying the structural validity of a synthetic DNA sequence.

Several prominent synthetic biologists have advocated an engineering approach to the design of genetic constructs [62, 85] well illustrated by the Registry of Standard Biological Parts, a service provided by MIT to promote the development and dissemination of well-specified, standardized and interchangeable biological parts. The records in this database are organized in different categories corresponding to different levels of abstraction [62]. At the bottom of this hierarchy lay the basic parts. Parts can be combined in functional modules called devices. Devices and parts can ultimately be combined in self-contained systems. The 'Parts' category is itself divided into subcategories (Regulatory, Terminators, RNA, DNA, Protein Coding,Ribosome Binding Sites and Conjugation) corresponding to biological functions. The

database enables users to create new records by combining existing records corresponding to basic parts, devices or construction intermediates. Standardized graphical representation of complex records makes it easy to visualize their structure. After examining a number of records, it is possible to identify common features shared by many entries. However, the record editing process is unconstrained; no structural rule is imposed on new records nor are the records automatically verified upon submission.

The development of Gene Designer [215], a software application to quickly design synthetic DNA molecules from a library of basic parts, has been inspired by a similar vision. The user interface includes a standard library of parts called the Design Toolbox. Its hierarchical organization is multilayered to accommodate sequences specific to multiple biological species and a broader spectrum of biological functions than in the MIT Registry. Gene Designer makes it very easy to drag elements of the toolbox into new DNA sequences. The structure of complex sequences combining multiple parts is represented by an icon view. Gene Designer does not provide a wizard to guide the user in the design of a construct nor does it have a feature to verify the structural validity of constructs.

In mathematics, logic and computer science, a formal language is a language that is defined by precise mathematical or machine processable formulas. Like natural languages, these formal languages generally have two aspects. The syntax of a language is what the language looks like (more formally: the set of possible expressions that are valid utterances in the language). The semantics of a language are what the utterances of the language mean. The syntax or grammar of the language can be formally defined by the specification of a set of non-terminal symbols or variables, a set of terminal symbols and a set of production rules also called transformation rules. The variables represent categories of words such a nouns and verbs; they are often referred to as syntactic categories. The terminals represent actual words such as 'dog' or 'sing'. The production rules map one string of symbols to another, where the first string contains at least one non-terminal symbol. The recursive application of production rules, beginning from the start variable often denoted S, generates the set of strings containing only terminal symbols, which is the language generated by the grammar in which every production rule is of the form $V \rightarrow w$, where V is a single non-terminal symbol and w a string of terminals and/or non-terminals (possibly empty). The term "context-free" expresses the fact that non-terminals are rewritten without regard to the context in which they occur.

We have developed a context-free grammar that formalizes the structure of a library of previously published artificial genetic constructs, which are derived by combining standard genetic parts. We show how this syntactic model provides a rigorous foundation for the organization of a parts library in syntactic categories defined according to the structural constraints affecting the position of parts in genetic constructs. In addition, we show how this model enables a systematic approach to the design of genetic constructs that can be

implemented in software. Last, this model can be used to build parsers capable of accepting constructs consistent with the design principles captured by the CFG production rules.

Early applications of linguistic models in the analysis of biological sequences have been reviewed in an article that also provides a short introduction to these types of models [188]. Most of these early works were attempting to analyze naturally occurring sequences. Grammars were developed with the goal of understanding genome structures [24, 25] and associating genes with their regulatory sequences [43]. Another body of work focused on predicting the secondary structures of RNA molecules [114, 115, 139, 172, 180]. The discovery of grammatical models from sets of curated biological sequences remains a very active field of research in the machine-learning community [178]. Linguistic models have also been used to analyze proteins with different purposes. Most of the work in this field attempts to understand the rules of protein organization in modular domains [76], but recently grammatical models have been developed with the goal of designing new antimicrobial peptides [132]. This work proceeded in two steps. In order to decipher the design principles of natural antimicrobial peptides, a set of grammars was inferred from natural sequences using a pattern discovery algorithm [169]. In a second step, 42 peptides consistent with the discovered grammars but not homologous to natural peptides were synthesized and tested. Approximately half of the new peptides exhibited an anti-microbial activity, which demonstrates the power of this approach. In the context of this article, we have also used formal grammars to support the design of new DNA sequences, which is a very different goal from the analysis of natural genomic sequences. Instead of inferring the production rules from a training data set, our production rules utilize pre-existing biological knowledge relative to the structure of functional genetic constructs.

2.2 Methods

2.2.1 Variables

The first step in the construction of the grammar is to recognize syntactic categories in categories used to organize genetic parts. These syntactic categories are represented by the CFG variables listed in Table 2.1. The specific CFG presented in this article has only 26 syntactic categories each represented by a single capital letter. The orientation of constructs can be left to right or right to left depending on which DNA strand is transcribed. If left to right is the direct orientation and right to left the reverse orientation, each category of genetic parts needs to be broken down into two syntactic categories corresponding to the direct and reverse orientations as different structural rules apply to each.

Variables have been organized in four hierarchical categories. The first category contains

only S, the start variable from which all derivations are initiated. The second category corresponds to complex fragments of DNA composed of multiple functional parts. This category includes the variables M and N which correspond to transcripts in the forward and reverse orientation, respectively. In the context of this article, a transcript is a DNA fragment located between a promoter and a transcription terminator. Also in this category are the variables E and F used to represent genes defined as DNA fragments composed of a "start" codon followed by one or more protein domains and terminated by a "stop" codon. The third category of variables includes parts that can be duplicated in a construct. For instance, it is common practice to put two transcription terminators G at the end of a transcript to ensure a tight termination of the transcript. The fourth category contains all the variables that represent basic genetic parts that cannot be decomposed into smaller functional blocks and are not used in series in genetic constructs such as A (promoter), C (ribosome binding site) or P (T7 promoter). Variables representing less frequently used parts such as I and J (riboregulators) are also included in this category. The boundaries between the four categories used in Table 2.1 are arbitrary and have no consequence on the rest of the development. Listing variables in alphabetical order would have been equally acceptable.

Instead of using single capital letters that are difficult to interpret, we initially used more descriptive variables such as "promoters", "RBS", "coding", etc. Using descriptive notations hindered the visual display of complex sequences on the GenoCAD web site. By using single letters as variables, it was possible to generate a more compact graphical representation of the sequence and production rules. The lack of information in the single letter variables was compensated for by creating icons associated with each variable and by displaying in a mouse-over tooltip the description of each variable as it appears in the center column of Table 2.1.

2.2.2 Terminal set

The terminal set is composed of the genetic parts themselves. A library of more than 100 parts has been organized according to the syntactic categories used in this article. Parts have been indexed by a unique identifier composed of a prefix corresponding to the part syntactic category and a numerical suffix indexing the parts within each category. For instance, the terminals a01 to a09 point to the promoters of the library, whereas genes are represented by the terminals e01 to e14, etc. This library is distributed in two computer-readable formats in the Supplementary Material. In addition to the unique identifiers used as terminals in the CFG, the library files include a part name. In addition, the DNA sequence of each part is included in the library as a proof of concept. These sequences have not been validated experimentally and it is sometimes difficult to extract precise sequence information from the sequences of previously published genetic constructs as the delimitation of parts within the construct is often sketchy in the literature. The development of an experimentally validated

Table 2.1: Variable set.

Name	Description	Category
S	Start	I
E	Gene coding region	II
F	Gene coding region reverse	
M	Transcript	
N	Transcript reverse	
G	Terminator	III
H	Terminator reverse	
O	Linker	
R	T7 terminator	
T	T7 terminator reverse	
U	Protein domain	
V	Protein domain reverse	
Y	Stop codon	
Z	Stop codon reverse	
A	Promoter	IV
B	Promoter reverse	
C	Ribosome binding site	
D	Ribosome binding site reverse	
I	Riboregulator	
J	Riboregulator reverse	
K	Hammerhead ribozyme	
L	Hammerhead ribozyme reverse	
P	T7 promoter	
Q	T7 promoter reverse	
W	Start codon	
X	Start codon reverse	

parts library is beyond the scope of this article.

2.2.3 Productions and construct design

Table 2.2 includes a list of production rules grouped according to the successive steps followed when designing a genetic construct. The process starts at S, the transcript. P01 can be applied to S several times to fix the construct total number of transcripts. Step 2 of the design process will specify each transcript by choosing a type of promoter and an orientation. Applying P02 to S will ensure that the transcript uses the endogenous RNA polymerase by selecting promoters and transcription terminators compatible with this enzyme. Alternatively, the transcript could rely on the bacteriophage T7 RNA polymerase in which case P04 will be applied to S. Using P02 or P04 will result in transcripts in the direct orientation. Alternatively, P03 or P05 can be used to generate transcripts in the reverse orientation. In Step 3, it is possible to specify if the transcript is polycistronic by applying P06 or P07 in the direct or reverse orientation, respectively. In Step 4, the architecture of transcripts is specified. P08 specifies that M is regular mRNA by decomposing it into a Ribosome Binding Site (RBS) C and a coding sequence E whereas P09 can be used when M is composed of a riboregulator I placed between two ribozymes K [14]. The coding sequence E can itself be broken down by P12 into a start codon W, a protein domain U and a stop codon Y. Productions P10, P11 and P13 are the counterparts of P08, P09 and P12 for sequences in the reverse orientation. It is not unusual to place more than one part of a particular type in a specific location. Step 5 can be used to specify the number of repetitions for each part of the construct that can be repeated. For instance, multiple linkers corresponding to different restriction sites can be placed between transcripts by applying P16 several times. Similarly, it is common to place two successive transcription terminator sequences (P14, P15) or two stop codons (P17, P18) to ensure a tight termination of transcription and translation, respectively. P19 and P20 can be used to place additional protein domains to the coding sequence of a gene. In Step 6, it is possible to add linkers, DNA elements having a structural role but not involved in the gene expression mechanisms, next to some parts in the constructs. Typical linkers include restriction sites that could be used to extract parts in a construct and replace them by ligation of a DNA fragment extracted from a different construct.

At this stage of the design process, the general architecture of the construct is completely specified as a series of parts belonging to specific functional categories. However, the specific parts used to build the construct are yet to be specified. For instance, the construct could be described by a string such as $ACWUUY$ (promoter, RBS, start codon, 2 protein domains, stop codon) but the particular promoter, RBS, start and stop codons, or the protein domains used to assemble a specific construct have not yet been specified. Therefore, this string does not describe a specific construct but a family of constructs expressing a protein. This family includes a wide range of transcription and transcription levels and any protein composed of

Table 2.2: Production rules.

P01	$S \rightarrow SOS$	Start symbol (S), linker (O), start symbol (S)	Step 1
P02	$S \rightarrow AMG$	Promoter (A), transcript (M), terminator (G)	Step 2
P03	$S \rightarrow HNB$	Terminator rev (H), transcript rev (N), promoter rev (B)	
P04	$S \rightarrow PMR$	T7 promoter (P), transcript (M), T7 terminator (R)	
P05	$S \rightarrow TNQ$	T7 terminator rev (T), transcript rev (N), T7 promoter rev (Q)	
P06	$M \rightarrow MM$	Transcript (M), transcript (M)	Step 3
P07	$N \rightarrow NN$	Transcript rev (N), transcript rev (N)	
P08	$M \rightarrow CE$	Ribosome binding site(C), gene (E)	Step 4
P09	$M \rightarrow KIK$	Hammerhead (K), riboregulator (I), hammerhead (K)	
P10	$N \rightarrow FD$	Gene rev (F), ribosome binding site rev (D)	
P11	$N \rightarrow LJL$	Hammerhead rev (L), riboregulator rev (J), hammerhead rev (L)	
P12	$E \rightarrow WUY$	Start codon (W), protein domain (U), stop codon (Y)	
P13	$F \rightarrow ZVX$	Stop codon rev (Z), protein domain rev (V), start codon rev (X)	
P14	$G \rightarrow GG$	Terminator (G), terminator (G)	Step 5
P15	$H \rightarrow HH$	Terminator rev (H), terminator rev (H)	
P16	$O \rightarrow OO$	Linker (O), linker (O)	
P17	$Y \rightarrow YY$	Stop codon (Y), stop codon (Y)	
P18	$Z \rightarrow ZZ$	Stop codon rev (Z), stop codon rev (Z)	
P19	$U \rightarrow UU$	Protein domain (U), protein domain (U)	
P20	$V \rightarrow VV$	Protein domain rev (V), protein domain rev (V)	
P21	$A \rightarrow OA$	Linkers can be added next to some parts	Step 6
P22	$B \rightarrow OB$		
$\cdots$	$\cdots$		
P0100$\cdots$	$A \rightarrow a1\vert\cdots$	All variables can be transformed into terminals	Step 7
P0200$\cdots$	$B \rightarrow b1\vert\cdots$		
$\cdots$	$\cdots$		
P2400$\cdots$	$Z \rightarrow z1\vert\cdots$		

three domains.

The last phase of the design process (Step 7) involves transforming variables into terminal symbols pointing toward specific DNA sequences. Productions corresponding to this step are the most numerous because there is one production for every part available to the designer. Table 2.2 provides only the general architecture of this last group of productions. Productions starting from the same variable have been grouped on a single line using the standard notation $Variable \rightarrow Terminal\,1|Terminal\,2|\ldots$ indicating that a variable can be transformed into any of the terminals separated by |. All the grammar variables can potentially be transformed into a terminal or this type of transformation can be restricted to a category of variables corresponding to the most basic genetic parts. For instance, a variable like E (gene) can be transformed into terminals corresponding to a self-contained coding sequence or it can be transformed into a coding sequence composed of multiple domains between a start and stop codon. The most extreme case would be to include productions allowing the transformation of the start symbol S into a terminal. Allowing this type of production in the grammar maximizes flexibility since any DNA fragment can be made valid. However, this option makes it possible to completely bypass the design process enforced by the grammar, which may not be desirable. The design process is completed when all non- terminal variables have been transformed into terminals. At this stage the construct is represented by a series of terminal part identifiers. This high-level description of the construct can be converted into a DNA sequence suitable for gene synthesis using the sequence data of each of the parts in the part library.

2.3 Results

2.3.1 Parsing for construct verification

The construct design process applies a series of productions starting from S to generate a construct with a structure consistent with the grammar rules. The design process therefore "derives" the construct from S. A computationally more complex problem is evaluating whether or not a specific construct can be generated by a given grammar. In order to answer this question it is necessary to construct the DNA sequence into S through the application of the grammar productions. This operation is called parsing. By parsing a construct, it is possible to verify its design, which is useful if the construct was not generated by the systematic process outlined in the previous section.

Prior to parsing the construct, it is necessary to perform a lexical analysis of the construct DNA sequence to transform it into a series of parts [7]. As a proof of concept, we have developed a basic lexical analyzer that scans the parts list and compares the sequence of

each part with the start (leftmost) sequence of the construct. If the part does not match the start of the construct sequence, the next part in the library is evaluated. At the end of the scan, it is possible that no part matches the beginning of the construct sequence, in which case, the construct is rejected. If only one match is found, then the part matching the construct sequence is recorded and the rest of the construct DNA sequence is analyzed in the same way. It is also possible that several matches will be found if the parts library includes complex parts composed of more basic parts. In this case, all the matches are recorded possibly leading to multiple lexical interpretations of the construct sequence. The presence of multiple interpretations of a construct DNA sequence is an indication that the parts list is redundant in the sense that it includes complex parts that can be obtained by concatenation of more basic parts. It would be preferable to ensure that the CFG defined in the parts library includes rules allowing the derivation of complex parts from the basic parts.

The development of efficient parsing algorithms is an important problem in computer science since its solutions directly affect the performance of interpreters and compilers of programming languages. An introduction to parsing methods can be found in computer science textbooks [129]. JFLAP is a very nice tool allowing the non-specialist to experiment with formal languages with a strong emphasis on automata theory [174]. JFLAP, however, is not suitable for the development of complex grammars or the analysis of large strings. YACC and Bison are production grade tools that can be used to develop compilers that rely on the LALR parsing algorithm. It is possible to code the grammar defined in Table 2.2 into Bison to build a custom parser capable of analyzing genetic constructs (data not shown). However, this requires proficiency in the C programming language and each time the grammar or the parts list is edited, the parser needs to be recompiled.

We have therefore developed a custom parser relying on the LR(0) algorithm and a specific precedence of the productions. After the lexical analysis step, the input string is converted into a series of non-terminal variables through the productions listed in Step 7 in Table 2.2. In a second step, the parser eliminates possible shift-reduce conflicts by eliminating series of identical variables that recursive productions can create in the construct (Steps 3 and 5 in Table 2.2). Finally, the resulting string is processed using the precedence set by the order of the productions. The Supplementary Material includes an animation that may help readers unfamiliar with parsing algorithms to visualize the process.

2.3.2 Validation

In order to validate the grammar in Table 2.2 and the parsing algorithm, a series of complex constructs described in the MIT Registry and in various publications [61, 72, 86] is reported in Table 2.3. Each construct is recognized by the identifier used in the source reference. Constructs are described by a series of lexical tokens corresponding to basic genetic parts.

Table 2.3: Parsing results of selected parts from the Registry.

ID	Source	Symbolic representation	Result	Comment
BBa_J04450	Registry	a03c01e01g03	Pass	
BBa_I13520	Registry	a01c01e01g01g02	Pass	
pMKN7a	[72]	a08c08e14g01g02	Pass	
BBa_J13004	Registry	a02c01e03c01e04g01g02	Pass	
BBa_I13513	Registry	a01c01e01o02c01e09g01g02	Pass	
pTAK102	[72]	h02h01f01d04b02a08c07e14g01g02	Pass	
pTAK117	[72]	h02h01f01d04b02a08c08e15c05e14g01g02	Pass	
BBa_J23022	Registry	I01g01g02	Fail	No promoter
BBa_J36335	Registry	a03c01e05a03c01e06	Fail	Lack of terminator
BBa_J44003	Registry	o01a04o01c02e07	Fail	Lack of terminator
BBa_J45119	Registry	c03e02g01g02	Fail	No promoter
BBa_J52038	Registry	Registry	Fail	No RBS, no terminator
BBa_E0241	Registry	c03e09g04	Fail	No promoter
BBa_J5516	Registry	a01c01e12g01g02a06	Fail	Orphan promoter in 3'

Most constructs were selected to illustrate different types of construct architectures generated by the grammar. However, some constructs outside of the language generated by the grammar have also been introduced in this validation set to illustrate structures outside the language generated by the grammar. The outcomes of the construct parsing are reported along with some comments explaining why some constructs failed the verification. A larger set of test cases than can be reported in Table 2.3 is available in the Supplementary Material.

In order to validate the lexical analyzer, we analyzed the sequences of several bistable genetic switches (US Patent 6841376). Tables 35 of this publication indicate the location of the promoters, RBS and genes used to implement the switches. In addition, the sequence list provides the sequence data for the promoter and RBS in addition to the complete sequences of the plasmids. The sequence of the transcription terminators labeled T1T2 in Figure 3 of [72] does not appear to be documented. The sequence of part g01 (also BioBrick BBa_B0010) was identified in the location of T1, but the sequence of the second terminator T2 could not be identified. We also noticed that the sequence of the *GFP-mut3* and LacI genes found in the published plasmid sequences is not exactly the same as the sequence published in the MIT Registry. We therefore created new parts (e17, f03, f04) corresponding to variants of these genes. We also observed that RBS sequences overlapped the sequence of the genes placed in 3' since all the RBS sequences included the start codon ATG. We have edited the RBS sequence to remove the ATG codon responsible for the overlap between the RBS and gene sequence. The sequences of the promoters used to build the switches included the AGGA

A

Step	Production	String
1	P01	**S**S
2	P03	HNB**S**
	P02	HNBA**M**G
3	P06	H**N**BAMMG
4	P10	HFDBA**M**MG
	P08	HFDBACE**M**G
	P08	**H**FDBACECEG
5	P15	HHFDBACECEG
	P14	HHFDB**A**CECEGG
6	P21	HHFDB0ACECEGG
7		h02h01f01d04b02o01a08c08e15c05e14g01g02
		tataa.....gcgtttata

B

[tataaacgcagaaaggcccacccgaaggtgagccagtgtga][gagagcgttcaccgacaaacaacaga
taaaacgaaaggcccagtctttcgactgagcctttcgttttatttgatgcctgg][ttaagc…cacca][
catcgaaccggtttcct][tcctttgcataccctgctgatgtgctcattataaccgccagtggtatttat
gtcaacaccgccagagataatttatcaccgcagatggttatctgtgcatgc][ttatcaaaaaccatggt
ttttgataa][ccatcgaatggctgaaatgagctgttgacaattaatcatccggctcgtataatgtgtgg
aattgtgagcggataacaatttcacacagga][aggaaaccggttatg][atgagca…ttacaa][agga
atttaaatg][atgcgt…aaataa][ccaggcatcaaataaaacgaaaggctcagtcgaaagactgggcc
tttcgttttatctgttgtttgtcggtgaacgctctc][tcacactggctcaccttcgggtgggcctttct
gcgtttata]

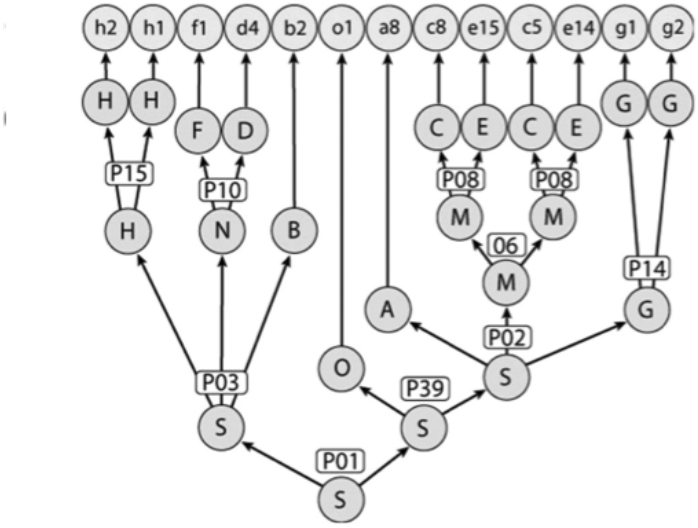


Figure 2.1: (A) The successive applications of productions starting from S provide a framework to guide the design of genetic constructs. (B) The verification of an existing DNA sequence requires the use of a lexical analyzer to identify the parts composing the sequence. The symbolic description of the sequence provided by the lexical analyzer can be parsed using an LR algorithm

motif of the Shine-Delgarno sequence which was also part of the RBS sequences. By removing AGGA from the promoter sequences, we resolved the overlap between the promoter and RBS sequences. We also observed a discrepancy between the sequence of the *PL-s1con* promoter (Sequence 2 in the patent and part a09 and b02 for the reverse orientation) and the sequence used in the plasmids that led to the introduction of part b04. Finally, we observed sequence variations in the *LacI* gene in the different plasmids, which we addressed by creating parts f03 and f04. The regions of the plasmids outside of the switches were treated as linkers in the context of this verification. The annotated sequence of the six switches is provided in the Supplementary Material. We introduced variants of the different parts found in the plasmid sequences into the parts list. The DNA sequences of the six switches could be analyzed by the lexical analyzer and their structure verified by the parser. All the plasmids could be parsed.

2.4 Discussion

2.4.1 Grammar form and limitations

Even though a single grammar has been presented in this article, it is important to stress that this grammar is a somewhat arbitrary set of design of principles. It certainly does not encompass all natural DNA sequences. Even designers of some synthetic constructs have used unusual architectures that are not included in the language generated by our grammar. For instance, multiple promoters have been used to control the expression of a gene [67]. By adding the production $A \rightarrow AA$ to the grammar in Table 2.2, it would be possible to authorize the use of multiple promoters in constructs. This example shows that the grammar is nothing more than a set of accepted rules selected by the user of GenoCAD to design new constructs or analyze pre-existing ones. These rules come from the current understanding of the molecular mechanisms controlled by the different genetic parts used in genetic constructs.

The set of production rules listed in Table 2.2 was structured with the goal of minimizing the number of variables to ensure a biological interpretation of these variables. In addition, the rules have been organized in a way that mimics the way biologists design genetic constructs. The priority was to illustrate the approach by a grammar that could be interpreted in biological terms. As a result, most of the rules recurse by simply duplicating non-terminals, which leads to needless non-determinism and greater complexity in parsing. The grammar could be transformed into an equivalent grammar in a normal form such as the Greibach Normal Form. This transformation would make parsing more natural but would be difficult to read for biologists. Ultimately, it is possible that equivalent grammars in different forms will be used for design and parsing purposes.

Even though it is presented in CFG form, the grammar in Table 2.2 could probably be represented by a simpler regular language. In addition, the grammar does not take advantage of more advanced features of CFG that could be used, for instance, to express long distance interactions between an enhancer and a promoter, a transcription factor and its operator sequences, etc. Natural sequences include complex features such as overlapping genes [144, 154], introns and splicing sites [122, 144] or alternative splicing [122] that are not readily expressed by a CFG but could be expressed by richer Definite Clause Grammars (DCG) or String Variable Grammars [54, 183, 184]. The readily available Prolog programming language is able to parse DCG and this environment has been extensively used to build complex gene grammar and parsers [186, 191]. Future efforts will attempt to use this approach for the verification of the DNA sequences of genetic constructs.

2.4.2 Data model for libraries of genetic parts

Software applications can use syntactic models of genetic constructs to increase the productivity of individual users. Syntactic models could also be used to improve infrastructures serving the entire community. Syntactic categories provide a rigorous foundation to the organization of genetic parts in different categories. The "Transcriptional regulator" category of the Registry contains a large collection of prokaryotic promoters. However, some complex constructs composed of multiple parts (BBa_I13005 or BBa_J24669) are also found in this category even though they would probably fit in a category corresponding to a higher level of abstraction. Similarly, a number of eukaryotic promoters are listed in the Transcriptional regulator category. It might be preferable to have Eukaryotic transcription activators listed in their own category as they are not compatible with other prokaryotic genetic parts. By using syntactic models to develop community infrastructures, it would be possible to verify user submissions and existing content. As artificial gene networks become more complex by combining parts coming from distant organisms [38, 67], a broader syntactic model than the one presented in this paper will help articulate rules of compatibility between parts. Of particular importance is the inclusion of existing knowledge relating to the use of prokaryotic transcription factors in eukaryotes [81, 143, 152].

As syntactic models of genetic constructs become broader, it might become necessary to specify the context in which the construct will be used. The tetracycline repressor has been shown to work in multiple organisms including mammalian cells and some plants [19, 83, 152, 219] but not all plants. In this context, the distinction between prokaryotes and eukaryotes may be not sufficient. The distinction between mammalian cells and plant cells may not be sufficient either as it may be desirable to specify the species in which this transcription factor can be used. Similarly, a number of eukaryotic promoters are tissue-specific whereas the activity of other promoters is not affected by the type of cells in which they are used. Each context will require the development of separate sets of production rules, but some parts should be useable in multiple contexts.

2.4.3 Limitations of syntactic models

The models and tools presented in this article rely on a higher level of abstraction than the DNA sequence. When using a syntactic model to guide the design of a new construct, it is straightforward to translate the description of the construct into a sequence since each genetic part corresponds to a unique sequence. However, verifying the sequence of genetic constructs is more complicated. The very basic lexical analyzer used in the context of this work is too rigid for practical use. The development of a more flexible analyzer capable of handling constructs with legacy sequences interspersed between functional parts will require dedicated efforts. Another limitation of this syntactic model is its purely structural nature. There is no reference to the function of the parts used in the construct. To capture structure-function relationships, it will be necessary to develop a semantic model of genetic constructs that would complement the syntactic model presented here.

2.4.4 Beyond the proof-of-concept

It will take some time after the publication of this proof-of-concept paper to gain a better perspective on the advantages and limitations of this approach. Previously published artificial gene networks appear to have been designed by a labor-intensive and error-prone process. To the best of our knowl-edge, no other framework has been proposed at this time to streamline and formalize this process. We are investing significant efforts in the development of a user-friendly web site allowing biologists to design new constructs from previously defined grammars and parts libraries. Users will also be provided with tools to customize their grammars and parts libraries. Will users having no previous experience with formal languages be comfortable using this approach? Will they use it preferably to design new sequences or to verify sequences designed using other approaches? The analysis of patterns of activity on the Genocad.org web site will, over time, provide the best evaluation of the approach.

2.5 Acknowledgements

We thank two anonymous reviewers for insightful comments on an earlier version of this work and suggestions of future research directions. We are indebted to Otto Folkerts and Rebecca Shelton for their thorough reading of the manuscript. This work was supported with a startup fund from the Virginia Bioinformatics Institute and a fellowship from the Genetics, Bioinformatics, and Computational Biology Program at Virginia Tech.

Conflict of Interest: CG holds an interest in DNA2.0, a gene synthesis company.

2.6 Attribution

Yizhi Cai – Graduate student (Genetics, Bioinformatics and Computational Biology interdisciplinary doctoral program, Virginia Tech), performed the experiments, and wrote part of the paper.

Brian Hartnett – was a lab manager at the synthetic biology group in Virginia Bioinformatics Institute, and contributed in performing the experiments.

Claes Gustafsson – Ph.D. (Department of Molecular Biology, University of Umea, Sweden) currently is the vice president for sales and marketing at DNA 2.0 Corporation, was a collaborator of the synthetic biology group at Virginia Bioinformatics Institute and contributed to this chapter with intensive discussions.

Jean Peccoud – Ph.D (Department of Molecular Biology and Bioinformatics, Universit Joseph Fourier (Grenoble I)) currently an associate professor of Virginia Bioinformatics Institute at Virginia Tech. JP conceived the idea of the paper, performed part of the experiment and wrote the paper.

Chapter 3

Targeted Development of Registries of Biological Parts

Published at: J. Peccoud, M. F. Blauvelt, Y. Cai, K. L. Cooper, O. Crasta, E. C. DeLalla, C. Evans, O. Folkerts, B. M. Lyons, S. P. Mane, R. Shelton, M. A. Sweede, and S. A. Waldon. Targeted development of registries of biological parts. PLoS ONE, 3(7):e2671, Jan 2008.

Authors: Jean Peccoud, Megan F. Blauvelt, Yizhi Cai, Kristal L. Cooper, Oswald Crasta, Emily C. DeLalla, Clive Evans, Otto Folkerts, Blair M. Lyons, Shrinivasrao P. Mane, Rebecca Shelton, Matthew A. Sweede, Sally A. Waldon

Abstract

Background: The design and construction of novel biological systems by combining basic building blocks represents a dominant paradigm in synthetic biology. Creating and maintaining a database of these building blocks is a way to streamline the fabrication of complex constructs. The Registry of Standard Biological Parts (Registry) is the most advanced implementation of this idea.

Methods/Principal Findings: By analyzing inclusion relationships between the sequences of the Registry entries, we build a network that can be related to the Registry abstraction hierarchy. The distribution of entry reuse and complexity was extracted from this network. The collection of clones associated with the database entries was also analyzed. The plasmid inserts were amplified and sequenced. The sequences of 162 inserts could be confirmed experimentally but unexpected discrepancies have also been identified.

Conclusions/Significance: Organizational guidelines are proposed to help design and manage this new type of scientific resources. In particular, it appears necessary to compare the cost of ensuring the integrity of database entries and associated biological samples with their value to the users. The initial strategy that permits including any combination of parts irrespective of its potential value leads to an exponential and economically unsustainable growth that may be detrimental to the quality and long-term value of the resource to its users.

3.1 Introduction

De novo gene synthesis [119, 211, 224] is catalyzing a transition from the *ad-hoc* methods of traditional genetic engineering to the development of industrial-scale fabrication processes enabling users to quickly obtain from commercial vendors genetic constructs that would have been assembled through a custom cloning strategy just a few years ago. Designing a construct for gene synthesis often consists in combining a number of previously defined DNA sequences [215]. The design of an expression cassette in *Escherichia coli* typically includes a promoter, a Ribosome Binding Site (RBS), a coding sequence, and a transcription terminator. These functional blocks are commonly referred to as biological parts or genetic parts. Catalogues of biological parts that are sufficiently well characterized to be used in the design of new genetic constructs can be described in review articles [216], embedded into software applications to design new DNA sequences [215], or made available through a web site [29, 62]. With four years of existence and 4,856 entries in July 2007, the Registry is the largest publicly available library of genetic parts. The Registry goes beyond just cataloguing parts. The parts in the Registry must meet the BioBrick standard, which requires the part sequence to be framed by standard cloning sites called the prefix and suffix. If the part sequences do not contain any of the restriction sequences used by the prefix and suffix, this standardization ensures that it is possible to use a generic cloning process to combine two BioBrick-compliant parts. The process is generic because the restriction enzymes and ligation steps it includes are independent of the sequences of the two parts being combined. This standardized assembly of new genetic constructs derived from standardized parts is therefore complementary to *de novo* gene synthesis since both approaches can be used to fabricate designer DNA sequences. Another benefit from standardizing parts is the physical composition of BioBrick parts. The restriction sites used by the BioBrick standard ensure that the combination of two BioBrick parts results in a new BioBrick part that can be added to the list of parts available for future design projects. The composition of parts leads to distinguishing two categories of parts. Composite parts are parts resulting from the composition of two parts whereas basic parts are parts that cannot be decomposed into smaller parts.

In addition to developing a large catalogue of parts, the Registry has developed a repository of 995 bacterial clones (as of July 2007) corresponding to physical implementations

of entries in the Registry database. The Registry database content and clone collection have been primarily developed by students enrolled in the in the International Genetically Engineered Machine (iGEM) competition [82, 157]. Each year, the iGEM organizers send the entire clone collection to all the teams enrolled in the competition. The teams use this toolkit to implement the designs required for their project. At the end of the summer, the teams contribute back to the Registry new basic parts and new composite parts they have made during the course of their project. This new material is included in the Registry and becomes available to the teams enrolled in the competition the following year. If students enrolled in iGEM still represent the largest group of Registry users, recent publications have demonstrated that this resource can enable the development of more mainstream research projects [3, 35, 89, 127, 195].

We have analyzed the Registry to identify usage patterns that could help design the next generation of infrastructures to host libraries of genetic parts. The analysis consists of two parts. First, the structure of the database itself is considered in terms of the relationships between database entries by examining their published sequences and categorization. The Registry uses two levels of categories to organize its content. Entries of different functional types (promoters, coding sequences, etc.) are regrouped into three classes according to their level of complexity. The simplest entries are found at the bottom of the hierarchy in a class labeled "Parts". Combinations of parts implementing specific functions like inversion of a signal, gene expression cassettes, or reporter genes are found in "Devices". Finally self-contained combinations of devices designed for a particular application are placed under "Systems" [62]. This categorization implements an abstraction hierarchy, an approach commonly used in engineering to manage complex engineering projects by allowing different groups of specialists to work at different levels. Ultimately, engineers with a domain expertise should be able to develop application-specific systems by combining previously characterized devices without having to know more about these devices than their operational characteristics. The second part of our analysis is a comparative analysis of the published sequences of database entries and the experimental sequences of the corresponding clones, which we obtained by sequencing the clones in one distribution of the DNA repository.

While a library of parts as a single centralized community resource has clear benefits, there are still many reasons for organizations or individual investigators to structure their own libraries of parts [91, 164]. These reasons may include the physical or legal availability of a limited set of parts, previous experience with a specific parts list, the use of specific organisms not included in community resources, the inclusion of proprietary parts in the design, and possibly others. Hence, our results have implications beyond the analysis of a specific resource at a particular point in time.

3.2 Results

3.2.1 Analysis of the database content

Since most Registry entries correspond to constructs that have not yet been fabricated, it appeared more interesting to limit the analysis of the database to the 995 entries for which a clone was available. Among these 995 entries, 279 were in the "Parts" layer of the Registry abstraction hierarchy. The remaining 716 entries were categorized in the "devices" and "systems" layers of the hierarchy. In this paper we use parts to refer to entries in the "parts" layer and design for entries categorized in the device or systems layers of the hierarchy.

We derived a network of relationships between entries in the Registry from their published sequence. First, inclusion relation- ships between entries were identified by pair wise comparison of the sequences in the database. Entry A is connected to entry B if the sequence of A includes the sequence of B. In a second step, this directed graph was pruned to eliminate transitive relationships. For instance, if A includes B and B includes C, then a relationship between A and C can be derived from the previous relationships. In this example the inclusion of C within A is pruned from the graph. This operation allowed us to draw a network of 1383 relationships among the 995 entries considered in this analysis (Figure 3.1 and Figure S1). We identified 496 relationships in which the sequences of designs included part sequences. We also found 826 inclusion relationships between design sequences. Since parts correspond to the bottom layer of the abstraction hierarchy, it was expected that there would be few if any connections among entries in this group. However, 49 relationships between parts have also been identified. Even more surprising, 12 relationships indicated that entries in the design group were present in the sequences of parts. These observations appear to be inconsistent with the Registry abstraction hierarchy.

Table 3.1: Joint-distribution of the parts complexity and popularity.

Popularity[1]	Complexity[2]							
	0	1	2	3	4	5	6	Sum
0	154	64	236	39	5	4	-	502
1	65	24	150	12	1	1	2	255
2	30	9	62	10	3	-	-	114
3	21	7	19	1	-	-	-	48
4	8	3	7	-	-	-	-	18
5	6	4	5	-	-	-	-	15
6	3	-	7	-	-	-	-	10
7	3	-	3	1	-	-	-	7

Continued on next page...

Table 3.1 (Continued)

	0	1	2	3	4	5	6	Sum
8	-	-	1	-	-	-	-	1
9	2	-	3	-	-	-	-	5
10	1	-	1	-	-	-	-	2
11	2	-	1	-	-	-	-	3
12	1	-	-	-	-	-	-	1
13	-	-	-	-	-	-	-	0
14	-	1	1	-	-	-	-	2
15	-	-	-	-	-	-	-	0
16	-	-	3	-	-	-	-	3
17	-	1	-	-	-	-	-	1
18	-	-	-	-	-	-	-	0
19	-	-	1	-	-	-	-	1
20	1	-	-	-	-	-	-	1
21	-	-	-	-	-	-	-	0
22	1	-	-	-	-	-	-	1
...	-	-	-	-	-	-	-	0
31	-	-	1	-	-	-	-	1
...	-	-	-	-	-	-	-	0
36	-	-	1	-	-	-	-	1
...	-	-	-	-	-	-	-	0
39	-	-	1	-	-	-	-	1
...	-	-	-	-	-	-	-	0
52	1	-	-	-	-	-	-	1
...	-	-	-	-	-	-	-	0
70	-	-	1	-	-	-	-	1
Sum	299	113	504	63	9	5	2	995

After having identified inclusion relationships within the Registry, we summarized this pruned connection matrix by computing for each entry, the number of other entries directly included in its sequence (a measure of its design complexity) and the number of entries in which its sequence is found (a measure of its popularity). The joint distribution of entry complexity and popularity provides a global perspective on the dynamics of design reuse to build more complex designs (Table 3.1). Entries in the first column (299 entries) are true basic parts while the entries that have never been reused are in the first line (502 entries). If some entries have been used in as many as 70 designs (Table 3.2), 80% have been used less than 3 times. Because indirect relationships have been removed from the pruned interaction network, the complexity axis on Table 3.1 does not refer to the total number of parts

[1]Number of times Registry entries are used in other entries

[2]Number of entries included in an entry sequence

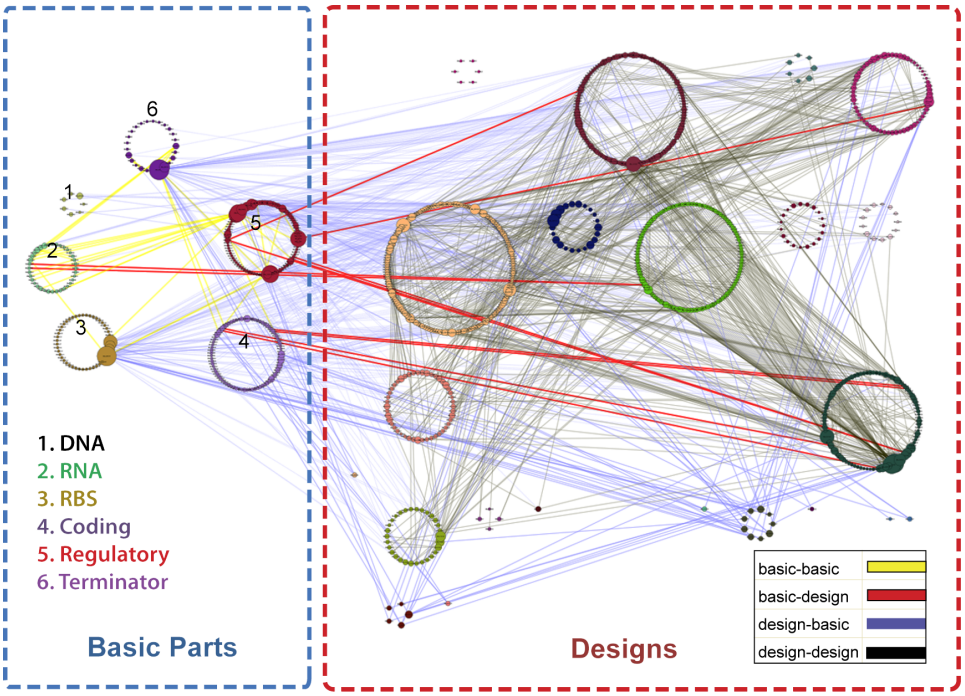


Figure 3.1: Network of inclusion relationships among the Registry entries. Nodes of this network correspond to entries in the Registry. Nodes are grouped in color-coded circles according to the Registry categories. Categories corresponding to parts are within the blue box on the left side of the figure whereas categories corresponding to designs are located within the red box on the right side. The diameter of the nodes corresponds to the node connectivity. The directed edges indicate that the sequence of one entry is included in the sequence of another entry. Edges are color-coded according to the type of relationship. If most of the edges correspond to natural relations (parts included in designs, and designs included in other designs), it is somewhat surprising that parts can include other parts (yellow edges) and it is unclear why some parts would include design in their sequence (red edges). Detailed analysis of individual entries can be conducted using a Cytoscape [194] file (Figure S1). doi:10.1371/journal.pone.0002671.g001

included in the design but it indicates the number of subcomponents an entry is composed of. Approximately 50% of the entries can be broken down into two other entries, which is consistent with a pair wise assembly process. It also indicates that users have recorded most of the construction intermediates. The ideal shape of this joint distribution is not clear except that few entries should be located near the origin. The value of having a lot of parts used very infrequently is questionable, so the weight of the popularity distribution should shift away from 0.

3.2.2 Analysis of the DNA repository

The analysis of the Registry database reveals some of the challenges in implementing the abstraction hierarchy upon which this community resource has been built. However, making the parts physically available adds another level of complexity. We have therefore systematically analyzed the library of plasmids shipped in May 2007 to teams enrolled in the iGEM competition.

The plasmids were distributed lyophilized in four 384-well plates. After suspending the DNA into water, the solutions were quantified using a spectrophotometer and only two wells did not appear to contain any DNA. In order to obtain enough material for DNA sequencing, the inserts were amplified using primers complementary to the standardized prefix and suffix used to clone them into the vector. The products of amplification were analyzed by electrophoresis to select clones suitable for sequencing. In particular, we eliminated 216 clones that did not amplify and 190 clones that resulted in multiple peaks of size greater than 120 bp. The lack of amplification product can either result from a problem with the amplification reaction or indicate the absence of sequences complementary to the primers. The presence of multiple peaks may be caused by primer dimers, non-specific amplification, or the presence of different plasmids in the well. Since parts with sequences shorter than 120 bp can easily be obtained as oligonucleotides, only the 789 clones that generated a single PCR fragment larger than 120 bp were sequenced. Of these 789 sequenced clones, 509 have published sequences that were used for subsequent analyses.

To get a global measure of the match between published and physical sequences, we plotted the length of the published sequence against the size of the PCR fragment for the 509 sequenced plasmids having a documented sequence. On Figure 3.2, 76 outliers were visually identified. The rest of the lengths remained close to the expected lengths. Yet, only 285 data points had less than a 10% difference between the two sequence lengths. The differences in size distribution between measured and expected lengths appears wider than 5%, the previously reported experimental error affecting the determination of fragment size by the microfluidic system used for this project [103]. We have not investigated all discrepancies, amplification failures, or multiplicity of amplification products. This would require a

Table 3.2: The Registry most popular parts.

ID	Category	N	Description	Parts included
BBa_B0015	Terminator	70	Double terminator consisting of BBa_B0010 and BBa_B0012	BBa_B0010, BBa_B0012
BBa_B0034	RBS	52	RBS based on Elowitz repressilator	-
BBa_E0430	Reporter	39	Standard YFP Output Device -LVA tag	BBa_E0130, BBa_S01014
BBa_E0432	Reporter	36	EYFP (RBS+ LVA+ TERM) (B0034.E0032.B0015)	BBa_I9045, BBa_S01638
BBa_J13002	Regulatory	31	TetR repressed POPS/RIPS generator	BBa_B0034, BBa_R0040
BBa_R0040	Regulatory	22	TetR repressible promoter	-
BBa_R0011	Regulatory	20	Promoter (lacI regulated, lambda pL hybrid)	-
BBa_I0500	Regulatory	19	Inducible pBad/araC	BBa_I13458, BBa_R0080
BBa_B0030	RBS	17	Strong RBS based on Ron Weiss thesis	BBa_B0034
BBa_I13507	Composite	16	Screening plasmid intermediate	BBa_I13501, BBa_I13502
BBa_I13504	Reporter	16	Screening plasmid intermediate	BBa_I13401, BBa_I13500
BBa_S03155	Intermediate	16	Terminators B0010+B0012+promoter R0040	BBa_B0015, BBa_R0040
BBa_J04500	Intermediate	14	IPTG inducible promoter with RBS	BBa_B0034, BBa_R0010
BBa_Q04121	Inverter	14	LacI QPI with strong RBS, hybrid promoter	BBa_P0412
BBa_R0062	Regulatory	12	Promoter activated by LuxR in concert with HSL	-
BBa_E0420	Reporter	11	Standard CFP output device w/o LVA tag	BBa_B0015, BBa_S01022
BBa_R0051	Regulatory	11	promoter (lambda cI regulated)	-
BBa_B0032	RBS	11	Weak1 RBS based on Ron Weiss thesis	-
BBa_Q04400	Inverter	10	TetR QPI with strong RBS	BBa_P0440, BBa_S03155
BBa_B0031	RBS	10	RBS.2 (weak) derivative of BBa_0030	-

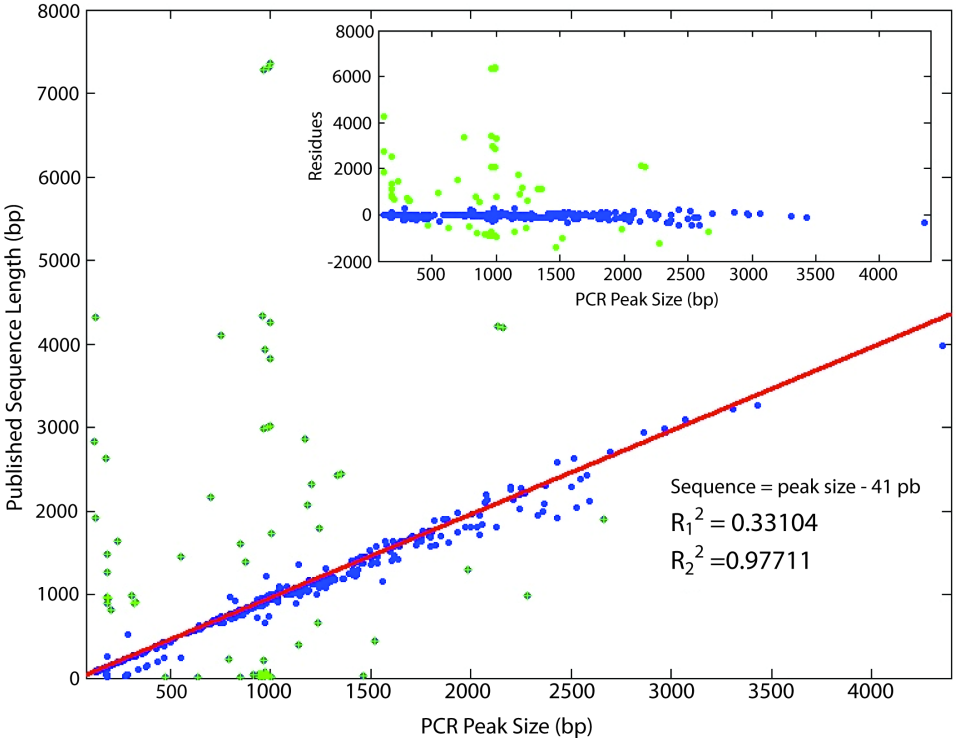


Figure 3.2: Comparison of the Registry published sequences with the size of the PCR amplification products. This plot is limited to the clones that generated a single PCR fragment greater than 120 bp. Theoretically, the size of the PCR fragment is 41 pb longer than the length of the published sequence because of the presence of the PCR primer sequences in the amplification product (n = 509). When all data points were used in the linear regression, the fit led to a coefficient of correlation $R_1^2 = 0.33$. Based on previously reported experimental error affecting fragment size determination [103], 76 outliers were eliminated manually (green points) leading to a greatly improved $R_2^2 = 0.98$. doi:10.1371/journal.pone.0002671.g002

systematic curation of the published sequences, as well as individual PCR troubleshooting for each clone, which were beyond the scope of this project.

The next step of our analysis was to look at sequences individually. Of the 789 clones sequenced, 591 could be assembled in contigs. The length of assembled sequences ranges from 166 bp to 1897 bp. Some of the inserts that could not be assembled may be too long to achieve a significant overlap between the two sequence files starting from both extremities of the insert. Assembling these clones would require additional sequencing runs utilizing clone-specific primers. Out of the 591 clones assembled, only 354 could be associated with a Registry entry as the other clones were undocumented in the particular distribution of the Registry used in this project. The assembled sequences were aligned with the published sequence using BLAST [5]. Out of 354 assembled sequences for which published sequences were available, 334 produced alignments with their published sequence and the complete results of this alignment analysis are reported in Table S1. This spreadsheet was used to identify clones for which the assembled sequence confirms the published sequence. Since the assembled sequence can include the primer sequences, the assembled sequence should not be longer than the published sequence plus the combined length of the two primers (41 bp). Since the primers used in this project are adjacent to the sequence being verified, the first and last 10 to 25 bases of the insert can be difficult to read. As a result, the assembled sequence may be up to 50 bp shorter than the published sequence. These two criteria led to the selection of 221 clones for which -41$\leq$length (published sequence) - length(assembled sequence)$\leq$50. In the second step of sequence analysis, we want to ensure that the alignment of the assembled and published sequences covers most of the shorter of the two sequences. In this second step, from the 221 clones meeting the assembled sequence length criteria, we selected 177 clones for which the alignment length is at least 99% of the length of the smaller of the two sequences being compared. After these two rounds of selections, the percentage of identity of the assembled and published sequence was always superior or equal to 97% and greater or equal to 99% for 162 of the 177 clones. It is obvious that different choices of parameters would lead to larger or smaller number of clones with a confirmed published sequence.

Just like in the case of PCR results, a systematic control of the published sequences could improve the clone confirmation statistics. It is quite possible that for a number of these clones the biological material is correct but their published sequence may be inaccurate. Additional sequencing runs starting from within the insert sequences would also increase the number of clones with long inserts that could be confirmed.

3.3 Discussion

3.3.1 A global analysis of the Registry

This analysis of the DNA library provides no more than a snapshot of one distribution of the Registry clone collection. Amplification and sequencing problems could result from technical problems during the experiments described in this paper just as they could indicate problems with the biological samples themselves. For instance, samples that lead to multiple amplification products could have been delivered contaminated, could have been contaminated during one of the steps described in this report, or could simply result from mispriming. In order to control the experiments, it would be necessary to repeat all the operations starting from a new series of samples. Unfortunately the lack of unique clone identifier makes such control problematic. The different distributions of the DNA repository do not share a common key necessary to relate one distribution to another. The data set described in this article is specific to the 2007 distribution of the DNA repository. Our results are valuable to understand global issues associated with the design, development, and management of a registry of biological parts but they would need data describing how different distributions relate to each other to be used for controlling the quality of specific clones in the Registry collection.

The high-level analysis of the Registry database led to the identification of several non-trivial issues that need to be addressed. The implementation of a workable abstraction hierarchy remains problematic. A single category of parts (DNA) appears to be exclusively composed of basic building blocks. However, our sequence analysis has revealed elements categorized as parts within the Registry that include other parts, indicating that not all clones categorized as parts have an atomic nature. Some part sequences even include designs, a higher level in the abstraction hierarchy. These observations result from the lack of consensus in the community on how biological parts should be defined. Nothing illustrates this confusion better than the complex architecture of promoters [46, 150]. On the one hand, promoters are generally considered as parts but on the other hand they have well characterized domains that can be associated with specific functions. When developing an abstraction hierarchy, should promoters lie at its bottom and be considered as atomic parts or should they be considered as composite parts composed of multiple functional domains? The case of genes is not simpler as proteins are also composed of multiple functional domains[76]. A complete access to the Registry database would have made it possible to investigate questions that could not be addressed using the partial dump of the database content used in this analysis. For instance, parts have a usefulness attribute used to report if a part works, works with issues, or does not work as anticipated. It would be interesting to relate the parts popularity to the usefulness status of a part as one would imagine that the most popular parts are reported as working. The structure of composite parts is also described as the series of basic parts they are composed of. Comparing the sequence and structure of composite parts could help investigate a number of interesting questions. Figure 3.1 reflects the laborious efforts of

the synthetic biology community to develop and implement the new theoretical framework it needs to support its scientific vision.

3.3.2 Organizational guidelines

Results presented in this report lead to a number of organizational guidelines that could help design or manage registries of biological parts.

The published DNA sequence of entries should be carefully curated. Lack of published sequences or incorrect ones hamper the quality control of the associated clones. It is important to clearly identify basic parts of a registry as they generate the rest of the database. Basic parts should be linked and compared to entries in other sequence databases and peer-reviewed publications [216]. Basic parts that have not been completely annotated should be flagged so that people considering using them may proceed with caution. The sequence redundancy of the basic part set is a difficult problem. Theoretically, a set of basic parts could be atomic in the sense that it generates all other entries in a registry. However, this approach may not always be practical. If certain projects need to identify several parts in a promoter sequence, this level of granularity may be excessive for other projects. The same argument can apply at higher levels of organization. For the same reasons, nothing prevents the definition of complete gene expression cassettes and other devices such as switches, inverters, etc. as basic parts. However, this option does not seem desirable as it would be inconsistent with the engineering vision of building complex systems from a limited numbers of building blocks.

The integrity of the sequence of composite parts is even more difficult to ensure. There could either be a static or dynamic link between the sequence of a composite part and the sequences of the basic parts it is composed of. In the first case, the sequence of a composite part is automatically derived from the sequence of its components when the composite part is created but future changes to the sequences of its basic components do not propagate to the composite part sequence. If such a policy is enforced, discrepancies between the composite part sequence and the sequences of its basic components can develop over time. It is desirable that such discrepancies be identified. In the case of a dynamic link, any change in the sequence of a basic part propagates to all composite parts using this basic part. The integrity of the composite part sequence is then always preserved but different versions of the composite parts that are automatically generated by this process may be very confusing to the users.

Clones in the DNA repository associated to a parts registry need to be uniquely identified independently of the parts in the registry. Parts numbers are not good identifiers of clones as many clones correspond to the same part in different plasmids or different bacterial strains. A clone key is necessary to compare data collected on different distributions of the same clone and therefore implement quality control procedures. A standardized quality control

process should be specified to ensure the integrity of the clone collection.

3.3.3 Targeted development of registries of parts

The idea of developing collections of standardized parts is a transformative idea in biology [193]. After a few years of a large scale experiment, it becomes apparent that developing and managing this new type of resource for synthetic biology raises a number of original questions. Specialized registries built on compatible standards are being developed by various groups that will experiment with different user interfaces, workflows, and modes of user interaction. These initiatives along with future developments of the original Registry will provide elements of solutions to these new questions.

It will be particularly interesting to see if different registries will adopt different editorial policies. The cost of maintaining a parts registry depends on its size as each entry needs to be properly documented and each clone needs to be verified. Parts registries are different from traditional collections used in biological research as any combination of parts in the registry can also be integrated in the registry. A small number of basic parts can therefore generate a potentially infinite collection of clones. Initially, it may be attractive to record any combination of parts without any consideration of its potential value, but this approach now appears unsustainable [91]. At some point, users and managers of a parts registry will need to analyze the allocation of their resources. Table 3.2 and similar analysis on other registries can help identify entries that are the most valuable to the users and least expensive to maintain. For instance, basic parts deserve special attention because they enable the development of new designs and errors affecting basic parts can propagate to the entire resource. This contrasts with the case of a large and specialized construct including multiple genes that would be expensive to control and might have a low probability of reuse. Even though it would be desirable to also include such construct in the database and clone collection, if finite resources require choosing between recording a few new basic parts with a broad reuse potential and a specialized and expensive part, it is likely that resources will be preferably allocated to adding basic parts. Similarly, including a switch that could be used in developing a number of applications will probably be deemed more valuable than the construction intermediates that were generated during its assembly. Managers of parts registries need to articulate editorial policies to set criteria for including new entries in their database so that resources can be targeted to developing content maximizing the benefits to their users.

Recognizing that repositories of biological parts are an essential component of the upcoming integrated development environments for synthetic biology [6, 8, 29] may help target the development of their content. In order to support this integration it is necessary to specify a minimal data model allowing programmatic access to the registry databases from multiple

client applications. A draft of such a data model is described in Text S1. Structured methods for designing synthetic genetic systems will provide a theoretical framework that will guide the development of user interfaces helping users combine basic parts into complex designs. Alternative solutions to the organization of parts in categories or the mechanism to define composite parts will probably be proposed. In this context, recent initiatives to organize forums aiming at defining technical standards for biological parts appear very timely and laudable.

3.4 Materials and Methods

The plasmids were resuspended in $30\,\mu l$ of nuclease free water (Ambion) at 4℃ overnight. They were quantified using the Nanodrop spectrophotometer. $20\,ng$ of Plasmid DNA was used in the PCR amplification of the plasmid inserts, using Qiagen's Taq PCR master mix kit, and $2\,\mu M$ primers forward and reverse primers at $100\,\mu l$ reaction volume. The forward primer was homologous to the BioBrick prefix (5' - GAA TTC GCG GCC GCT TCT AG - 3') whereas the reverse primer was complementary to the suffix sequence (5' - CTG CAG CGG CCG CTA CTA GTA - 3'). PCR conditions: 94℃ 45 sec, (94℃ 30 sec, 55℃ 45 sec, 72℃ 45 sec) for 24 cycles, 72℃ 5 minutes, 4℃ hold.

The PCR product was purified using Qiagen's QIAquick PCR purification kit, resuspended in $25\,\mu l$ of nuclease free water, and quality controlled using the Agilent Bioanalyser DNA 7500 assay. The amplified products were quantified and diluted to $10\,ng$ per ml. The PCR product and corresponding primers were submitted to the VBI Core Laboratory for Sanger sequencing using the primers used in the amplification step. Sequencing conditions: $400\,ng$ template DNA, $3.2\,pmol$ primer, $2.5\,ul$ BigDye Terminator mix v3.1, water to a total volume $15\,\mu l$.

Base calling and quality control of sequence chromatograms was done by PHRED [63, 64]. The sequences were assembled using CAP3 [96] with default options except for minimum overlap size of 21 bp. The assembled sequences were aligned with their respective published sequences using BLAST [5] with default parameters.

3.5 Supporting Information

Figure S1 Cytoscape file used to generate Figure 3.1. Can be used to interactively explore the network of relationships within the Registry
Found at: doi:10.1371/journal.pone.0002671.s001 (0.10 MB ZIP)

Table S1 Blast analysis of the clones assembled sequences against the published sequence.
Found at: doi:10.1371/journal.pone.0002671.s002 (0.14 MB XLS)

Text S1 Describes the supporting database and its data model. Also describes the other files included in the supplement.
Found at: doi:10.1371/journal.pone.0002671.s003 (0.12 MB PDF)

3.6 Acknowledgments

We are indebted to Randy Rettberg, Tom Knight, Mackenzie Cowell, and Meagan Lizarazo for providing the collection of clones and a text version of the Registry database used in this report. Michael Czar's careful reading of the manuscript and valuable suggestions helped improve this article.

3.7 Author Contributions

Conceived and designed the experiments: CE OF JP. Performed the experiments: CE MFB KLC ECD BML MAS. Analyzed the data: OC SPM JP YC RS SAW. Wrote the paper: OC CE OF SPM JP YC RS SAW.

3.8 Attribution

Jean Peccoud – Ph.D (Department of Molecular Biology and Bioinformatics, Universit Joseph Fourier (Grenoble I)) currently an associate professor of Virginia Bioinformatics Institute at Virginia Tech. JP conceived and designed the experiments, and wrote part of the paper.

Megan F. Blauvelt - was a research associate in the core lab of Virginia Bioinformatics Institute and currently and senior laboratory technologist at RedPath Integrated Pathology Corporation, and contributed to this chapter in terms of performing the experiments.

Yizhi Cai - Graduate student (Genetics, Bioinformatics and Computational Biology interdisciplinary doctoral program, Virginia Tech), analyzed the data, and wrote part of the paper.

Kristal L. Cooper – Genomics Specialist at Virginia Bioinformatics Institute at Virginia Tech, performed the experiments.

Oswald Crasta – Ph.D. (Department of Molecular Genetics, Texas Tech University) cur-

rently Associate Director at Chromatin, Inc was a project director at Virginia Bioinformatics Institute at Virginia Tech and contributed to this chapter in terms of conceiving and designing the experiments, and writing part of the paper.

Emily C. DeLalla – was an undergraduate student (Department of Biology, Virginia Tech), performed the experiments.

Clive Evans – is the director of the core lab at Virginia Bioinformatics at Virginia Tech, contributed to the chapter in terms of conceiving and designing the experiments, performing the experiments and writing part of the paper.

Otto Folkerts – Ph.D. (Department of Plant molecular biology, Cornell University) currently the director of Transgenic Programs at Chromatin, Inc, was associate director technology development at Virginia Bioinformatics Institute at Virginia Tech, conceived and designed the experiments and wrote part of the paper.

Blair M. Lyons – was an undergraduate (Department of Biochemistry, Virginia Tech) currently a master student (Department of Biomedical Visualization, University of Illinois at Chicago), performed the experiments.

Shrinivasrao P. Mane – Ph.D. (Department of Plant Physiology, Virginia Tech) currently a computational biologist at Virginia Bioinformatics Institute at Virginia Tech, analyzed the data and wrote part of the paper.

Rebecca Shelton – was a graduate student (Department of Electrical Engineering, Virginia Tech), analyzed the data and wrote part of the paper.

Matthew A. Sweede – was an undergraduate student (Department of Biochemistry, Virginia Tech) currently a graduate student (Department of Molecular Cancer Biology, Duke University), performed the experiments.

Sally A. Waldon – was a database specialist at Virginia Bioinformatics Institute at Virginia Tech, analyzed the data and wrote part of the paper.

Chapter 4

Writing DNA with GenoCAD™

Published at: M. J. Czar, Y. Cai, and J. Peccoud. Writing DNA with genocad. Nucleic Acids Res, 37(Web Server issue):W407, Jul 2009.

Authors: Michael J. Czar, Yizhi Cai and Jean Peccoud

Abstract Chemical synthesis of custom DNA made to order calls for software streamlining the design of synthetic DNA sequences. GenoCAD™(`www.genocad.org`) is a free web-based application to design protein expression vectors, artificial gene networks and other genetic constructs composed of multiple functional blocks called genetic parts. By capturing design strategies in grammatical models of DNA sequences, GenoCAD guides the user through the design process. By successively clicking on icons representing structural features or actual genetic parts, complex constructs composed of dozens of functional blocks can be designed in a matter of minutes. GenoCAD automatically derives the construct sequence from its comprehensive libraries of genetic parts. Upon completion of the design process, users can down- load the sequence for synthesis or further analysis. Users who elect to create a personal account on the system can customize their workspace by creating their own parts libraries, adding new parts to the libraries, or reusing designs to quickly generate sets of related constructs.

4.1 Introduction

In order to fully reap the potential benefits of *de novo* chemical gene synthesis [47] it has become necessary to develop tools and methodologies to streamline the design of custom DNA sequences [80]. Protein expression for structural studies [44], functional genomics [74, 99], metabolic engineering [108, 125], or gene expression studies [27, 46, 87, 150] are only some

of the numerous possible applications of this emerging technology. Beyond small scale genetic constructs encompassing no more than a few interacting genes, it becomes possible to reengineer viral [34, 35, 42, 213], bacterial [75], and even eukaryotic [56] genomes. While the number of users of this technology increases, so does the need to streamline the design of synthetic DNA sequences. GenoCAD is a web-based application filling this need by providing users with an integrated graphical development environment that no other software provides.

GenoCAD's design philosophy derives from the notion of genetic parts, which was first articulated to analyze genomics data [10]. Thinking of genetic systems as composed of parts, each with its own function and characteristics, is akin to the way parts are described and used in various engineering fields. Designing complex systems through a bottom up integration of components is a dominant paradigm in engineering. It was therefore natural that engineers approaching DNA as an engineering substrate, rather than a natural macromolecule, used the notion of biological parts as building blocks [62, 85]. For instance, promoters, ribosome-binding sites (RBS), genes and terminators are all categories of parts that are needed for designing complex prokaryotic genetic constructs such as switches [9, 72, 102] and oscillators [9, 61, 205]. One could argue that systematic efforts to decompose biological sequences into functional modules that can be recombined to meet user-defined specifications is one of the most distinctive features of synthetic biology compared to more traditional uses of recombinant DNA technologies [12, 55, 80, 137].

GenoCAD facilitates the design of artificial DNA sequences in three ways. First, GenoCAD includes a flexible system to manage libraries of public and user-defined genetic parts. Second, GenoCAD relies on formal design strategies to guide both novice and experienced users in the design of structurally valid constructs for various bio- logical applications. Finally, GenoCAD's sophisticated data model enables individual users and research groups to customize their workspace to their specific needs.

4.2 Flexible Management of Genetic Parts Libraries

Nothing better attests the benefits of a parts-based approach to the design of genetic constructs than the success of the Registry of Standard Biological Parts (www.partsregistry.org). By defining the BioBrickTM standard allowing the composition of parts and implementing mechanisms to share parts, the Registry has been critical in fostering the development of a vibrant synthetic biology community [47, 62, 85, 155]. We recently analyzed the content of the Registry database and the associated collection of clones to better understand how the successes and limitations of this pioneering experiment could guide the development of a second generation of registries of biological parts [155]. GenoCAD attempts to refine some of the concepts upon which the Registry was developed.

By assuming that genetic designs can be synthesized, GenoCAD eliminates the need for standardizing the means by which parts are connected. It also eliminates the need to develop a collection of bacterial clones to manage the physical implementation of the parts. Our analysis also stressed the importance of basic parts used to generate new combinations of parts with specific functions. Ensuring the accuracy of the sequence and annotation of the basic parts is essential since inaccuracies at this level may affect numerous designs. As a result of this observation, GenoCAD parts libraries are exclusively composed of basic parts while sequences composed of multiple parts are called designs. The libraries of parts available to all GenoCAD users are limited to sequences used in peer-reviewed publications or commercial vectors. Parts are curated by a small number of experts according to a process that will be described in a future publication.

Categorizing parts into functional groups has also proved challenging as the number and diversity of parts increases. It would be, for instance, questionable to record bacterial and eukaryotic promoters in the same group. Developing a more granular categorization system may lead to an exponential growth of categories that would prove cumbersome to navigate. GenoCAD overcomes this challenge by relying on the notion of grammar [29]. A grammar is composed of rules describing the structure of DNA sequences. One of the rules of the *Escherichia coli* Expression Grammar is $CAS \rightarrow PRO\,, CIS\,, TER$ which reads: an expression cassette is composed of a promoter, a cistron and a terminator. Another rule of the same grammar is $CIS \rightarrow RBS\,, GEN$ (a cistron is composed of a RBS and a gene). The two rules can be used successively to create a basic expression cassette PRO, RBS, GEN, TER. Different grammars can be developed for different applications and each grammar has its own parts categorization hierarchy. This approach ensures, for instance, that parts suitable for designing constructs for specific organisms like *E. coli* or yeast can easily be identified. It also enables the development of grammars for specific applications like protein production, homologous recombination in yeast, etc. Instead of attempting to develop a universal parts categorization system, GenoCAD provides a generic framework for the development of smaller more manage- able application-specific parts-libraries. The 'Parts' tab of the GenoCAD user interface provides a parts library browser (Figure 4.1).

4.3 Point-And-Click Design of Genetic Constructs

In addition to providing a hierarchy of categories, gram- mars include sets of rewriting rules that formalize design strategies for various types of genetic constructs [29, 80]. The design feature of GenoCAD embeds the grammars in a graphical user interface that leads users through the design workflow formalized in the grammar. Grammars usually prompt users to begin by choosing high-level structures of their system and systematically decomposing them into individual part categories. The last step of the design process consists in selecting actual parts corresponding to specific DNA sequences (Figure 4.2). When starting from one

of the public design templates, users can quickly design constructs by simply selecting parts in a parts library instead of going through the entire design process described below.

Here, we use the design of a bistable genetic switch to illustrate GenoCAD's workflow [72]. Selection of a gram- mar and an associated parts library (Figure 4.2) is the first step of this process. By selecting the grammar the user defines the type of construct that is possible, and by selecting the library they define the set of parts available to complete the design. Determining the high level structure of a functional genetic system *de novo* could potentially be confusing to users that do not have an intimate knowledge of the role of each part in the regulation of gene expression. However, each grammar behind GenoCAD provides the design strategy for specific types of constructs. GenoCAD's default grammar, '*E.coli* Expression Grammar', is suitable for the design of prokaryotic expression constructs. Additional grammars will be added for other applications.

The toggle switch construct was designed in nine steps as shown in Figure 4.2. The history pane (left side) allows users to review their work at any stage of the design process by clicking on that step. Users may click in reverse or in forward steps, and they are able to redesign from any point if they wish. Figure 4.3 shows a slightly condensed version of the design process of the toggle switch construct. Displayed below each part of a design are the options available to the user to transform the design. Choices in gray correspond to transformations that affect the design structure while options in white correspond to the selection of a specific part. Structural trans- formations are either part categories for which a specific part selection can be made or a higher level feature that must be decomposed to features lower in the abstraction hierarchy before the design can be finalized. An example of a high level part is the cistron (CIS in Figure 4.3), which is transformed into a RBS and gene (GEN). For each part category users have a choice of one or more specific sequences such as a specific RBS or gene. A mouse-over feature in the interface provides more information about the available choices. For example, the name of choice 04 for the promoter (PRO) category is displayed through this mechanism.

A design is complete when specific sequences have been selected for each of its structural features. It is then possible to click on the 'Download Sequence' button to export the construct sequence as a text file that can be imported into software to design oligos for gene synthesis [94, 104, 168, 215]. Alternatively the sequence synthesis can be ordered from a fast growing number of vendors providing contract gene synthesis services [47].

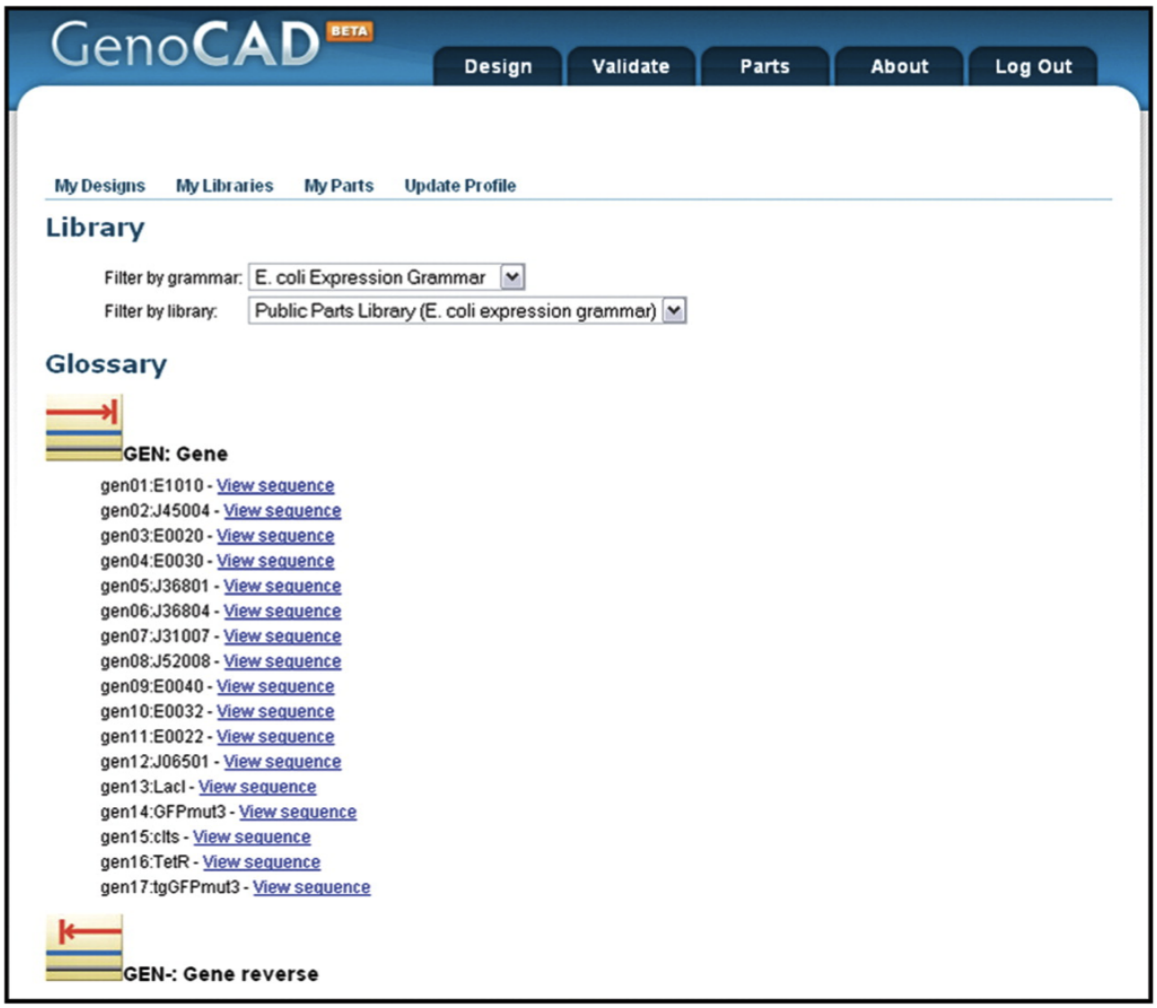


Figure 4.1: The GenoCAD parts library browser (used with permission of J. Peccoud, 2010). Parts are associated with individual libraries, each of which is associated with a specific grammar. Users select which parts library they view through choice of a grammar and specific library in drop down boxes on the page. The part category 'Gene' is displayed in this figure along with the icon that represents genes in the designs. By clicking on the icon, the list of genes expands, allowing the user to see the available choices in the library. Selecting the link to 'View Sequence' for any part opens a small window containing the sequence of the individual part.

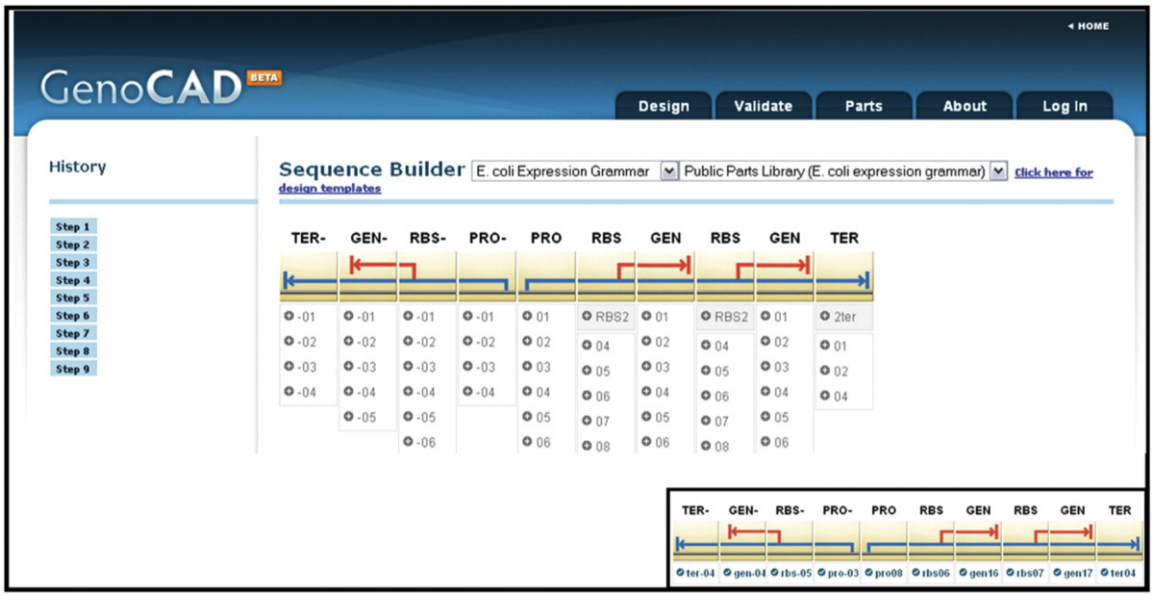


Figure 4.2: The design interface showing the structure of a genetic toggle switch (used with permission of J. Peccoud, 2010). The interface has drop down boxes at the top to select the grammar and parts library that will be used in the design. The history panel allows users to select one of the steps in the design process and see the structure of the design at that step. Users are permitted to go back to any step and redesign from that point. The design is presented in the main panel of the page, and icons for each part and the abbreviated parts categories are shown at the top of the design. Choices for each part are shown underneath the part icon. The inset shows the final design for this construct after specific choices (terminals) are selected for each part category.

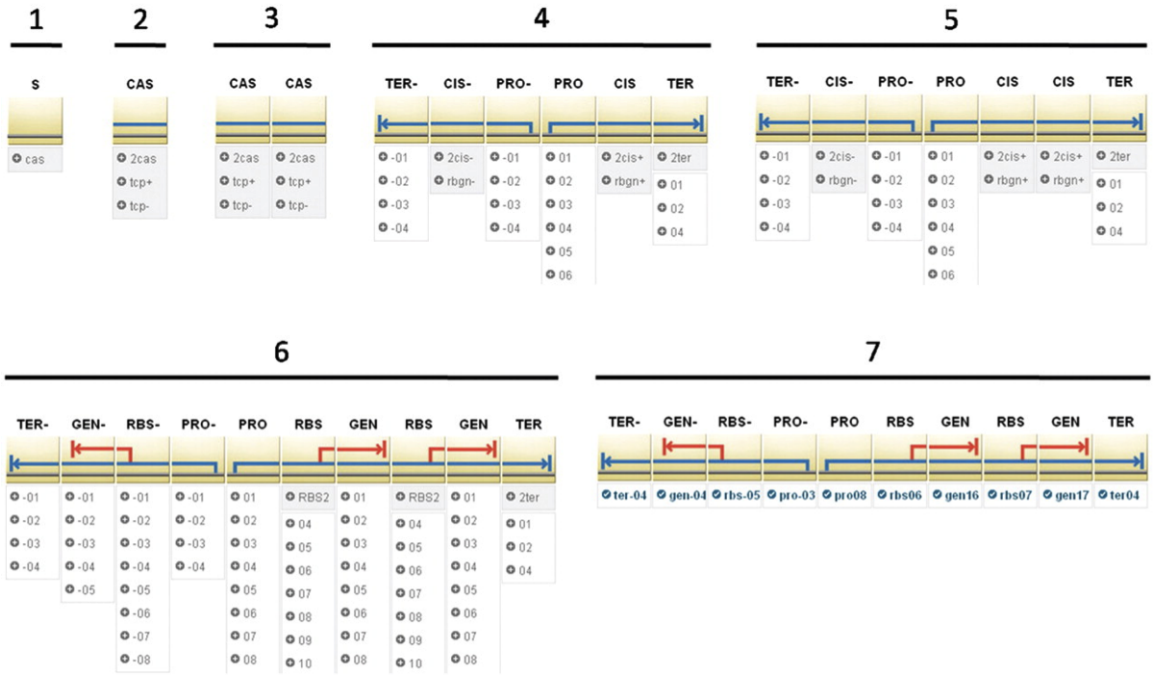


Figure 4.3: A progression through the design of a bistable toggle switch (used with permission of J. Peccoud, 2010). The starting symbol, S, (1) is where each design begins, and it is transformed into a transcription cassette, CAS, (2). Since the toggle switch contains two transcription cassettes, the single cassette is doubled (3). The design we are following has the cassettes oriented in opposite directions, and we achieve this by transforming the left cassette to the tpc- option, and the right to the tcp+ option (4), which contain a promoter, cistron, and terminator, but in opposite orientations. The right cassette is meant to express a transcriptional repressor and reporter gene in a bicistronic manner, so the cistron is doubled by selecting the 2cis+ option (5). Each cistron is then decomposed to a RBS and gene (6), with the RBS and gene in the reverse orientation in the left cassette. Selection of the specific promoters, RBSs, genes, and terminators produces a final construct that is associated with a DNA sequence (7).

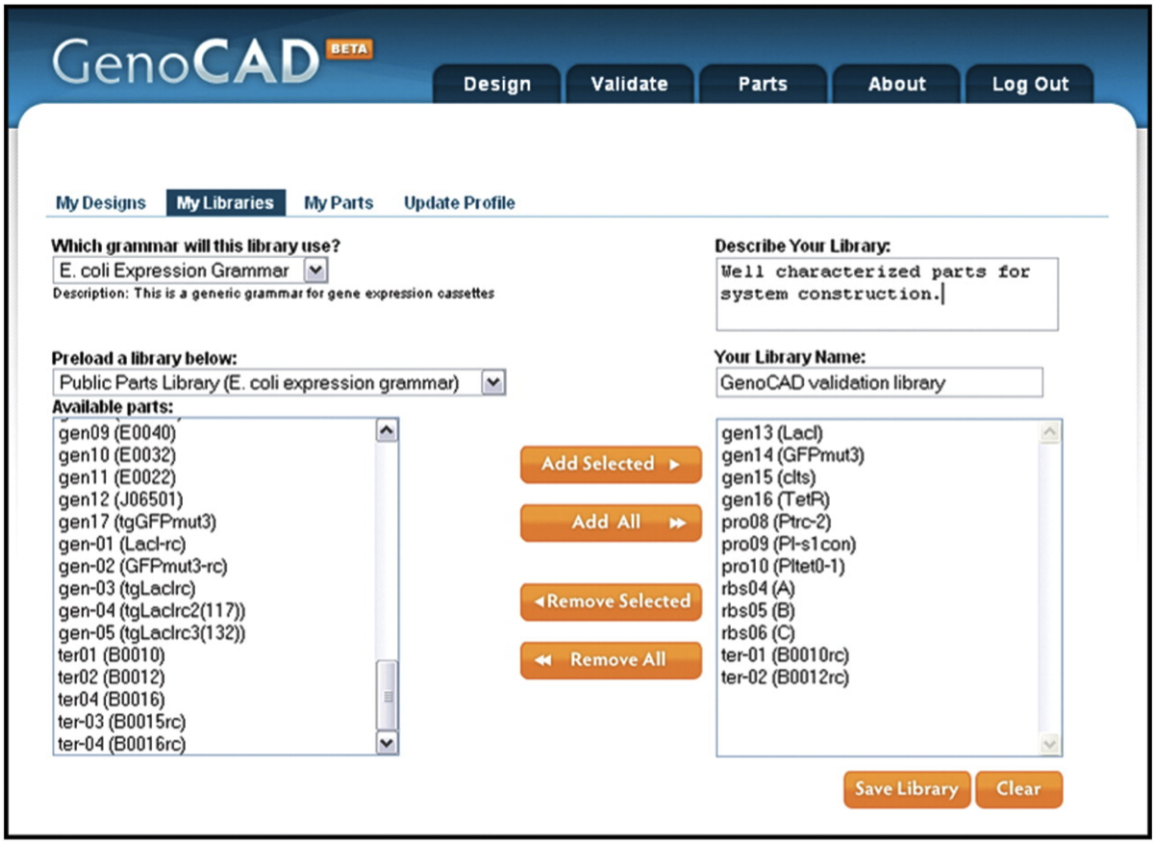


Figure 4.4: Creating a custom parts library. Users who create an account at the website are able to create their own parts libraries, and are then able to add custom parts to these libraries (used with permission of J. Peccoud, 2010). Through the library creation interface, users select the grammar that their library will belong to, provide a name for their library, and can enter a description. Parts can be added to a new library from other libraries of the same grammar, which are loaded in the lower left box on the web page. All parts or a select subset of parts, from the existing library can be copied into or removed from the new library using the orange add and 'remove' buttons.

4.4 Custom User Workspace

Parts available to all users in the public parts library have been derived from peer-reviewed publications and the documentation of commercial vectors. Registered users are able to create both their own parts library and their own parts. To create a new parts library, the user selects the 'My Libraries' link on the 'Design' or 'Parts' tab. When the user selects the link to create a new library they are presented with a user-friendly interface to do so (Figure 4.4). Since parts libraries are associated with specific grammars, the user must select the grammar to which the new library will be linked. Parts can then be copied from other libraries linked to the same grammar by first pre- loading that library at the lower left of the interface. Parts can be copied into and removed from the new library with the 'add' and 'remove' buttons. Once a user has created their own library, they are then able to create new parts and associate them with that library. In Figure 4.5, the part creation page allows the user to specify a grammar, select which part category it belongs to, and enter the part name, sequence and description. One or more user libraries associated with the grammar must be selected at the bottom of the page before the part can be saved into specific libraries.

Users can also save their designs in their workspace. Designs can be saved and named at any stage of the design process. The 'My Designs' page provides links to delete and clone (i.e. copy) previously saved designs. By saving a design prior to selecting parts, users can quickly clone a design template into multiple variants without having to go through the entire design process for each of them.

4.5 Implementation and Data Model

The GenoCAD website is written in a combination of PHP and JavaScript and runs on an Apache server. The MySQL database is on a different server, and both servers use the Linux operating system. The validation page relies on a custom parser developed in C++.

The data model for GenoCAD is summarized on Figure 4.6. Each design is associated with a specific parts library which, in turn, is linked to a specific grammar. Multiple public and user-defined libraries can be associated with each grammar and multiple designs can be associated with a specific library. Parts defined by users need to be associated with one of the user's parts libraries.

This simple data model has several limitations. Since many parts such as coding sequences can be used in different organisms, it would be desirable to replace the current hierarchical data model with a more refined model allowing the same part to be used in multiple libraries and grammars. It would also be desirable to define parts corresponding to coding regions

Figure 4.5: Interface to add a new part. Users that have created a custom library are able to add and save parts that can be used in their designs (used with permission of J. Peccoud, 2010). The categories of parts permissible in designs are defined in the grammar, so the grammar must be chosen first through a drop down menu. A second drop down, then allows users to choose which part category the new part will belong to; in this case a new terminator is being created. The sequence and description of the part are entered in text boxes, and the library(s) to which the part will be added must be checked.

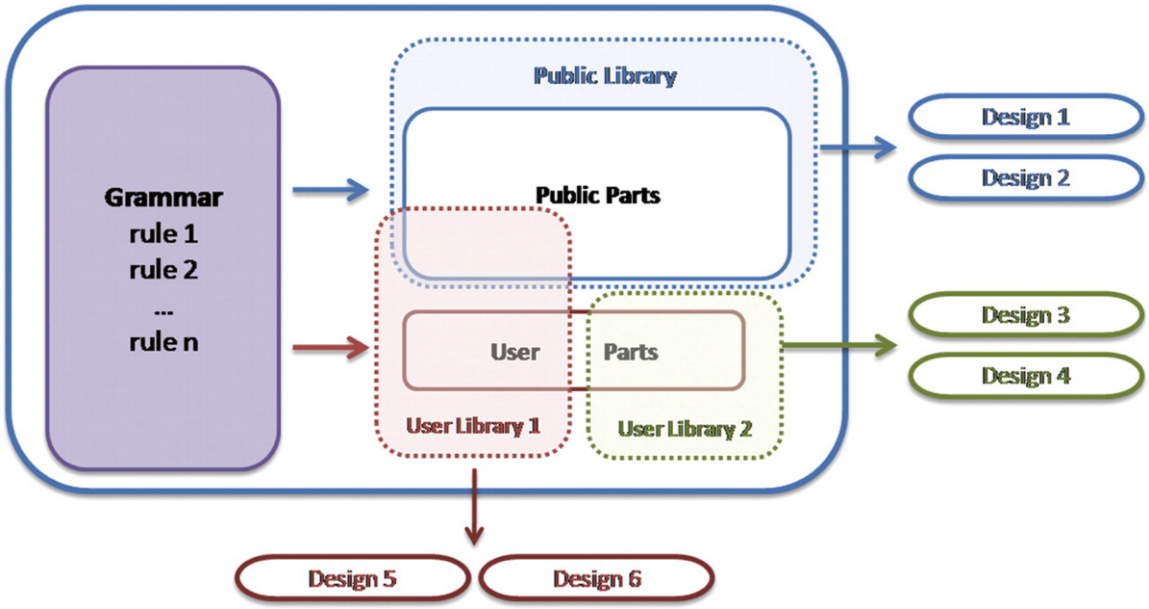


Figure 4.6: GenoCAD data model. Each grammar encompasses a set of rules by which constructs can be designed. The grammar also defines the categories of parts that are available to design the constructs. For each grammar there is a collection of public parts (solid, blue rectangle), which constitute a publicly available parts library (dashed, blue rectangle). 'User Libraries' can be created from any subset of the public parts, and this library can be supplemented with user-created parts (solid, red rectangle). Two user libraries (dashed, red and dashed, green rectangles) are shown here that contain different subsets of public and user-created parts. User library 2 contains all user-created parts. When a design is created, all the parts to complete the design must be contained within a single library.

by their amino-acid sequences instead of being limited to a DNA sequence with codons that are optimal for expression in a specific organism.

Additional grammars will be defined for the use of specific applications or for applications relevant to specific organisms in collaboration with organism or domain experts. Defining parts categories and design strategies suitable for a particular application will require a dialogue between biologists and computer scientists having experience in grammar development. Once agreed upon, grammars can easily be recorded in the MySQL database. Even though it would be attractive to guide the user in the definition of new grammars, the complexity of this process makes it unlikely that it will be possible to develop a grammar-building wizard in the foreseeable future.

4.6 Summary and Future Work

Like the stand-alone Gene Designer [215] or the web-based Registry of Standard Biological Parts, GenoCAD allows users to quickly design new genetic constructs by combining sequences corresponding to various functional elements known as parts. Unlike its predecessors though, GenoCAD guides the user through a design workflow corresponding to previously agreed upon design principles captured in grammars. Since it relies on linguistic models of DNA sequences [188], GenoCAD is a tool to help users write in the language of DNA sequences.

GenoCAD is a work in progress. The sequence validation tool makes it possible to test whether a sequence developed outside of GenoCAD is consistent with a specific grammar and parts library. This tool is still rudimentary since it simply provides pass/fail information. Eventually, more sophisticated error messages will be generated to help user troubleshoot their sequence. As the GenoCAD user base grows, GenoCAD will support workgroups by allowing them to share parts, libraries and grammars.

The next major improvement to GenoCAD will include tools to predict the design behavior. By augmenting GenoCAD data model it is possible to compile a design DNA sequence into a SBML file [97] that can be simulated by one of the numerous applications that supports this standard [93]. GenoCAD will then join a growing number of applications experimenting with mechanisms to derive the gene network model encoded in genetic con- structs composed of standard biological parts [80, 92, 137, 175]. GenoCAD will also integrate tools to track the synthesis and assembly of designs generated in GenoCAD. Optimizing the DNA fabrication process based on the strategies used to design a series of constructs would be extremely valuable.

While the GenoCAD web site is stable and has been in operation since 2007, the experimental validation of the concepts upon which it has been developed is still ongoing. As a reminder of the necessity to test designs in the lab, we will consider GenoCAD in beta test until extensive characterization of GenoCAD-designed systems has been described in peer-reviewed publications.

4.7 Funding

Virginia Commonwealth Research Initiative; fellowship from the Virginia Tech Genetics, Bioinformatics and Computational Biology graduate program awarded to YC. Funding for open access charge: Virginia Bioinformatics Institute.

4.8 Attribution

Michael Czar – Ph.D. (Department of Pharmacology, University of Michigan), was a senior project associate at Virginia Bioinformatics Institute at Virginia Tech, wrote the paper.

Yizhi Cai – Graduate student (Genetics, Bioinformatics and Computational Biology interdisciplinary doctoral program, Virginia Tech), performed the experiments and wrote the paper.

Jean Peccoud – Ph.D (Department of Molecular Biology and Bioinformatics, Universit Joseph Fourier (Grenoble I)) currently an associate professor of Virginia Bioinformatics Institute at Virginia Tech. JP conceived and designed the experiments, and wrote part of the paper.

Chapter 5

GenoCAD for iGEM: a grammatical approach to the design of standard-compliant constructs

Published at: Y. Cai, M. Wilson, and J. Peccoud. Genocad for iGEM: a grammatical approach to the design of standard-compliant constructs. Nucleic Acids Res, page gkq086v1, Feb 2010.

Authors: Yizhi Cai, Mandy L. Wilson and Jean Peccoud

Abstract: One of the foundations of synthetic biology is the project to develop libraries of standardized genetic parts that could be assembled quickly and cheaply into large systems. The limitations of the initial BioBrick standard have prompted the development of multiple new standards proposing different avenues to overcome these shortcomings. The lack of compatibility between standards, the compliance of parts with only some of the standards or even the type of constructs that each standard supports have significantly increased the complexity of assembling constructs from standardized parts. Here, we describe computer tools to facilitate the rigorous description of part compositions in the context of a rapidly changing landscape of physical construction methods and standards. A context-free grammar has been developed to model the structure of constructs compliant with six popular assembly standards. Its implementation in GenoCAD makes it possible for users to quickly assemble from a rich library of genetic parts, con- structs compliant with any of six existing standards.

5.1 Introduction

The compelling vision of libraries of biological components with standardized interfaces enabling a fast and cheap assembly of large biological systems is one of the foundations of synthetic biology [62, 85]. The BioBrick Foundation (BBF) has been instrumental in promoting the BioBrick standard. A BioBrick compliant part is a DNA fragment flanked by a prefix and a suffix sequence having specific restriction sites [8, 32]. Two BioBrick parts can be assembled by using a specific series of restriction digestions and ligations independent of the parts sequences. The different restriction sites used by the prefix and suffix result in complementary overhangs that can be ligated without recreating any of the prefix and suffix restriction sites. The legacy sequence between two adjoining parts is called the scar. BioBrick parts are physically composable in the sense that the assembly of two BioBricks results in a new part compliant with the same standard. The first BioBrick assembly standard, BBa1.0, was proposed by Knight in a BBF Request For Comments (BBF RFC 10). It uses EcoRI, NotI and Xbal in the prefix, and SpeI, NotI and PstI in the suffix. Later on, it has been proposed to replace PstI with SbfI, an enzyme with a longer restriction site less likely to be found in parts sequences (BBF RFC 11). Both standards have been well received by the community and widely used by teams enrolled in the international Genetically Engineered Machine (iGEM) competition [82, 201]. However, both BBa1.0 and BBa2.0 create an eight-base scar (TACTAG AG), which results in a frame shift when assembling two protein-coding sequences. To address this problem, several new standards have been proposed (BBF RFCs 12, 21, 23 and 25) to allow protein fusion by introducing six-base scars. These standards are summarized in Table 5.1.

'The best thing about standards is that there are so many to choose from!' summarizes well the difficulty of navigating this increasingly complicated technical land- scape. The multiplication of assembly standards creates a number of new difficulties. Most parts are only compliant with some of the assembly standards due to the presence of reserved restriction sites in their sequence. A design framework that could automatically manage the constraints associated with the different standards could help the community better leverage ongoing standardization efforts. Here, we introduce a context-free grammar (CFG) [29] to model the structure of genetic con- structs compliant with any of the existing assembly standards. A CFG is a set of rewriting rules, which defines the set of all designs that can be derived by the grammar. A context-free rule can be written as $\chi \rightarrow \gamma$, where χ is a single non-terminal and γ is any string of terminals and/or non-terminals (possibly empty). In the case of the BioBrick grammar presented in this article, non-terminals include parts categories (e.g. promoter) and categories of composite parts (e.g. cistron), while terminals are specific BioBricks (e.g. BBa_R0040) and standard-specific prefixes, suffixes and scars. For instance, a rule "$Cass1 \rightarrow Prom1\,C1\,Cist1\,C1\,Term1$" is interpreted as an expression cassette (Cass1) can be transformed into a DNA sequence comprising a promoter (Prom1), a BioBrick scar (C1), a cistron (Cist1), a BioBrick scar (C1) and a terminator (Term1).

The grammar was implemented in GenoCAD (`www.genocad.org`), a web-based application to design synthetic genetic constructs [48]. GenoCAD is built upon a solid computational linguistic foundation. Yet, its point-and-click graphical user interface enables users to design complex constructs in a matter of minutes. GenoCAD captures design strategies of synthetic genetic constructs in the form of grammatical models. The linguistic models can be used in two ways: a user can design a synthetic construct by successively selecting design rules to transform the structure of the design; or a user can upload a DNA sequence designed outside GenoCAD to validate its consistency with the grammatical model. GenoCAD provides a central parts database with each grammar, and the BioBrick grammar comes with a library of 2312 basic genetic parts available in the Registry of Standard Biological Parts in May 2009. Users, who elect to create a GenoCAD personal account, can log in the system to create project-specific parts libraries, upload new parts into their workspace and save designs for later use.

5.2 Materials and Methods

A static snapshot of the Registry content is available as a FASTA file at `http://partsregistry.org/fasta/parts/All_Parts`. For each part, the file includes its identifier, category, a short description and the part sequence. The version of this file published in May 2009 included 9526 parts. A Perl script was developed to parse out the content of this file into structured data format, which could be imported into a MySQL database.

5.3 Results

Compliance with different BioBrick standards

The Registry includes both basic parts (e.g. promoter and RBS) and composed parts, which include multiple basic parts (e.g. device, project and composite). As the set of composed parts can be regenerated from the basic parts [155], we only focused on the basic parts which include categories of Regulatory, RBS, Coding, Terminator and, Plasmid Backbone. By querying the MySQL database, we extracted a set of 2312 basic parts with DNA sequences. Because a part is compatible with a BioBrick standard if its sequence does not include any of the restriction sites used by the assembly standard, we developed SQL queries to check for the presence of the restriction sites listed in Table 5.1.

Interestingly, there are 2166 parts compliant with the BBa1.0, BBa2.0 and Biofusion standards. This observation is not surprising because these three standards use almost identical

Table 5.1: Summary of prefix, suffix and scar groups used in different BioBrick assembly standards

Standard	Reference	Prefix	Suffix	Scar	Compatible parts				
					Prom.	RBS	Gene	Ter.	PB
BBa1.0	RFC 10	EcoRI NotI	SpeI NotI	TACTAGAG	2166				
		XbaI	PstI		761	149	1149	98	9
BBa2.0	RFC 11	EcoRI NotI	SpeI NotI	TACTAGAG	2166				
		XbaI	SbfI/PstI		761	149	1149	98	9
Biofusion	RFC 23	EcoRI NotI	SpeI NotI	ACTAGA	2166				
		XbaI	PstI		761	149	1149	98	9
Freiburg	RFC 25	EcoRI NotI	AgeI SpeI	ACCGGC	1969				
		XbaI Met NgoMIV	NotI PstI		743	148	973	96	9
BBb	RFC 21	EcoRI	BamHI	GGATCT	2019				
		BglII	XhoI		636	149	1112	83	39
Knight	RFC 12	EcoRI	NheI	GCTAGT	2140				
		SpeI	PstI		724	150	1159	97	10

restriction sites. There are slightly fewer parts available for newly proposed standards like the BBb standard.

Grammar design

The general methodology of developing grammars to model the structure of synthetic genetic constructs has been described elsewhere [29]. Here, we highlight the introduction of new rewriting rules and non-terminals that augment the previously described grammars. The full grammar is described in Table 5.2.

Table 5.2: A CFG describing different BioBrick assembly standards. Terminals are italicized. P, C and S are terminals representing prefix, scar and suffix, respectively. As BBa2.0 uses the same prefix and scar as BBa1.0, there is no P2 and C2 in the grammar.

Rule	Comments	Left term	Right term
1	Select a standard (BBa1.0)	S	BBa1.0
2	Similar to rule 1	S	BBa2.0
3	Similar to rule 1	S	Biofusion
4	Similar to rule 1	S	Freiburg
5	Similar to rule 1	S	BBb
6	Similar to rule 1	S	Knight
7	Transform a standard (BBa1.0) into a plasmid backbone (PB1), a prefix (P1), a cassette (Cass1) and a suffix (S1)	BBa1.0	PB1 P1 Cass1 S1
8	Transform a cassette (Cass1) into two cassettes (Cass1) with a scar (C1) in between	Cass1	Cass1 C1 Cass1
9	Reverse the sequence orientation of a cassette (Cass1)	Cass1	[Cass1]
10	Transform a cassette (Cass1) into a promoter (Prom1), a scar (C1), a cistron (Cist1), a scar (C1) and a terminator (Term1)	Cass1	Prom1 C1 Cist1 C1 Term1
11	Transform a cistron (Cist1) into two cistrons (Cist1) with a scar (C1) in between	Cist1	Cist1 C1 Cist1
12	Transform a cistron (Cist1) into a RBS (RBS1), a scar (C1) and a gene (Gene1)	Cist1	RBS1 C1 Gene1
13	Transform a terminator (Term1) into two terminators (Term1) with a scar (C1) in between	Term1	Term1 C1 Term1
14	Similar to rule 7	BBa2.0	PB2 P1 Cass2 S2
15	Similar to rule 8	Cass2	Cass2 C1 Cass2

Continued on next page...

Table 5.2 (Continued)

Rule	Comments	Left term	Right term
16	Similar to rule 9	Cass2	[Cass2]
17	Similar to rule 10	Cass2	Prom2 C1 Cist2 C1 Term2
18	Similar to rule 11	Cist2	Cist2 C1 Cist2
19	Similar to rule 12	Cist2	RBS2 C1 Gene2
20	Similar to rule 13	Term2	Term2 C1 Term2
21	Similar to rule 7	Biofusion	PB3 P3 Cass3 S3
22	Similar to rule 8	Cass3	Cass3 C3 Cass3
23	Similar to rule 9	Cass3	[Cass3]
24	Similar to rule 10	Cass3	Prom3 C3 Cist3 C3 Term3
25	Similar to rule 11	Cist3	Cist3 C3 Cist3
26	Similar to rule 12	Cist3	RBS3 C3 Gene3
27	Transform a gene (Gene3) into two genes (Gene3) with a scar (C3) in between, i.e. protein fusion	Gene3	Gene3 C3 Gene3
28	Similar to rule 13	Term3	Term3 C3 Term3
29	Similar to rule 7	Freiburg	PB4 P4 Cass4 S4
30	Similar to rule 8	Cass4	Cass4 C4 Cass4
31	Similar to rule 9	Cass4	[Cass4]
32	Similar to rule 10	Cass4	Prom4 C4 Cist4 C4 Term4
33	Similar to rule 11	Cist4	Cist4 C4 Cist4
34	Similar to rule 12	Cist4	RBS4 C4 Gene4
35	Similar to rule 27	Gene4	Gene4 C4 Gene4
36	Similar to rule 13	Term4	Term4 C4 Term4
37	Similar to rule 7	BBb	PB5 P5 Cass5 S5
38	Similar to rule 8	Cass5	Cass5 C5 Cass5
39	Similar to rule 9	Cass5	[Cass5]
40	Similar to rule 10	Cass5	Prom5 C5 Cist5 C5 Term5
41	Similar to rule 11	Cist5	Cist5 C5 Cist5
42	Similar to rule 12	Cist5	RBS5 C5 Gene5
43	Similar to rule 27	Gene5	Gene5 C5 Gene5
44	Similar to rule 13	Term5	Term5 C5 Term5
45	Similar to rule 7	Knight	PB6 P6 Cass6 S6
46	Similar to rule 8	Cass6	Cass6 C6 Cass6
47	Similar to rule 9	Cass6	[Cass6]
48	Similar to rule 10	Cass6	Prom6 C6 Cist6 C6 Term6
49	Similar to rule 11	Cist6	Cist6 C6 Cist6
50	Similar to rule 12	Cist6	RBS6 C6 Gene6
51	Similar to rule 27	Gene6	Gene6 C6 Gene6
52	Similar to rule 13	Term6	Term6 C6 Term6

Figure 5.1 lists the non-terminals along with the icons used for their graphical representation.

CATEGORY	NON-TERMINALS	ICONS
Start symbol	S	
Reverse Orientation	[...]	
Standards	BBa1.0, BBa2.0, Biofusion...Knight	
Plasmid backbones	PB1, PB2, PB3...PB6	
Prefix / Scar / Suffix	P1...C1...S1 P1...C1...S2 P3...C3...S3 P4...C4...S4 P5...C5...S5 P6...C6...S6	
Cassette	Cass1, Cass2, Cass3...Cass6	
Cistron	Cist1, Cist2, Cist3... Cist6	
Promoter	Prom1, Prom2, Prom3...Prom6	
RBS	RBS1, RBS2, RBS3...RBS6	
Terminator	Term1, Term2, Term3...Term6	
Gene	Gene1, Gene2, Gene3...Gene6	

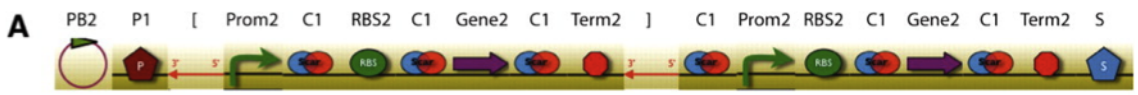
Figure 5.1: Correspondence between parts categories, non-terminals and icons used to graphically represent construct structures.

S is the start symbol used to initiate the design process. In order to ensure the consistency of a design with a specific standard, it is necessary to introduce for each category of parts a different non-terminal for each standard. For instance, instead of having a single non-terminal for genes, we defined the non-terminal Gene1 to represent genes compliant with the BBa1.0 standard, Gene2 for genes compliant with BBa2.0 standards and so on. Non-terminals P, C and S were introduced to represent the prefixes, scars and suffixes of different standards. Non-terminals PB1PB6 represent the plasmid backbone. Finally, we used non-terminals that do not correspond to specific DNA sequences. A class of non-terminals is used to represent the different assembly standards. Square brackets are introduced to represent that part of a construct is coded on the reverse strand of the DNA molecule, as illustrated in Figure 5.2.

Table 5.2 lists all the production rules of the six standards. Rules P1P6 specify the assembly standard the design complies with. Rules P7, P14, P21, P29, P37 and P45 specify that a design is composed of a plasmid backbone and a gene expression cassette flanked by the standard prefix and suffix. P8, P15, P22, P30, P38 and P46 allow a single cassette to be transformed into two cassettes with a scar in the middle. Applying these rules n times will create $n + 1$ cassettes in the design. P9, P16, P23, P31, P39 and P47 can be used to reverse the orientation of a cassette. P10, P17, P24, P32, P40 and P48 define the structure of a cassette to be a promoter, a cistron and a terminator, separated by scars. P11, P18, P25, P33, P41 and P49 allow multiple cistrons in a cassette. P12, P19, P26, P34, P42 and P50 specify that a cistron is composed of a RBS, a scar and a gene. P13, P20, P28, P36, P44 and P52 allow introducing multiple terminators. As BBa1.0 and BBa2.0 both use an eight-base scar (TACTAGAG), which results in the frame shift, protein fusion is not permissible. However, the other standards use six-base scars (such as ACTAGA for the Biofusion standard) compatible with in-frame fusion of protein-coding parts. The grammar reflects this fact by having rules P27, P35, P43 and P51 for protein fusion while using those standards.

GenoCAD implementation

We imported all the basic parts present in the Registry of Standard Biological Parts into the GenoCAD-backend database. We also implemented the BioBrick grammar in GenoCAD. The large number of parts included in the BioBrick parts library may be difficult to navigate when working on a specific project. After they have logged into the system, users can customize their workspace by adding new parts and creating new parts libraries. When starting a project, it is suggested to first create a parts library for the project. This parts library should contain all the parts needed for the project. Most parts will be imported from the general BioBrick library. However, if there is a need for additional parts, it is possible to define new parts and include them in the project parts library.



B

ggatcctaactcgaggttacattgtcgatctgttcatggtgaacagctttaaatgcaccaaaaa
ctcgtaaaagctctgatgtatctatctttttacaccgttttcatctgtgcatatggacagttt
tccctttgatatctaacggtgaacagttgttctacttttgtttgttagtcttgatgcttcactg
atagatacaagagccataagaacctcagatccttccgtatttagccagtatgttctctagtgtg
aattcgcggccgcttctagag [TATAAACGCAGAAAGGCCCACCCGAAGGTGAGCCAGTGTGAC
TCTAAGTAGAGAGCGTTCACCGACAAACAACAGATAAAACGAAAGGCCCAGTCTTTCGACTGAG
CCTTTCGTTTTATTTGATGCCTGGTACTAGAGCATTATCACCGCCAGAGGTAAAATAGTCAACA
CGCACGGTGTTAGATATTAAAGAACACTTAAATTAGATTTAACGTATACTAGAGAAAGAGGAGA
AATACTAGATGAGCACAAAAATACTAGAGATTAAAGAGGAGAAATACTAGAGTACTAGAGCCAG
GCATCAAATAAAACGAAAGGCTCAGTCGAAAGACTGGGCCTTTCGTTTTATCTGTTGTTTGTCG
GTGAACGCTCTC] tactagaggcaaccattatcaccgccagaggtaaaatagtcaacacgcacg
gtgttagatatttataaatagtggtgatagattattaaagaacacttaaataaagaggagaaat
actagagattaaagaggagaaatactagagtactagagtcacactggctcaccttcgggtgggc
ctttctgcgtttatatactagtagcggccgccctgcagg

C

ggatcctaactcgaggttacattgtcgatctgttcatggtgaacagctttaaatgcaccaaaaa
ctcgtaaaagctctgatgtatctatctttttacaccgttttcatctgtgcatatggacagttt
tccctttgatatctaacggtgaacagttgttctacttttgtttgttagtcttgatgcttcactg
atagatacaagagccataagaacctcagatccttccgtatttagccagtatgttctctagtgtg
aattcgcggccgcttctagag GAGAGCGTTCACCGACAAACAACAGATAAAACGAAAGGCCCAG
TCTTTCGACTGAGCCTTTCGTTTTATTTGATGCCTGGCTCTAGTACTCTAGTATTTCTCCTCTT
TAATCTCTAGTATTTTTGTGCTCATCTAGTATTTCTCCTCTTTCTCTAGTATACGTTAAATCTA
ATTTAAGTGTTCTTTAATATCTAACACCGTGCGTGTTGACTATTTTACCTCTGGCGGTGATAAT
GCTCTAGTACCAGGCATCAAATAAAACGAAAGGCTCAGTCGAAAGACTGGGCCTTTCGTTTTAT
CTGTTGTTTGTCGGTGAACGCTCTCTACTTAGAGTCACACTGGCTCACCTTCGGGTGGGCCTTT
CTGCGTTTATA tactagaggcaaccattatcaccgccagaggtaaaatagtcaacacgcacggt
gttagatatttataaatagtggtgatagattattaaagaacacttaaataaagaggagaaatac
tagagattaaagaggagaaatactagagtactagagtcacactggctcaccttcgggtgggcct
ttctgcgtttatatactagtagcggccgccctgcagg

Figure 5.2: Three different representations of a BBa2.0 design. (A) This design includes two gene expression cassettes in opposite orientations. The first icon represents the construct backbone. The icons P1 (second to the left) and S2 (last) represent the construct prefix and suffix. The brackets [and] indicate the reverse orientation of the first cassette. Because BBa1.0 and BBa2.0 share the same prefix and scars, the design includes P1, C1 and S2. (B) The sequence generated by the grammar includes the special characters [and] to indicate the fragment in reverse orientation in bold characters. (C) The sequence of the construct is generated by replacing the sequence in bold character by its reverse complement.

Once the project library is complete, the design phase can start. After selecting the BioBrick grammar, the project-specific parts library can be selected. The construct design proceeds through a series of rewriting operations corresponding to the selection of specific grammar rules. The BioBrick grammar first prompts the user to select a particular assembly standard and then a cloning vector. The design then proceeds through a series of steps to specify the structure of the constructs and specific parts to implement this structure. A more detailed description of the design workflow and GenoCAD various features have been published recently [48].

The recent multiplication of assembly standards has led to new design challenges. When all parts complied with a single standard, it was very straightforward to combine them. Now, it becomes necessary to verify that all parts used in a project are consistent with the standard selected for the project. GenoCAD structured approach to the design of genetic constructs makes it possible to gracefully navigate complex libraries of genetic parts compliant with multiple assembly standards. Once a standard has been selected, only the parts compatible with this standard are available to the designer. The construct prefix and suffix along with the scars are properly represented along with the sequence of the cloning vector used to propagate the design.

Designing an iGEM project using GenoCAD

To demonstrate how to use GenoCAD and the BioBrick grammar to quickly design an iGEM project, we selected the wintergreen odor biosynthetic system (`http://bit.ly/85Hhgd`) designed and implemented by the MIT iGEM team in 2006. The system contains two expression cassettes: one produces salicylate acid from the cellular metabolite, and the second one converts the salicylic acid to methyl salicylate that produces the wintergreen odor. We designed this system with the BBa1.0 assembly standard using GenoCAD. The step-by-step design process is depicted in Figure 5.3. The design process starts with selecting the BBa1.0 assembly standard (P1); P7 is used to transform the design into a plasmid backbone, a prefix, a cassette and a suffix; as there are two cassettes needed in the wintergreen odor biosynthetic system, P8 is applied to allow two cassettes in the design; by applying P10 to both cassettes, we specified the structure of each cassette to be a promoter, a scar, a cistron, a scar and a terminator; by applying P12 to each cistron, the structure of a cistron is expanded to a RBS, a scar and a gene; finally, we used P13 to allow the usage of a double terminator in each cassette, which ensures a tight transcription termination. After specifying the structure of the design, the last step is to select a specific part for each category, and the DNA sequence of the design is ready for export as a text file.

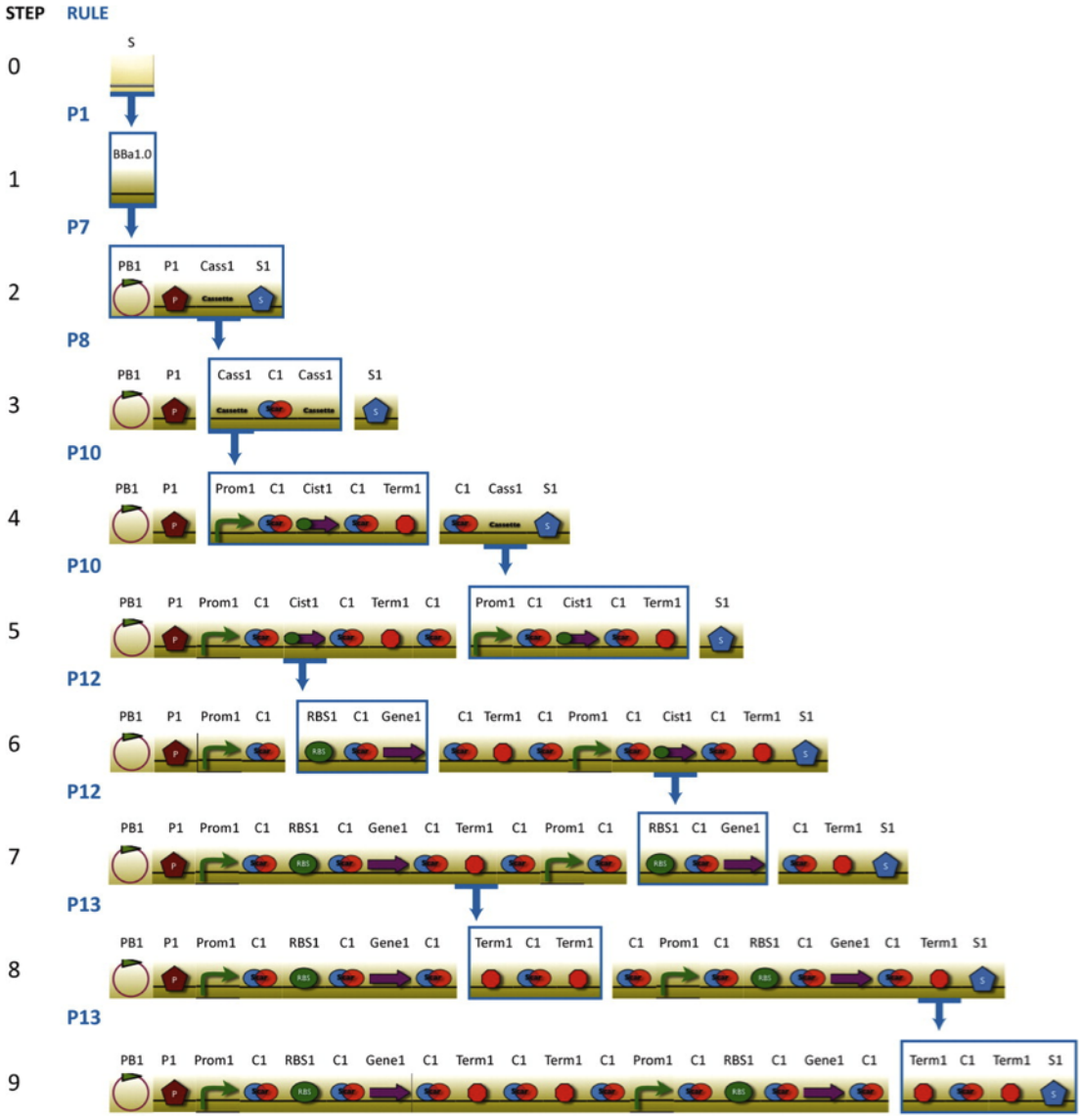


Figure 5.3: Step-by-step design process of a wintergreen odor system using GenoCAD. The construct is designed in nine steps. For each step, the rewriting rule used is indicated in blue in the second column using the same number as in Table 5.2. The rewriting resulting from the rule selection is indicated in the graphical representation of the construct. The icon underlined by the base of the arrow indicates the left term of the rule. The icons enclosed in a blue rectangle correspond to the right term of the rule. For instance, applying the rule P8 to Cass1 in step 2 transforms this element into Cass1 C1 Cass1 in step 3.

5.4 Discussion

Data exchange

The method used to import in GenoCAD data from the Registry suffers from a number of obvious limitations. A live connection between parts registries has been envisioned for some time. The recently launched BioBrick Parts Catalog (`www.biobrickparts.org`) provides an API to read its content using the JavaScript Object Notation (JSON). We are also working on the development of web services allowing other clients to communicate with the GenoCAD database.

However, setting up web services to access databases solves only part of the data exchange challenge. As data are available easily, it will become apparent that the nature of the data exchanged needs to be documented. It is safe to assume that all registries will associate a unique identifier, a DNA sequence and a description with the parts. The description of the nature of parts is a more difficult issue. The Registry of Standard Biological Parts, the BioBrick Parts Catalog and GenoCAD use their own system of categories, but these categorization systems are developed independently of each other making it difficult to map categories of one resource into categories used by another system. This problem can be solved by the development of an ontology giving the community a common controlled vocabulary to describe genetic parts. Early efforts to develop the Synthetic Biology Open Language have been somewhat hampered by the magnitude of the task. In particular, it is difficult to properly appreciate the scope of what needs to be described by this language. It is also challenging to evaluate the possibility of using existing ontologies like the Sequence Ontology [59] for this new application.

Ensuring that parts are properly delimited at the DNA sequence level is another challenge. The BioBrick grammar presented in this article carefully handles the fusion of coding sequences when using assembly standards allowing this type of construct. However, the possible inclusion of a stop codon in the sequence of genes may prevent the actual fusion of two adjacent proteins. It is, therefore, necessary to set standards to delimit the DNA sequences of different categories of parts (BBF RFC 13).

Advantages and limitations of the BioBrick grammar

The syntactic model proposed in this article constrains the design space of BioBrick-based systems. The point- and-click approach to the design process makes it easy for someone to quickly design constructs compliant with any of the proposed standards. The design strategy embedded in the grammar is very conservative to maximize the chances of designing functional systems. However, GenoCAD currently excludes some 'out of the box'

designs, e.g. an expression cassette with multiple promoters. Advanced users can overcome this limitation by creating new parts in their personal workspace. For instance, it is possible to use a sequence editor to combine the sequence of two promoters and then save it in GenoCAD as a regular promoter. Domain-specific languages like Eugene (`http://sourceforge.net/projects/eugene`) or GEC [158] provide users with richer frameworks and greater design flexibility, but these programming environments may have steeper learning curves than GenoCAD. The Registry of Standard Biological Parts or Gene Designer [215] provides the ultimate design flexibility by allowing users to combine any parts in any order, but the lack of verification or guidance creates more opportunities for design errors that will manifest only when the part is fabricated or characterized.

Fabrication

GenoCAD and the BioBrick grammar described in this article do not provide users with a path to fabrication, but it generates the theoretical DNA sequence of a design that can be used to analyze sequencing data collected to verify physical implementations of a design.

It is fairly common for expression vectors to include expression cassettes in opposite orientations [72] as this configuration limits interferences between promoters. The BioBrick grammar allows users to select the orientation of gene expression cassettes. The reverse complementation operation, necessary to generate the final sequence, includes a reverse complementation of the scar sequences between parts. The final sequence is identical to the sequence of a cassette first assembled in a direct orientation and then flipped before its insertion into cloning vector. Because most parts are defined in the same orientation in the various registries, this scenario is the most likely assembly strategy, but other strategies leading to different DNA sequences can be imagined.

Choosing an assembly standard is only one element in the development of an assembly strategy. The availability of clones, representing physical implementations of various elements of the design, guides a fabrication process that often includes de novo synthesis steps and assembly of existing DNA fragments using various cloning techniques[47]. Note that the choice of a particular assembly standard does not automatically restrict the user to a specific assembly process. As they do not rely on restriction enzymes, USER fusion (BBF RFC 39) [167, 199] or In-Fusion cloning (BBF RFC 26) [20] are compatible with any assembly standard. The determination of an optimal assembly process can be solved by dynamic programming algorithms (18).

GenoCAD for iGEM

GenoCAD provides users having limited domain expertise with a user-friendly environment to quickly design structurally valid BioBrick constructs compliant with different assembly standards. Students enrolled in the iGEM competition represent an important group of potential users, and the BioBrick grammar has been developed with this group in mind. By importing parts available in the Registry, reusing the system of categories used by the Registry, capturing physical and basic functional composition rules, the BioBrick grammar customizes the GenoCAD environment for the needs of iGEM participants. As a result, any curation of the data imported from the registry has been avoided.

GenoCAD is part of a quickly growing arsenal of software tools for synthetic biology [36, 92, 137, 138]. It has been recently proposed to use attribute grammars, an extension of the CFG formalism used in this report, to develop semantic models of DNA sequences [30]. Embedding this formalism in GenoCAD will enable users to translate their designs into SBML files describing their expected behavior. This capability will make it possible to investigate the possible influence on gene expression of the scars associated with the different assembly standards. It was initially assumed that scars would not significantly influence the phenotype coded in a genetic design, but rapid progress in the characterization of the relations between structure and functions of ribosome-binding sites [121, 182] and promoters [46, 60] may contribute to re-evaluate this hypothesis.

5.5 Acknowledgements

Authors acknowledge Jenny Wang for assisting in the preparation of the figures and the BBF and the iGEM community for contributing the RFCs and BioBricks parts upon which this project was developed.

5.6 Funding

National Science Foundation Award EF-0850100; Genetics, Bioinformatics and Computational Biology interdisciplinary graduate program at Virginia Tech (a graduate fellowship to Y.C.). Funding for open access charge: National Science Foundation Award EF-0850100.

5.7 Attribution

Yizhi Cai – Graduate student (Genetics, Bioinformatics and Computational Biology interdisciplinary doctoral program, Virginia Tech), conceived, designed and performed the experiments, and wrote the paper.

Mandy L. Wilson – is a senior developer and database administrator at Virginia Bioinformatics Institute at Virginia Tech, performed the experiments.

Jean Peccoud – Ph.D (Department of Molecular Biology and Bioinformatics, Universit Joseph Fourier (Grenoble I)) currently an associate professor of Virginia Bioinformatics Institute at Virginia Tech. JP conceived and designed the experiments, and wrote the paper.

Chapter 6

Modeling Structure-Function Relationships in Synthetic DNA Sequences using Attribute Grammars

Published at: Y. Cai, M. W. Lux, L. Adam, and J. Peccoud. Modeling structure-function relationships in synthetic DNA sequences using attribute grammars. PLoS Comput Biol, 5(10):e1000529, Oct 2009.

Authors: Yizhi Cai, Matthew W. Lux, Laura Adam, Jean Peccoud

Abstract: Recognizing that certain biological functions can be associated with specific DNA sequences has led various fields of biology to adopt the notion of the genetic part. This concept provides a finer level of granularity than the traditional notion of the gene. However, a method of formally relating how a set of parts relates to a function has not yet emerged. Synthetic biology both demands such a formalism and provides an ideal setting for testing hypotheses about relationships between DNA sequences and phenotypes beyond the gene-centric methods used in genetics. Attribute grammars are used in computer science to translate the text of a program source code into the computational operations it represents. By associating attributes with parts, modifying the value of these attributes using rules that describe the structure of DNA sequences, and using a multi-pass compilation process, it is possible to translate DNA sequences into molecular interaction network models. These capabilities are illustrated by simple example grammars expressing how gene expression rates are dependent upon single or multiple parts. The translation process is validated by systematically generating, translating, and simulating the phenotype of all the sequences in the design space generated by a small library of genetic parts. Attribute grammars represent a flexible framework connecting parts with models of biological function. They will be instrumental for building mathematical models of libraries of genetic constructs synthesized to

characterize the function of genetic parts. This formalism is also expected to provide a solid foundation for the development of computer assisted design applications for synthetic biology.

Author Summary: Deciphering the genetic code has been one of the major milestones in our understanding of how genetic information is stored in DNA sequences. However, only part of the genetic information is captured by the simple rules describing the correspondence between gene and proteins. The molecular mechanisms of gene expression are now understood well enough to recognize that DNA sequences are rich in functional blocks that do not code for proteins. It has proved difficult to express the function of these genetic parts in a computer readable format that could be used to predict the emerging behavior of DNA sequences combining multiple interacting parts. We are showing that methods used by computer scientists to develop programming languages can be applied to DNA sequences. They provide a framework to: 1) express the biological functions of genetic parts, 2) how these functions depend on the context in which the parts are placed, and 3) translate DNA sequences composed of multiple parts into a model predicting how the DNA sequence will behave in vivo. Our approach provides a formal representation of how the biological function of genetic parts can be used to assist in the engineering of synthetic DNA sequences by automatically generating models of the design for analysis.

6.1 Introduction

"How much can a bear bear?" This riddle uses two homonyms of the word "bear". The first instance of the word is a noun referring to an animal, and the second is a verb meaning "endure". Although the word "bear" has over 50 different meanings in English, its meaning in any given sentence is rarely ambiguous. In a simple case like this riddle, the meaning of each word can be deciphered by looking at other words in the same sentence. In other cases, it is necessary to take into account a broader context to properly interpret the word. For instance, it may be necessary to read several sentences to decide if "bear claw" refers to a body part or a pastry. A reader will progressively derive the meaning of a text by recognizing structures consistent with the language grammar. It is often difficult to understand the meaning of a text by relying exclusively on a dictionary.

It is interesting to compare this bottom-up emergence of meaning with the top-down approach that made genetics so successful. The discipline was built upon a quest to define hereditary units that could be associated with observable traits well before the physical support of heredity was discovered [109, 206]. The one-to-one relationship between genes and traits was later refined by Beadle and Tatum's hypothesis that the gene action was mediated by enzymes [197, 210]. Cracking the genetic code has been one of the major milestones in understanding the information content of nucleic acids sequences. By demonstrating the colinearity of DNA, RNA, and protein sequences, the genetic code was instrumental in the

identification of specific DNA sequences as genes. The influence of this legacy on contemporary biology cannot be underestimated. Models used in quantitative genetics predict phenotypes from unstructured lists of alleles at different loci [65, 136]. Similarly, genome annotations remain very gene-centric. Most bioinformatics databases have been designed to collect information relative to coding regions or candidate genes. Few, if any, annotations of non-coding regions or higher order structures are being systematically recorded even for model organisms like yeast [88, 223].

Yet, despite its success, the notion of gene appears insufficient to express the complexity of the relation between an organism genome and its phenotype [109, 110] The elucidation of the molecular mechanisms controlling gene expression has revealed a web of molecular interactions that have been modeled mathematically to show that important phenotypic traits are the emerging properties of a complex system [37, 166, 205, 212, 218, 220]. The development of this more integrated understanding of the cell physiology leads to a progressive adoption of the more neutral notion of genetic part as a replacement for the notion of genes associated with specific traits. Making sense of the list of parts generated in genomics, proteomics, and metabolomics has been a major challenge for the systems biology community [10, 21, 58, 68, 146, 203].

It is becoming apparent that the genetic code captures only a small fraction of the information content of DNA molecules [163, 192]. Yet, if there is a general agreement that the cell dynamics is somehow coded in genetic sequences, no formal relationship between DNA sequences and dynamical models of gene expression has been proposed so far. In particular, the formalization of the biological functions of genetic parts has remained elusive. As a result, building models of gene networks encoded in DNA sequences remains a labor-intensive process. This limitation has hampered the development of large families of models needed to analyze phenotypic data generated by libraries of related genetic constructs [46, 72, 73, 86, 87, 150].

Synthetic biology is likely to be instrumental in refining our understanding of the design of natural biological systems [55]. Just like the genetic code was partly elucidated through the de novo chemical synthesis of DNA molecules [2, 107], the redesign of genomic sequences will shed a new light on the relations between structure and function in genetic sequences [35, 56, 75]. By considering biological parts as the building blocks of artificial DNA sequences [62], designing new parts that do not exist in nature [46, 73, 150], and making parts physically available to the community [155], synthetic biology calls for a systematic functional characterization of genetic parts [32]. These efforts are still limited by the difficulty in expressing how the function of biological parts may be influenced by the structure of the DNA sequence in which they are used. It has been shown that a partial redesign of the genomic sequences of two viruses had a significant effect on the virus fitness even though the redesigns preserved the protein sequences [35, 42]. Just as the context of the expression

"bear claw" helps understand its meaning, it is necessary to consider the entire structure of the DNA molecule coding for particular genes to appreciate how those genes contribute to the phenotype.

One possible approach to this problem is to extend the linguistic metaphor used to formulate the central dogma. The notions of genetic code, transcription, and translation are derived from a linguistic representation of biological sequences. Several authors have modeled the structure of various types of biological sequences using syntactic models [39, 54, 76, 115, 172, 183, 186, 188]. However, these structural models have not yet been complemented by formal semantic models expressing the sequence function. An interesting attempt to use grammars to model the dynamics of gene expression did not rely on a description of the DNA sequence structure. Instead, this grammar described how various inducible or repressible promoters can transition between different states under the control of environmental parameters [18]. The simple semantic model stored in a knowledge base established a correspondence between the strings generated by the syntax and the physiological state of the cell. The Sequence Ontology [59] and the Gene Regulation Ontology [15] represent other attempts to associate semantic values with biological sequences. Their controlled vocabularies can be used by software applications to manage knowledge. However, the semantics derived from these ontologies is a semantics of the sequence annotation, not of the sequences themselves.

6.2 Model

We recently described a fairly simple syntactic model of synthetic DNA sequences [29] capable of generating a large number of previously published synthetic genetic constructs [61, 72, 86]. We have now enhanced this initial syntactic model with a formal semantic model capable of expressing the dynamics of the molecular mechanisms coded by the DNA sequences. Specialized terms like syntax, semantics, and others are defined in Table 6.1. Our approach uses attribute grammars [151], a theoretical framework developed in the 60s to establish a formal correspondence between the text of a computer program and the series of microprocessor operations it codes for [116, 117]. Even though other types of semantic models have been developed since then [198, 204], attribute grammars still represent a good compromise between simplicity and expressivity, an important characteristic to ensure that the framework can be used by non-computer scientists. Attribute grammars make it possible to use well characterized compilation algorithms to translate a DNA sequence into a mathematical model of the molecular interactions it codes for. As the static source code of a program directs the dynamic series of operations carried out by the microprocessor based on user inputs, the compilation process translates the static information of cells coded by DNA sequences into a dynamical model of the development of a phenotype in response to environmental influences [128].

Table 6.1: Glossary of specialized terms used throughout this article.

Attribute grammar	An attribute grammar is a context free grammar augmented with attributes, semantic rules, and conditions. Attribute grammars were developed as a means of formalizing the semantics of a context free grammar.
Context free grammar	A context free grammar is a quadruple $(V, \sum, P, S)$ where V is a finite set of non-terminal symbols, $\sum$ (the alphabet) is a finite set of terminal symbols, P is a finite set of rules, and S is a distinguished element of V called the start symbol. A rule P is of the following form $A \rightarrow \omega$ where A is a single non-terminal symbol and ω is a string of terminals and/or non-terminals (possibly empty). The term "context-free" expresses the fact that non-terminals are rewritten without regard to the context in which they occur.
Cusp bifurcation	A codimension 2 bifurcation formed by the tangential meeting of two loci of saddle-node bifurcations. In other words, a cusp bifurcation traces the path of the points bounding a bistable region as they change with changes in two parameters. Bistability is implied within the cusp bounds.
Direct left recursion	A direct left recursion in context free grammar refers to rules of the form $A \rightarrow A\omega$. Parsing left recursion can possibly lead the parser down an infinite branch of the search tree in the corresponding logic program.
PoPS	The measurement of polymerase per second transcribing past a defined point of DNA.
SBML	The Systems Biology Markup Language (SBML) is a machine-readable language, based on XML, for representing models of biochemical reaction networks.
Semantics	Semantics reveals the meaning of syntactically valid strings in a language. For natural languages, this means correlating sentences and phrases with the objects, thoughts, and feelings of our experiences. For programming languages, semantics describes the behavior that a computer follows when executing a program in the language.
Syntax	Syntax refers to the ways symbols may be combined to create well-formed sentences (or programs) in a language. Syntax defines the formal relations between the constituents of a language, thereby providing a structural description of the various expressions that make up legal strings in the language. Syntax deals solely with the form and structure of symbols in a language without any consideration given to their meaning.

The translation of a gene network model from a genetic sequence is very similar to the compilation of the source code of a computer program into an object code that can be executed by a microprocessor (Figure 6.1). The first step consists in breaking down the DNA sequence into a series of genetic parts by a program called the lexer or scanner. Since the sequence of a part may be contained in the sequence of another part, the lexer is capable of backtracking to generate all the possible interpretations of the input DNA sequences as a series of parts. All possible combinations of parts generated by the lexer are sent to a second program called the parser to analyze if they are structurally consistent with the language syntax. The structure of a valid series of parts is represented by a parse tree [29] (Figure 6.2). The semantic evaluation takes advantage of the parse tree to translate the DNA sequence into a different representation such as a chemical reaction network. The translation process requires attributes and semantic actions. Attributes are properties of individual genetic parts or combinations of parts. Semantic actions are associated with the grammar production rules. They specify how attributes are computed. Specifically, the translation process relies on the semantic actions associated with parse tree nodes to synthesize the attributes of the construct from the attributes of its child nodes, or to inherit the attributes from its parental node. In our implementation, the product of the translation is a mass action model of the network of molecular interactions encoded in the DNA sequence. By using the standardized format of Systems Biology Markup Language (SBML), the model can be analyzed using existing simulation engines [1, 84, 93].

6.3 Results

Compilation of a DNA sequence

We have developed a simple grammar compact enough to be presented extensively, yet sufficiently complex to represent basic epistatic interactions. The grammar generates constructs com- posed of one or more gene expression cassettes. The gene expression cassettes are themselves composed of a promoter, cistron, and transcription terminator. Finally, a cistron is composed of a Ribosome Binding Site (RBS) and a coding sequence (gene). The syntax is composed of 12 production rules (P1 to P12) displayed in bold characters in Figure 6.3 where each entry is composed of a rewriting rule (bold), and semantic actions (curly brackets). The symbol e refers to an empty string, [,] to a list, [] to an empty list, and the '+' sign indicates the concatenation operation on two lists. This syntax is comparable to the one described previously [29] except that we introduced the extra non- terminal restConstructs to allow the generation of constructs with multiple cassettes without introducing parsing problems due to direct left recursions [145].

The attributes of a part include the kinetic rates related to this part and the interaction information. For example, the attributes of a promoter include a transcription rate along with

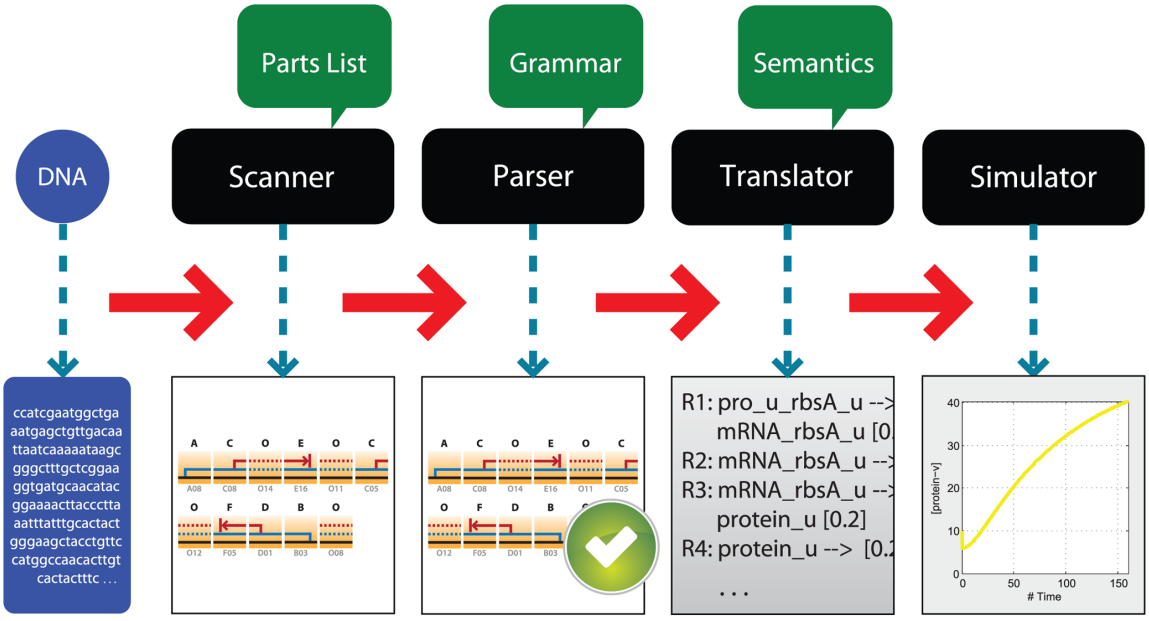


Figure 6.1: Workflow of generating the gene network model encoded in a DNA sequence. The input for this process is a DNA sequence that is first broken down into parts by the scanner. The combination of the parts is validated by the parser according to a syntactic model. After validation by the parser, the sequence is translated by applying semantic actions attached to the rules to transform the series of parts into a set of chemical equations. The resulting equations can then be solved using existing simulation engines. Each step takes the output of the previous step as input, so the workflow can start from any step if the appropriate input is provided.

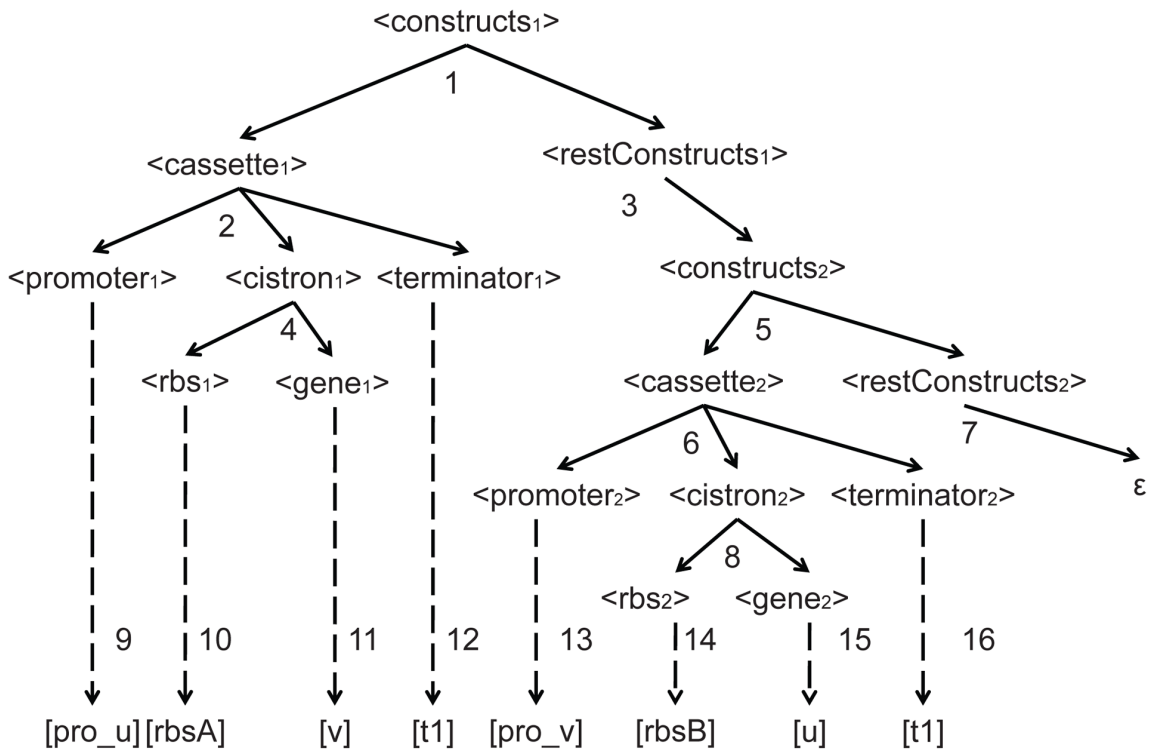


Figure 6.2: Parse tree showing the derivation process of a two-cassette genetic construct. In the derivation tree, terms in ,. corresponds to the non-terminals in the grammar, while terms in [ ] are terminals, and the dashed lines indicate the transformation to terminals. The subscripts are used to distinguish different instances of the same category.

P1. $constructs \rightarrow cassette, restConstructs\{$ $constructs.promoter\_list = cassette.promoter\_list + restConstructs.promoter\_list$ $constructs.equation\_list = cassette.equation\_list + restConstructs.equation\_list$ $cassette.protein\_list = constructs.protein\_list$ $restConstructs.protein\_list = constructs.protein\_list\}$
P2. $restConstructs \rightarrow constructs\{$ $restConstructs.promoter\_list = constructs.promoter\_list$ $restConstructs.equation\_list = constructs.equation\_list$ $constructs.promoter\_list = restConstructs.protein\_list\}$
P3. $restConstructs \rightarrow \varepsilon\{$ $restConstructs.promoter\_list = [\,]$ $restConstructs.equation\_list = [\,]$ $restConstructs.protein\_list = [\,]\}$
P4. $cassette \rightarrow promoter, cistron, terminator\{$ $cassette.promoter\_list = [promoter.name, cistron.transcript]$ $cassette.equation\_list = cistron.equation\_list + promoter\_protein\_interaction($ $cassette.promoter\_list, cassette.protein\_list) + transcription(promoter, cistron.transcript)\}$
P5. $cistron \rightarrow rbs, gene\{$ $cistron.transcript = rbs.name + gene.name$ $cistron.equation\_list = translation(rbs, gene)\}$
P6. $promoter \rightarrow pro\_u\{$ $promoter.name = [pro\_u]$; $promoter.transcription\_rate = k_1$ $promoter.leakiness\_rate = k_{11}$; $promoter.repressor\_list = [(u, 2, k_9, k_{9r})]$
P7. $promoter \rightarrow pro\_v\{$ $promoter.name = [pro\_v]$; $promoter.transcription\_rate = [k_2]$ $promoter.leakiness\_rate = [k_{12}]$; $promoter.repressor\_list = [(v, 4, k_{10}, k_{10r})]\}$
P8. $rbs \rightarrow rbsA\{$ $rbs.name = [rbsA]$; $rbs.translation\_rate = [k_3]\}$
P9. $rbs \rightarrow rbsB\{$ $rbs.name = [rbsB]$; $rbs.translation\_rate = [k_4]\}$
P10. $gene \rightarrow u\{$ $gene.name = [u]$; $gene.mRNA\_degradation\_rate = [k_5]$ $gene.protein\_degradation\_rate = [k_7]\}$
P11. $gene \rightarrow v\{$ $gene.name = [v]$; $gene.mRNA\_degradation\_rate = [k_6]$ $gene.protein\_degradation\_rate = [k_8]\}$
P12. $terminator \rightarrow t_1\{$ $terminator.name = [t_1]\}$

Figure 6.3: An example of attribute grammar.

Table 6.2: Attributes associated with non-terminals.

Non-terminals	Inherited Attribute	Synthesized Attributes
constructs	protein_list	promoter_list, equation_list
cassette	protein_list	promoter_list, equation_list
restConstructs	protein_list	promoter_list, equation_list
cistron	protein_list	transcript, equation_list
promoter	-	name, transcription_rate, leakiness_rate, repressor_list
RBS	-	name, translation_rate
gene	-	name, mRNA_degradation_rate, protein_degradation_rate
terminator	-	name

a list of proteins repressing it and the kinetic parameters of the protein-DNA interactions. For non-terminal variables corresponding to combinations of parts such as cistrons, the attributes include a list of proteins, a list of promoters, and a list of chemical equations. The equation list is used to store the model of the system behavior, while the lists of promoters and proteins are recorded for computing the molecular interactions resulting from the DNA sequence. The complete set of attributes used in this simple grammar is listed in Table 6.2.

If many attributes can be computed locally by only considering a small fragment of the DNA sequence, other attributes are global properties of the system. For instance, the computation of protein-DNA interactions requires access to a global list of proteins expressed by the constructs. However, this list is not available until all of the different cassettes have been parsed. The problem is overcome by using a multiple-pass compilation method. In the first pass, the compiler does not do any structural validation but builds the list of proteins in the system and passes the list as an inherited attribute to the second pass. In the second pass, the promoter-protein interactions can be calculated locally at the level of each cassette. Rules P1 to P5 define the structure of a design, while rules P6 to P12 cover the selection of a specific part for each category. In the semantic action, the relation between an attribute and its variable is indicated by a dot and constants are enclosed by brackets. For instance, $gene.mRNA\_degration\_rate = [k6]$ indicates that the value of the attribute $mRNA\_degration\_rate$ of a gene is a constant k6. The attribute $repressor\_list$ used in P6 and P7 includes the name of the repressor, the stoichiometry, and the kinetic constants of the forward and reverse reactions of the protein-DNA interaction. Table S1 details the parsing steps and computational dependence of each step. Finally, the equation writing operations are handled by functions typed in italics in Figure 6.3 and defined in Figure 6.4.

The translation of the DNA sequence into a mathematical model is available as the $equation_list$

Generators	Parameters	Equations

promoter_protein_interaction	promoter_list,
	protein_list
		for [promoter, transcript] in promoter_list {
		for [repressor, binding_rate, release_rate] in
		promoter.repressor_list {
		if repressor is in protein_list {
		promoter_transcript + repressor ↔
		promoter_transcript_x(binding_rate,release_rate)
		promoter_transcript_x → promoter_transcript_x +
		mRNA_transcript(promoter.leakiness_rate)
		} endif
		} end
		} end

transcription	promoter, transcript	promoter_transcript → promoter_transcript + mRNA_transcript (promoter.transcription_rate) mRNA_transcript → Ø (transcript.mRNA_degradation_rate)
translation	rbs, gene	mRNA_rbs_gene → mRNA_rbs_gene + protein_gene (rbs.translation_rate) protein_rbs_gene → Ø (gene.protein_degradation_rate)

Figure 6.4: Equation generators.

[Reactions]
R1: $pro\_u\_rbsA\_v \rightarrow pro\_u\_rbsA\_v + mRNA\_rbsA\_v\,[k_1]$
R2: $mRNA\_rbsA\_v \rightarrow \phi\,[k_6]$
R3: $mRNA\_rbsA\_v \rightarrow mRNA\_rbsA\_v + protein\_v\,[k_3]$
R4: $protein\_v \rightarrow \phi\,[k_8]$
R5: $pro\_v\_rbsB\_u \rightarrow pro\_v\_rbsB\_u + mRNA\_rbsB\_u\,[k_2]$
R6: $mRNA\_rbsB\_u \rightarrow \phi\,[k_5]$
R7: $mRNA\_rbsB\_u \rightarrow mRNA\_rbsB\_u + protein\_u\,[k_4]$
R8: $protein\_u \rightarrow \phi\,[k_7]$
R9: $pro\_v\_rbsB\_u + 4protein\_v \leftrightarrow pro\_v\_rbsB\_u\_x\,[k_{10}, k_{10r}]$
R10: $pro\_v\_rbsB\_u\_x \rightarrow pro\_v\_rbsB\_u\_x + mRNA\_rbsB\_u\,[k_{12}]$
R11: $pro\_u\_rbsA\_v + 2protein\_u \leftrightarrow pro\_u\_rbsA\_v\_x\,[k_9, k_{9r}]$
R12: $pro\_u\_rbsA\_v\_x \rightarrow pro\_u\_rbsA\_v\_x + mRNA\_rbsA\_v\,[k_{11}]$

[Initial Values]
$pro\_u\_rbsA\_v = 1$
$pro\_v\_rbsB\_u = 1$
$protein\_v =$ user input
$protein\_v =$ user input

Figure 6.5: Chemical equations translated from a DNA sequence.

attribute of constructs. The model outputs are generated by equations generators, which are purposely decoupled from the semantic actions. The decoupling enables the flexibility of using different equation formats to describe a biological process. The translation of the construct composed of the parts *pro_u* rbsA *gene_v* t1 *pro_v* rbsB *gene_u* t1 generates the equations displayed in the [Reactions] section of Figure 6.5. Each line is composed of a reaction index (R1 to R12), the chemical equation itself, and one or two reaction parameters depending on the reaction reversibility. The initial values have been computed by assigning 1 to variables representing DNA sequences and prompting the user to set the initial condition of proteins. The scripts and data used in this report are available in Dataset S1.

Expressing context-dependencies of parts function

The semantic model presented in the previous section is completely modular since the parameters of the model describing the construct behavior are attributes of individual parts, not of higher order structures. For instance, in the previous model (Figures 3 and 4), translational efficiency is primarily determined by the RBS sequence [196, 214]. This association between RBS and translation rate was successfully used to design one of the first artificial gene networks [72] and is still used by many synthetic biology software applications [92, 137, 158, 175]. Yet, it is also well known that translation initiation can be attenuated by stable mRNA secondary structures [50, 52, 121]. This leads to a situation where a translational rate can no

longer be considered the attribute of an individual part but needs to be considered as the attribute of a specific combination of parts. This type of context-dependency can naturally be expressed using attribute grammars since the translation reaction is computed at the cistron level, not at the level of individual parts. Rule P5 of Figure 6.3 can be modified by introducing a new function to retrieve the translation rate for specific combination of gene and RBS.

```
P5. cistron -> rbs, gene {
cistron.translation_rate = get_translation_rate (rbs, gene)
cistron.transcript = rbs.name+gene.name
cistron.equation_list = translation(rbs, gene, cistron.translation_rate)
}
```

The *get_translation_rate* function checks for specific cases of interactions between an RBS and coding sequence first. If none is found, then the default RBS translation rate is used.

```
If exists translation_rate(rbs, gene)
translation_rate = translation_rate(rbs, gene)
else
translation_rate = translation_rate(rbs)
endif
```

This approach is illustrated in Table 6.3 using previously published data demonstrating the interference between the RBS and coding sequence [50]. Specifically, this report provides the relation expression observed in 23 different constructs generated by combining different variants of the RBS and MS2 coat protein gene. This data set has been reorganized in Table 6.3 by sorting the constructs according to the RBS and gene variants they used. Three of the constructs using the WT RBS sequence resulted in a maximum level of expression while the expression of the gene variants ORF4, ORF5, and ORF6 were expressed at a much lower level due to the greater stability of the mRNA secondary structure. A similar pattern is observed for other RBS variants (RBS1, RBS2, RBS3, RBS7). For all of these RBS variants, it is possible to define the translation_rate function by associating the default translation rate with the maximum expression rate. Specific translation rates associated with particular pairs of RBS and gene variants are recorded separately.

Exploration of genetic design space

The semantic model in Figures 3 and 4 is a compact proof of concept example, but it does not capture a number of features commonly found in actual genetic constructs. In order to demonstrate that our approach is capable of modeling more realistic DNA sequences, we have

Table 6.3: Context-dependency of experimentally determined translation rates.

Mutant	RBS	ORF	Expression	Translation rate function
1	RBS WT	ORF WT	100	*translation_rate*(RBS WT)
6	RBS WT	ORF2	100	*translation_rate*(RBS WT)
7	RBS WT	ORF3	100	*translation_rate*(RBS WT)
17	RBS WT	ORF4	3	*translation_rate*(RBS WT, ORF4)
20	RBS WT	ORF5	6	*translation_rate*(RBS WT, ORF5)
23	RBS WT	ORF6	0.3	*translation_rate*(RBS WT, ORF6)
4	RBS1	ORF WT	100	*translation_rate*(RBS1)
2	RBS1	ORF1	100	*translation_rate*(RBS1)
3	RBS1	ORF2	100	*translation_rate*(RBS1)
5	RBS1	ORF3	4	*translation_rate*(RBS1, ORF3)
14	RBS1	ORF4	< 0.003	*translation_rate*(RBS1, ORF4)
9	RBS2	ORF WT	100	*translation_rate*(RBS2)
8	RBS2	ORF1	100	*translation_rate*(RBS2)
10	RBS2	ORF3	100	*translation_rate*(RBS2)
12	RBS3	ORF WT	100	*translation_rate*(RBS3)
11	RBS3	ORF1	20	*translation_rate*(RBS3, ORF1)
13	RBS3	ORF3	100	*translation_rate*(RBS3)
15	RBS4	ORF4	0.1	*translation_rate*(RBS4)
16	RBS5	ORF4	0.05	*translation_rate*(RBS5)
22	RBS6	ORF WT	0.2	*translation_rate*(RBS6, ORF WT)
18	RBS6	ORF4	80	*translation_rate*(RBS6)
21	RBS7	ORF WT	100	*translation_rate*(RBS7)
19	RBS7	ORF4	100	*translation_rate*(RBS7)

extended this semantic model (Supplementary Materials) to translate the DNA sequences of previously published DNA plasmids that include polycistronic cassettes in different orientations [72]. This plasmid library was generated by 32 different genetic parts (three promoters: pLtetO-1, pLs1con, ptrc-2; eight RBS: rbsA to rbsH; and four genes: tetR, cIts, lacI, and gfp and one terminator, all in both orientations). The syntax generates 72 different single gene expression constructs in each orientation. By combining two genes repressing each other in a construct, it is possible to make bistable artificial gene networks that are represented in Figure 6.6. These bistable networks can be used as a genetic switch.

To demonstrate the potential use of a semantic model to search for a desirable behavior in a large genetic design space, we have generated the DNA sequences of all 41,472 possible sequences (72268 RBS for the reporter gene) having the same structure as previously described switches. All sequences were translated into separate model files and a script was developed to perform a bistability analysis of each model. Parameters of the semantic model were obtained by qualitatively matching the experimental results of the six previously published switches [72] and are summarized in Table S2. Most of the automatically generated sequences led to inherently non-bistable networks because the necessary repressor/promoter pairs did not match. Since this specific example is particularly well understood, we could have generated a limited number of targeted constructs. Yet, we chose to generate all possible sequences to demonstrate the generality of our approach. In particular, it was important to evaluate the computational cost of generating and translating DNA sequences to ensure that it would not prevent a systematic exploration of more complex design spaces. It takes only minutes to generate 41,472 sequences and translate them into SBML files. Hence, the computational cost of this step is negligible compared to the time required by the simulation of the SBML files.

Bistability was tested numerically by integrating the differential equations until they converged to a steady state starting from two different initial conditions. The two initial conditions started with one protein level very high and the other very low and vice versa. We characterized the bistability by computing the ratio of reporter concentration for the two steady state values. In order to globally verify the behavior of this large population of models, we focused on the 3,072 constructs potentially capable of bistability, 1,408 of which were found to be bistable. We further reduced the number of constructs used to verify the translation process from 3,072 to 384 by assuming that two constructs differing only in the RBS in 59 of the reporter gene would produce the same ratio of steady state values. Figure 6.6 visualizes the behavior of these 384 constructs. Constructs that are not bistable have a ratio of 1. This ratio gives insight into how the construct is expected to be experimentally detectable. Since most experimental methods cannot give an exact value of protein concentration, a high ratio is desired to rise above experimental noise. Each of the 6 windows is analogous to the previously described two-parameter bifurcation diagram for that pair of repressors [72]. This gives confidence that both the semantic model of DNA sequences and

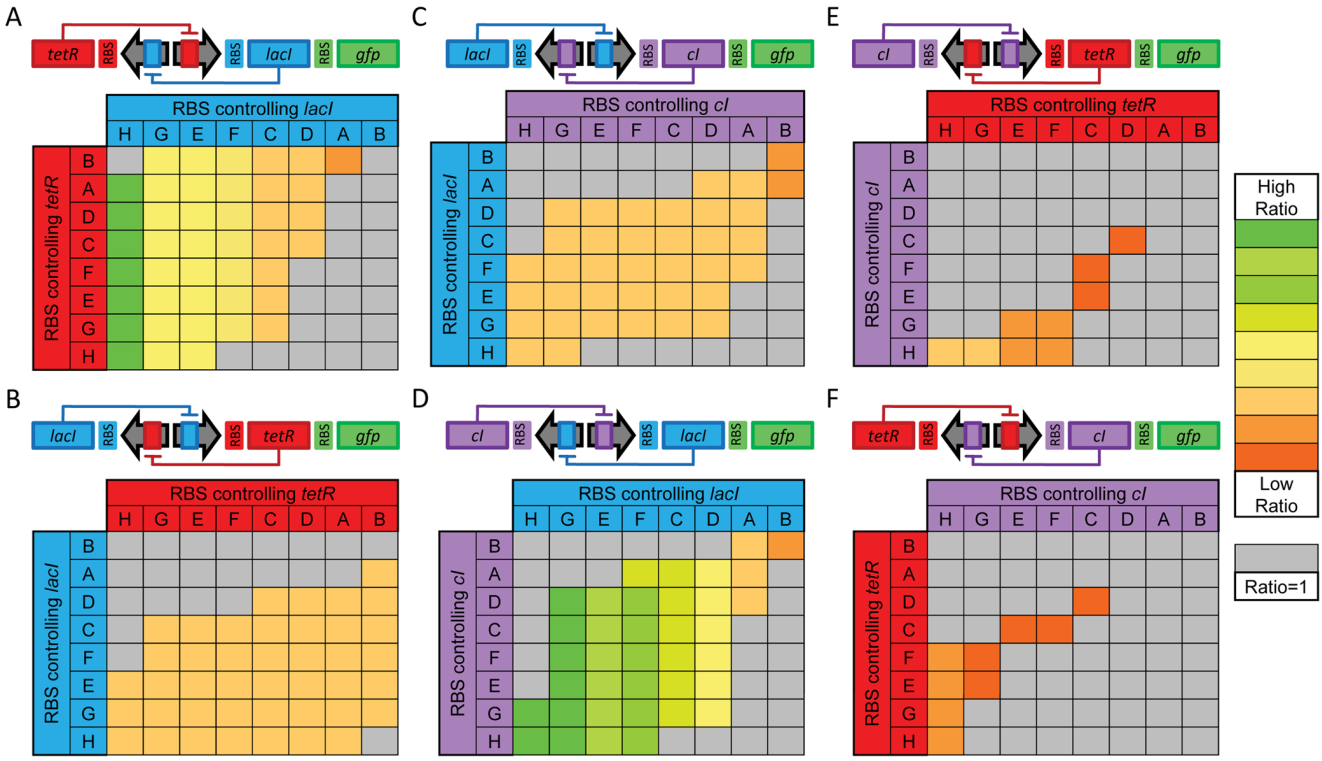


Figure 6.6: Mapping the behavior of 384 genetic constructs. Each section A to F indicates a different selection of repressors within a toggle switch: (A) tetR and lacI, (B) lacI and tetR, (C) lacI and cI, (D) cI and lacI, (E) cI and tetR, and (F) tetR and cI. Other networks that cannot give rise to bistability (e.g. a construct with tetR as both genes) are excluded as are designs that only vary the GFP RBS (see text). Each pair is explored by varying the RBS (ordered by translational efficiency from low (RBS H) to high (RBS B) as determined by a qualitative fit of the results of Gardner et al. [72] with consistent letter-based labels) and calculating the detectability ratio, defined as the steady state GFP concentration in the "on" state divided by the concentration in the "off" state. These ratios are displayed using a color map as indicated by the legend to the right. Monostable constructs have a ratio of 1 and are indicated by gray boxes. The ratio gives a measure of how easily the two steady states can be distinguished, which is important due to high experimental noise. Each pane also elucidates the traditional two-parameter bifurcation diagram of each gene pair as the translational rates are varied by changing RBSs. Constructs near the edge of the cusp operate near saddle-node bifurcations and are more prone to noise-induced switching. Thus, constructs from the cusp interior are preferred for robust behavior.

the compiler used to translate automatically generated DNA sequences give results consistent with manually developed models of this family of gene networks. In the long term, the advantage to our approach over a traditional two-parameter bifurcation is the association of discrete parameter values with specific parts. This will prove particularly valuable when the context-dependencies of parameter values are better documented experimentally.

This example demonstrates the benefit of building a semantic model of synthetic DNA sequences. Even a small library of genetic parts can generate large numbers of artificial gene networks having no more than a few interacting genes. A syntactic model describing how parts can be combined into constructs is a compact representation of the genetic design space generated from the parts library. While it is possible to manually build mathematical models capturing the dynamics of some of these artificial gene networks individually, it becomes desirable to automate the process to ensure the model consistency when building large families of related models derived from the same parts library. By considering genetic parts as the terminal symbols of an attribute grammar, it becomes possible to automatically generate models of numerous artificial gene networks derived from this parts library and quickly identify the optimal designs [80].

6.4 Discussion

Computer assisted design of synthetic genetic constructs

The parameter values used in the previous example were selected to match an extremely small set of six experimental data points. Although the under-determination of the model does not make it possible to precisely estimate the value of these parameters, the example illustrates how the framework could provide valuable guidance in selecting specific parts for a design. Considering that the exact value of parameters for parts is still a far off perspective, the automatic exploration of the design space presented here will provide useful guidance in construct design. For example, robust constructs from the cusp interior of the tetR/cI and lacI/cI pairings could be built and tested while less robust switches based on the lacI/tetR pairing would be avoided. As more is learned about these parts including the specific rates in different genetic contexts, the predictive ability of such maps will increase. Other motifs could be explored in a similar manner. For example, oscillators [205] could be explored by permuting parts and calculating the model-predicted existence of oscillations as well as their period or amplitude.

The approach presented in this report will be implemented into GenoCAD [48], the web-based tool we have developed to give biologists access to our syntactic design framework. Through GenoCAD, users will benefit from the syntactic and semantic models of various parts sources

(GenoCAD provided library, MIT Registry of Standard Biological Parts, or user created parts library). Initially, users will be able to translate their designs into SBML files that could be imported in SBML-compliant simulation tools (`www.sbml.org/SBML_Software_Guide`) for further analysis. At a later stage, simulation results and more advanced numerical analyses will be seamlessly integrated in GenoCAD's workflow. One of the major obstacles toward the implementation of such semantic models in GenoCAD is the development of a data model allowing users to understand and possibly edit the functional model of the parts they use.

A function description language called Genetic Engineering of living Cells (GEC) was recently introduced to specify the properties of a design [158]. GEC is capable of finding a DNA sequence that implements the desirable phenotypic functions. Several other software applications have been recently released to design biological systems from standardized genetic parts. ASMPART [175], SynBioSS [92], a specialized ProMot package [137] and TinkerCell (`www.tinkercell.com`) illustrate this trend. These tools are still exploratory. One of their limitations is the requirement to define parts in a specialized format, such as SBML or Modeling Description Language (MDL). Furthermore, instead of defining parts interactions in the underlying parts data models, these tools rely on the user to manually define them textually [92] or graphically [137]. As a result of this specific limitation, several of these tools do not appear suitable for the automatic exploration of a design space. Moreover, they tend to rely on a loosely defined relationship between the structure of the genetic constructs and their behavior. They allow parts to be assembled in any order without regard for biological viability.

Still, the scripts developed to generate our results are of lesser importance than the application of the theory of semantics-based translation using attribute grammars to the translation of DNA sequences into dynamical models representing the molecular interactions they encode. Since this approach is used to develop the compilers of many computer languages [7, 198], a wealth of existing theoretical results and software tools can find new applications in the life sciences. For instance, we have implemented semantic models of DNA sequences into two widely used but very different programming environments, Prolog [22] and ANTLR [153]. Future research efforts will need to investigate the pros and cons of different compiler generators and different parsing algorithms for analyzing even genome-scale DNA sequences and how they impact the ability of grammars to express various features of DNA sequences. Also, the type of attributes associated with parts is flexible. Here we primarily use mass action kinetic rates as attributes, but we could just as easily have used an emerging synthetic biology unit like polymerase per second (PoPS) [32, 111].

Ultimately, tools capable of automatically generating models of the behavior of synthetic DNA sequences will be important for the advancement of synthetic biology [80]. However, these tools will need to be able to express that the contribution of a genetic part to the phenotype of an organism depends largely on the local and global context in which it is placed.

The interference between RBS and coding sequence is just one example of the biological complexity that computer assisted design applications will have to properly consider.

Functional characterization of genetic parts

Before it will be used to build synthetic genetic systems meeting user-defined specifications, the semantic model of DNA sequences presented in this report will be instrumental in the quantitative characterization of structure-function relationships in synthetic DNA sequences. The vision of applying quantitative engineering methods to biological problems has been recognized as a promising avenue to biological discovery [55]. The critical role of artificial gene networks in the characterization of molecular noise affecting the dynamics of gene networks [165] illustrates the potential of synthetic biology as a route to refine the understanding of basic biological processes.

Ongoing efforts aim to carefully define how parts should fit together syntactically and what attributes are needed to characterize their function. For example, the sequence between the RBS and the start codon has been shown to play an important role in translation rate [214]. The question arises whether the RBS should be defined to include the spacing, or if there should be a separate parts category for the spacer. The rapid development of gene synthesis techniques [47] will make it possible to investigate these questions with a base-level resolution. Beyond libraries of parts for designing expression vectors, similar curation efforts could lead to the identification of parts in genomic sequences, whereby the hypothetical function of these parts as they are expressed in attribute grammars could be tested by genome refactoring [35].

6.5 Supporting Information

Table S1 Computation dependence corresponding to the derivation tree in Fig. 2 The computation starts from the leaves of the tree, and the semantic values computed are transferred to upstream nodes. The computation of each node cannot proceed until all of its sub-trees are computed. For example, the computation of semantic values of $< constructs1 >$ (2) is pending until its subtrees $< cassette1 >$ (3) and $< restConstructs1 >$ (4) are computed.
Found at: doi:10.1371/journal.pcbi.1000529.s001 (0.01 MB PDF)

Table S2 List of parts used in the "exploration of genetic space" section and values of associated attributes.
Found at: doi:10.1371/journal.pcbi.1000529.s002 (0.01 MB PDF)

Dataset S1 Zip file containing the scripts and data used in this report.

Found at: doi:10.1371/journal.pcbi.1000529.s003 (0.03 MB ZIP)

6.6 Acknowledgments

The authors would like to thank Drs. Jacques Cohen and Mark Cooper for their critical reading of an early version of this manuscript, Stephan Hoops for helping us use COPASI in batch mode, and Emily Alberts for her editorial skills.

6.7 Author Contributions

Conceived and designed the experiments: JP. Performed the experiments: YC MWL LA. Analyzed the data: YC MWL JP. Wrote the paper: YC MWL JP.

6.8 Attribution

Yizhi Cai – Graduate student (Genetics, Bioinformatics and Computational Biology interdisciplinary doctoral program, Virginia Tech) performed the experiments, and wrote the paper.

Matthew W. Lux – Graduate student (Genetics, Bioinformatics and Computational Biology interdisciplinary doctoral program, Virginia Tech), performed the experiments, and wrote the paper.

Laura Adam – Graduate student (Genetics, Bioinformatics and Computational Biology interdisciplinary doctoral program, Virginia Tech), performed the experiments.

Jean Peccoud – Ph.D (Department of Molecular Biology and Bioinformatics, Universit Joseph Fourier (Grenoble I)) currently an associate professor of Virginia Bioinformatics Institute at Virginia Tech. JP conceived and designed the experiments, and wrote the paper.

Chapter 7

Outlooks and Perspectives

Standardization of parts is one of the most important concepts in synthetic biology. The MIT Registry of standard biological parts is quite successfully in implementing this idea, evidenced by the largest parts collections (around 10,000 parts at this moment) and the largest user base (more than 1000 new users per year). In the work presented here, two chapters are directly related to the Registry: a comprehensive analysis of the Registry is presented at Chapter 3, and a domain-specific grammar for BioBricks has been developed and presented in Chapter 5. There are a lot of efforts and motivations to improve the Registry, or even to build new Registries to better support the community of synthetic biology. Stanford University hosts a BioBrick Parts Catalog (`www.biobrickparts.org`) which provides Application Programming Interface (API) to support the third-party software development. Joint BioEnergy Institute (JBEI) is building yet another Registry of parts – JBEIR (`https://www.jbeir.org/`), and recently it has been chosen as the CAD tool for the BioFab project. As more and more parts and Registries are available to the public, there are a few issues the whole community should think carefully to address so that together we can develop a better home for parts, which ultimately will facilitate the development of synthetic biology. Right now the available information of a part is sequence-centric, but the information of its functionality is somehow scarce. For example, what's the primary function of a particular part (in what context)? how is this part measured? how reliable is this part (in a context sense and also in an evolutionary sense)? This type of information is as important as the sequence itself. Kelly *et al.* undertook a pioneering step in measuring the transcription strength of BioBrick promoters using an *in vivo* reference standard in different laboratories world wide [111]. Canton *et al.* calibrate a particular BioBrick device BBa_F2620 and reported the result in a proposed datasheet [32]. As the calibration of parts progresses, attributes of parts will be available for developing the semantic models described in Chapter 6. The next question will be how do we associate those attributes with parts, or how do we represent parts? Standardizing the part representation will make it much easier for data exchange and communication between different registries. Cooling *et al.* recently proposed to use CellML [130, 131] to describe parts, which makes it possible to develop mathematical

models for biological systems in a modular manner [45]. However, it should be noted that CellML is ODE (Ordinary Differential Equation) based, which makes it incompatible with stochastic simulation. Another option to represent parts is to utilize the ontology techniques being developed by the bioinformatics community [15, 59]. Sharing information, synchronizing data among registries or even interfacing existing bioinformatics databases such as RegulonDB [98] will be a very challenging task as well. The community will need to develop a protocol for sharing parts information which allows for dynamical data integration. The distributed Annotation System (DAS) [105] which has been widely used in both the genome and protein bioinformatics communities, seems to be a good starting point towards this goal.

GenoCAD itself is undergoing the second phase of development. One significant new feature is the integration of the semantic model presented in Chapter 6 into the GenoCAD web application. This feature will allow users to design a biological sequence in GenoCAD, translate the sequence into an SBML file using the built-in compiler, and run a computer simulation to check whether its performance fits the user's expectation. However, the implementation of this feature is non-trivial. The main hindrance is to find a suitable data model for presenting the semantic models. As the parts information is part of a semantic model, the model should have the capability to be dynamically updated to reflect the parts information in the database. A proof-of-concept semi-dynamic compiler has been implemented thus far, but a considerable amount of effort is still required to make it production grade. The rule-based design in GenoCAD is very useful and elegant in guiding small scale genetic constructs. However, it is difficult to scale up the design for applications such as an artificial yeast chromosome. One possible approach is to allow free-style design and use the built-in compiler to debug the genetic code. This is similar to computer programming: most sophisticated programming environments allow programmers to write source code as they like, but provide a compiler for them to debug the code. Design optimization is also essential for CAD tools, and we can consider the integration of some optimization methods such as codon optimization into GenoCAD to enhance the gene expression of a design. Finally, we also would like to explore the new research direction of grammatical inference. At this moment, all the grammatical models in GenoCAD were built by human experts with necessary domain specific knowledge. We would like to employ learning algorithms being developed in computational linguistics to automate the knowledge mining process.

We have shown here that the field of synthetic biology holds great promise to shed light on fundamental questions in life science and improve the living conditions for human beings. We introduced the linguistic methodologies to design DNA sequences, and developed a computer assisted design tool (GenoCAD) for the synthetic biology community. In the work presented in this dissertation, both syntactic and semantic aspects of synthetic DNA sequences have been explored. The results suggest that linguistic approaches will find new applications in the area of synthetic biology, and GenoCAD will play an important role in the design and verification of synthetic DNA sequences.

Bibliography

[1] D. Adalsteinsson, D. McMillen, and T. C. Elston. Biochemical network stochastic simulator (bionets): software for stochastic modeling of biochemical networks. *BMC Bioinformatics*, 5:24, Mar 2004.

[2] K. L. Agarwal, H. Büchi, M. H. Caruthers, N. Gupta, H. G. Khorana, K. Kleppe, A. Kumar, E. Ohtsuka, U. L. Rajbhandary, J. H. V. de Sande, V. Sgaramella, H. Weber, and T. Yamada. Total synthesis of the gene for an alanine transfer ribonucleic acid from yeast. *Nature*, 227(5253):27–34, Jul 1970.

[3] C. Ajo-Franklin, D. A. Drubin, J. Eskin, and E. Gee. Rational design of memory in eukaryotic cells. *Genes and Development*, Jan 2007.

[4] J. Aleksic, F. Bizzari, Y. Cai, B. Davidson, K. D. Mora, S. Ivakhno, S. Seshasayee, J. Nicholson, J. Wilson, and A. Elfick. Development of a novel biosensor for the detection of arsenic in drinking water. *Synthetic Biology, IET*, 1(1.2):87–90, 2007.

[5] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *J Mol Biol*, 215(3):403–10, Oct 1990.

[6] E. Andrianantoandro, S. Basu, D. K. Karig, and R. Weiss. Synthetic biology: new engineering rules for an emerging discipline. *Mol Syst Biol*, 2:2006.0028, Jan 2006.

[7] A. W. Appel and J. Palsberg. Modern compiler implementation in java. *Cambridge University Press*, Jan 2002.

[8] A. P. Arkin and D. A. Fletcher. Fast, cheap and somewhat in control. *Genome biology*, 7(8):114, Jan 2006.

[9] M. R. Atkinson, M. A. Savageau, J. T. Myers, and A. J. Ninfa. Development of genetic circuitry exhibiting toggle switch or oscillatory behavior in escherichia coli. *Cell*, 113(5):597–607, May 2003.

[10] W. Bains. The parts list of life. *Nat Biotechnol*, 19(5):401–2, May 2001.

[11] P. Ball. Synthetic biology: starting from scratch. *Nature*, 431(7009):624–6, Oct 2004.

[12] P. Ball. Synthetic biology: designs for life. *Nature*, 448(7149):32–3, Jul 2007.

[13] S. Basu, R. Mehreja, S. Thiberge, M.-T. Chen, and R. Weiss. Spatiotemporal control of gene expression with pulse-generating networks. *Proc Natl Acad Sci USA*, 101(17):6355–60, Apr 2004.

[14] T. S. Bayer and C. D. Smolke. Programmable ligand-controlled riboregulators of eukaryotic gene expression. *Nat Biotechnol*, 23(3):337–43, Mar 2005.

[15] E. Beisswanger, V. Lee, J.-J. Kim, D. Rebholz-Schuhmann, A. Splendiani, O. Dameron, S. Schulz, and U. Hahn. Gene regulation ontology (gro): design principles and use cases. *Studies in health technology and informatics*, 136:9–14, Jan 2008.

[16] S. Benner and A. Sismour. Synthetic biology. *Nat Rev Genet*, Jan 2005.

[17] S. A. Benner and A. M. Sismour. Synthetic biology. *Nat Rev Genet*, 6(7):533–43, Jul 2005.

[18] S. Bentolila. A grammar describing 'biological binding operators' to model gene regulation. *Biochimie*, 78(5):335–50, Jan 1996.

[19] C. Berens and W. Hillen. Gene regulation by tetracyclines. constraints of resistance regulation in bacteria shape tetr for application in eukaryotes. *Eur J Biochem*, 270(15):3109–21, Aug 2003.

[20] N. S. Berrow, D. Alderton, S. Sainsbury, J. Nettleship, R. Assenberg, N. Rahman, D. I. Stuart, and R. J. Owens. A versatile ligation-independent cloning method suitable for high-throughput expression screening applications. *Nucleic Acids Res*, 35(6):e45, Jan 2007.

[21] M. A. Brasch, J. L. Hartley, and M. Vidal. Orfeome cloning and systems biology: standardized mass production of the parts from the parts-list. *Genome research*, 14(10B):2001–9, Oct 2004.

[22] I. Bratko. Prolog programming for artificial intelligence. *Addison Wesley*, page 678, Jan 2001.

[23] A. Brazma, I. Jonassen, I. Eidhammer, and D. Gilbert. Approaches to the automatic discovery of patterns in biosequences. *J Comput Biol*, 5(2):279–305, Jul 1998.

[24] V. Brendel, J. S. Beckmann, and E. N. Trifonov. Linguistics of nucleotide sequences: morphology and comparison of vocabularies. *J Biomol Struct Dyn*, 4(1):11–21, Aug 1986.

[25] V. Brendel and H. Busse. Genome structure described by formal languages. *Nucleic Acids Res*, 12(5):2561–2568, Jan 1984.

[26] M. Brown and C. Wilson. Rna pseudoknot modeling using intersections of stochastic context free grammars with applications to database search. *Pacific Symposium on Biocomputing Pacific Symposium on Biocomputing*, pages 109–25, Jan 1996.

[27] L. Cai, C. K. Dalal, and M. B. Elowitz. Frequency-modulated nuclear localization bursts coordinate gene regulation. *Nature*, 455(7212):485–U16, Jan 2008.

[28] L. Cai, R. L. Malmberg, and Y. Wu. Stochastic modeling of rna pseudoknotted structures: a grammatical approach. *Bioinformatics*, 19:i66–73, Jan 2003.

[29] Y. Cai, B. Hartnett, C. Gustafsson, and J. Peccoud. A syntactic model to design and verify synthetic genetic constructs derived from standard biological parts. *Bioinformatics*, 23(20):2760–7, Oct 2007.

[30] Y. Cai, M. W. Lux, L. Adam, and J. Peccoud. Modeling structure-function relationships in synthetic dna sequences using attribute grammars. *PLoS Comput Biol*, 5(10):e1000529, Oct 2009.

[31] Y. Cai, M. Wilson, and J. Peccoud. Genocad for igem: a grammatical approach to the design of standard-compliant constructs. *Nucleic Acids Res*, page gkq086v1, Feb 2010.

[32] B. Canton, A. Labno, and D. Endy. Refinement and standardization of synthetic biological parts and devices. *Nat Biotechnol*, 26(7):787–93, Jul 2008.

[33] J. Carothers, J. Goler, and J. D. Keasling. Chemical synthesis using synthetic biology. *Curr Opin Biotech*, Aug 2009.

[34] J. Cello, A. V. Paul, and E. Wimmer. Chemical synthesis of poliovirus cdna: generation of infectious virus in the absence of natural template. *Science*, 297(5583):1016–8, Aug 2002.

[35] L. Y. Chan, S. Kosuri, and D. Endy. Refactoring bacteriophage t7. *Mol Syst Biol*, 1:2005.0018, Jan 2005.

[36] D. Chandran, F. T. Bergmann, and H. M. Sauro. Tinkercell: modular cad tool for synthetic biology. *Journal of biological engineering*, 3:19, Jan 2009.

[37] K. C. Chen, L. Calzone, A. Csikasz-Nagy, F. R. Cross, B. Novak, and J. J. Tyson. Integrative analysis of cell cycle control in budding yeast. *Mol Biol Cell*, 15(8):3841–62, Aug 2004.

[38] M.-T. Chen and R. Weiss. Artificial cell-cell communication in yeast saccharomyces cerevisiae using signaling elements from arabidopsis thaliana. *Nat Biotechnol*, 23(12):1551–5, Dec 2005.

[39] D. Chiang, A. K. Joshi, and D. B. Searls. Grammatical representations of macromolecular structure. *J Comput Biol*, 13(5):1077–100, Jun 2006.

[40] J. W. Chin. Programming and engineering biological networks. *Curr. Opin. Struct. Biol.*, 16(4):551–6, Aug 2006.

[41] N. Chomsky. Three models for the description of language. *Information Theory*, Jan 1956.

[42] J. R. Coleman, D. Papamichail, S. Skiena, B. Futcher, E. Wimmer, and S. Mueller. Virus attenuation by genome-scale changes in codon pair bias. *Science*, 320(5884):1784–1787, Jun 2008.

[43] J. Collado-Vides. Grammatical model of the regulation of gene expression. *Proc Natl Acad Sci USA*, 89(20):9405–9, Oct 1992.

[44] S. G. Consortium, C. S. G. Consortium, N. S. G. Consortium, S. Gräslund, P. Nordlund, J. Weigelt, B. M. Hallberg, J. Bray, O. Gileadi, S. Knapp, U. Oppermann, C. Arrowsmith, R. Hui, J. Ming, S. dhe Paganon, H. won Park, A. Savchenko, A. Yee, A. Edwards, R. Vincentelli, C. Cambillau, R. Kim, S.-H. Kim, Z. Rao, Y. Shi, T. C. Terwilliger, C.-Y. Kim, L.-W. Hung, G. S. Waldo, Y. Peleg, S. Albeck, T. Unger, O. Dym, J. Prilusky, J. L. Sussman, R. C. Stevens, S. A. Lesley, I. A. Wilson, A. Joachimiak, F. Collart, I. Dementieva, M. I. Donnelly, W. H. Eschenfeldt, Y. Kim, L. Stols, R. Wu, M. Zhou, S. K. Burley, J. S. Emtage, J. M. Sauder, D. Thompson, K. Bain, J. Luz, T. Gheyi, F. Zhang, S. Atwell, S. C. Almo, J. B. Bonanno, A. Fiser, S. Swaminathan, F. W. Studier, M. R. Chance, A. Sali, T. B. Acton, R. Xiao, L. Zhao, L. C. Ma, J. F. Hunt, L. Tong, K. Cunningham, M. Inouye, S. Anderson, H. Janjua, R. Shastry, C. K. Ho, D. Wang, H. Wang, M. Jiang, G. T. Montelione, D. I. Stuart, R. J. Owens, S. Daenke, A. Schütz, U. Heinemann, S. Yokoyama, K. Büssow, and K. C. Gunsalus. Protein production and purification. *Nat Methods*, 5(2):135–46, Feb 2008.

[45] M. T. Cooling, V. Rouilly, G. Misirli, J. Lawson, T. Yu, J. Hallinan, and A. Wipat. Standard virtual biological parts: A repository of modular modeling components for synthetic biology. *Bioinformatics*, Feb 2010.

[46] R. S. Cox, M. G. Surette, and M. B. Elowitz. Programming gene expression with combinatorial promoters. *Mol Syst Biol*, 3:145, Jan 2007.

[47] M. J. Czar, J. C. Anderson, J. S. Bader, and J. Peccoud. Gene synthesis demystified. *Trends in Biotechnology*, 27(2):63–72, Feb 2009.

[48] M. J. Czar, Y. Cai, and J. Peccoud. Writing dna with genocad. *Nucleic Acids Res*, 37(Web Server issue):W40–7, Jul 2009.

[49] T. Danino, O. Mondragón-Palomino, L. Tsimring, and J. Hasty. A synchronized quorum of genetic clocks. *Nature*, 463(7279):326–30, Jan 2010.

[50] M. H. de Smit and J. van Duin. Secondary structure of the ribosome binding site determines translational efficiency: a quantitative analysis. *Proc Natl Acad Sci USA*, 87(19):7668–72, Oct 1990.

[51] T. L. Deans, C. R. Cantor, and J. J. Collins. A tunable genetic switch based on rnai and repressor proteins for regulating gene expression in mammalian cells. *Cell*, 130(2):363–72, Jul 2007.

[52] M. Desmit and J. Vanduin. Control of translation by messenger-rna secondary structure in escherichia-coli - a quantitative-analysis of literature data. *J Mol Biol*, 244(2):144–150, Jan 1994.

[53] W. Doerfler. In search of more complex genetic codes–can linguistics be a guide? *Medical hypotheses*, 9(6):563, 1982.

[54] S. Dong and D. B. Searls. Gene structure prediction by linguistic methods. *Genomics*, 23(3):540–51, Oct 1994.

[55] D. A. Drubin, J. C. Way, and P. A. Silver. Designing biological systems. *Genes & development*, 21(3):242–54, Feb 2007.

[56] J. S. Dymond, L. Z. Scheifele, S. M. Richardson, P. Lee, S. Chandrasegaran, J. S. Bader, and J. D. Boeke. Teaching synthetic biology, bioinformatics and engineering to undergraduates: the interdisciplinary build-a-genome course. *Genetics*, 181(1):13–21, Jan 2009.

[57] S. R. Eddy. How do rna folding algorithms work? *Nat Biotechnol*, 22(11):1457–8, Nov 2004.

[58] U. S. Eggert, T. J. Mitchison, and C. M. Field. Animal cytokinesis: from parts list to mechanisms. *Annu Rev Biochem*, 75:543–66, Jan 2006.

[59] K. Eilbeck, S. Lewis, C. Mungall, M. Yandell, and L. Stein. The sequence ontology: a tool for the unification of genome annotations. *Genome biology*, Jan 2005.

[60] T. Ellis, X. Wang, and J. J. Collins. Diversity-based, model-guided construction of synthetic gene networks with predicted functions. *Nat Biotechnol*, Mar 2009.

[61] M. B. Elowitz and S. Leibler. A synthetic oscillatory network of transcriptional regulators. *Nature*, 403(6767):335–8, Jan 2000.

[62] D. Endy. Foundations for engineering biology. *Nature*, 438(7067):449–53, Nov 2005.

[63] B. Ewing and P. Green. Base-calling of automated sequencer traces using phred. ii. error probabilities. *Genome research*, 8(3):186–94, Mar 1998.

[64] B. Ewing, L. Hillier, M. C. Wendl, and P. Green. Base-calling of automated sequencer traces using phred. i. accuracy assessment. *Genome research*, 8(3):175–85, Mar 1998.

[65] D. S. Falconer and T. F. C. Mackay. Introduction to quantitative genetics. *Benjamin Cummings*, page 464, Jan 1996.

[66] D. Ferber. Synthetic biology: microbes made to order. *Science*, Jan 2004.

[67] J. Finn, A. C. H. Lee, I. MacLachlan, and P. Cullis. An enhanced autogene-based dual-promoter cytoplasmic expression system yields increased gene expression. *Gene Ther*, 11(3):276–83, Feb 2004.

[68] N. C. Fitzkee, P. J. Fleming, H. Gong, N. Panasik, T. O. Street, and G. D. Rose. Are proteins made from a limited parts list? *Trends Biochem Sci*, 30(2):73–80, Feb 2005.

[69] C. E. French. Synthetic biology and biomass conversion: a match made in heaven? *J R Soc Interface*, 6 Suppl 4:S547–58, Aug 2009.

[70] A. E. Friedland, T. K. Lu, X. Wang, D. Shi, G. Church, and J. J. Collins. Synthetic gene networks that count. *Science*, 324(5931):1199–202, May 2009.

[71] E. Fung, W. W. Wong, J. K. Suen, T. Bulter, S. gu Lee, and J. C. Liao. A synthetic gene-metabolic oscillator. *Nature*, 435(7038):118–22, May 2005.

[72] T. S. Gardner, C. R. Cantor, and J. J. Collins. Construction of a genetic toggle switch in escherichia coli. *Nature*, 403(6767):339–42, Jan 2000.

[73] J. Gertz, E. D. Siggia, and B. A. Cohen. Analysis of combinatorial cis-regulation in synthetic and genomic promoters. *Nature*, 457(7226):215–8, Jan 2009.

[74] 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*, 425(6959):737–41, Oct 2003.

[75] D. G. Gibson, G. A. Benders, C. Andrews-Pfannkoch, E. A. Denisova, H. Baden-Tillson, J. Zaveri, T. B. Stockwell, A. Brownley, D. W. Thomas, M. A. Algire, C. Merryman, L. Young, V. N. Noskov, J. I. Glass, J. C. Venter, C. A. Hutchison, and H. O. Smith. Complete chemical synthesis, assembly, and cloning of a mycoplasma genitalium genome. *Science*, 319(5867):1215–20, Feb 2008.

[76] M. Gimona. Protein linguistics - a grammar for modular protein assembly? *Nat Rev Mol Cell Biol*, 7(1):68–73, Jan 2006.

[77] M. Gimona. Protein linguistics and the modular code of the cytoskeleton. *The Codes of Life the Rules of Macroevolution*, pages 189–206, Jan 2008.

[78] D. V. Goeddel, H. L. Heyneker, T. Hozumi, R. Arentzen, K. Itakura, D. G. Yansura, M. J. Ross, G. Miozzari, R. Crea, and P. H. Seeburg. Direct expression in escherichia coli of a dna sequence coding for human growth hormone. *Nature*, 281(5732):544–8, Oct 1979.

[79] J. Goler. Biojade: A design and simulation tool for synthetic biological systems. *dlrg.mit.edu*, Jan 2004.

[80] J. Goler, B. Bramlett, and J. Peccoud. Genetic design: rising above the sequence. *Trends in Biotechnology*, Aug 2008.

[81] V. Gonzalez-Nicolini and M. Fussenegger. A novel binary adenovirus-based dual-regulated expression system for independent transcription control of two different transgenes. *J Gene Med*, 7(12):1573–85, Dec 2005.

[82] C. Goodman. Engineering ingenuity at igem. *Nat Chem Biol*, 4(1):13, Jan 2008.

[83] M. Gossen and H. Bujard. Tight control of gene expression in mammalian cells by tetracycline-responsive promoters. *Proc Natl Acad Sci USA*, 89(12):5547–51, Jun 1992.

[84] M. Griffith, T. Courtney, J. Peccoud, and W. H. Sanders. Dynamic partitioning for hybrid simulation of the bistable hiv-1 transactivation network. *Bioinformatics*, 22(22):2782–9, Nov 2006.

[85] B. F. Group, D. Baker, G. M. Church, J. J. Collins, D. Endy, J. Jacobson, J. D. Keasling, P. Modrich, C. D. Smolke, and R. Weiss. Engineering life: building a fab for biology. *Sci Am*, 294(6):44–51, Jun 2006.

[86] C. C. Guet, M. B. Elowitz, W. Hsing, and S. Leibler. Combinatorial synthesis of genetic networks. *Science*, 296(5572):1466–70, May 2002.

[87] N. J. Guido, X. Wang, D. Adalsteinsson, D. McMillen, J. Hasty, C. R. Cantor, T. C. Elston, and J. J. Collins. A bottom-up approach to gene regulation. *Nature*, 439(7078):856–60, Feb 2006.

[88] U. Güldener, M. Münsterkötter, G. Kastenmüller, N. Strack, J. van Helden, C. Lemer, J. Richelles, S. J. Wodak, J. García-Martínez, J. E. Pérez-Ortín, H. Michael, A. Kaps, E. Talla, B. Dujon, B. André, J. L. Souciet, J. D. Montigny, E. Bon, C. Gaillardin, and H. W. Mewes. Cygd: the comprehensive yeast genome database. *Nucleic Acids Res*, 33(Database issue):D364–8, Jan 2005.

[89] K. A. Haynes, M. L. Broderick, A. D. Brown, T. L. Butner, J. O. Dickson, W. L. Harden, L. H. Heard, E. L. Jessen, K. J. Malloy, B. J. Ogden, S. Rosemond, S. Simpson, E. Zwack, A. M. Campbell, T. T. Eckdahl, L. J. Heyer, and J. L. Poet. Engineering bacteria to solve the burnt pancake problem. *Journal of biological engineering*, 2:8, Jan 2008.

[90] M. Heinemann and S. Panke. Synthetic biology–putting engineering into biology. *Bioinformatics*, 22(22):2790–9, Nov 2006.

[91] J. Henkel and S. M. Maurer. The economics of synthetic biology. *Mol Syst Biol*, 3:117, Jan 2007.

[92] A. D. Hill, J. R. Tomshine, E. M. B. Weeding, V. Sotiropoulos, and Y. N. Kaznessis. Synbioss: the synthetic biology modeling suite. *Bioinformatics*, 24(21):2551–3, Nov 2008.

[93] 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*, 22(24):3067–74, Dec 2006.

[94] D. M. Hoover and J. Lubkowski. Dnaworks: an automated method for designing oligonucleotides for pcr-based gene synthesis. *Nucleic Acids Res*, 30(10):e43, May 2002.

[95] D. Horn. Syntactic structures in languages and biology. *Cognitive processing*, 9(3):153–8, Aug 2008.

[96] X. Huang and A. Madan. Cap3: A dna sequence assembly program. *Genome Res*, 9(9):868–77, Sep 1999.

[97] 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. L. 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, E. Shapiro, T. S. Shimizu, H. D. Spence, J. Stelling, K. Takahashi, M. Tomita, J. Wagner, J. Wang, and S. Forum. The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–31, Mar 2003.

[98] A. M. Huerta, H. Salgado, D. Thieffry, and J. Collado-Vides. Regulondb: a database on transcriptional regulation in escherichia coli. *Nucleic Acids Res*, 26(1):55–9, Jan 1998.

[99] W.-K. Huh, J. V. Falvo, L. C. Gerke, A. S. Carroll, R. W. Howson, J. S. Weissman, and E. K. O'Shea. Global analysis of protein localization in budding yeast. *Nature*, 425(6959):686–91, Oct 2003.

[100] I. F. Institute, A. I. of Bolt, Nut, R. M. Bolt, nut, and rivet standards. Bolt, nut and rivet standards. *Industrial Fastners Institute*, page 255, Jan 1952.

[101] F. J. Isaacs, D. J. Dwyer, and J. J. Collins. Rna synthetic biology. *Nat Biotechnol*, 24(5):545–54, May 2006.

[102] F. J. Isaacs, D. J. Dwyer, C. Ding, D. D. Pervouchine, C. R. Cantor, and J. J. Collins. Engineered riboregulators enable post-transcriptional control of gene expression. *Nat Biotechnol*, 22(7):841–7, Jul 2004.

[103] M. Jabasini, L. Zhang, F. Dang, F. Xu, M. R. Almofli, A. A. Ewis, J. Lee, Y. Nakahori, and Y. Baba. Analysis of dna polymorphisms on the human y-chromosome by microchip electrophoresis. *Electrophoresis*, 23(10):1537–42, May 2002.

[104] S. Jayaraj, R. Reid, and D. V. Santi. Gems: an advanced software package for designing synthetic genes. *Nucleic Acids Res*, 33(9):3011–6, Jan 2005.

[105] A. M. Jenkinson, M. Albrecht, E. Birney, H. Blankenburg, T. Down, R. D. Finn, H. Hermjakob, T. J. P. Hubbard, R. C. Jimenez, P. Jones, A. Kähäri, E. Kulesha, J. R. Macías, G. A. Reeves, and A. Prlić. Integrating biological data–the distributed annotation system. *BMC Bioinformatics*, 9 Suppl 8:S3, Jan 2008.

[106] Y. Kato, H. Seki, and T. Kasammi. On the generative power of grammars for rna secondary structure. *IEICE Transactions on Information and Systems*, E88D(1):53–64, Jan 2005.

[107] L. E. Kay. Who wrote the book of life?: a history of the genetic code. *Stanford University Press*, page 441, Jan 2000.

[108] J. D. Keasling. Synthetic biology for synthetic chemistry. *ACS Chem Biol*, 3(1):64–76, Jan 2008.

[109] E. F. Keller. The century of the gene. *Harvard University Press*, page 192, Jan 2002.

[110] E. F. Keller, D. Harel, and T. Zwaka. Beyond the gene. *PLoS ONE*, 2(11):e1231, Nov 2007.

[111] J. R. Kelly, A. Rubin, J. Davis, C. Ajo-Franklin, J. Cumbers, M. J. Czar, K. D. Mora, A. Glieberman, D. Monie, and D. Endy. Measuring the activity of biobrick promoters using an in vivo reference standard. *Journal of biological engineering*, 3(1):4, Mar 2009.

[112] D. Kiga and M. Yamamura. Synthetic biology. *New Generation Computing*, Jan 2008.

[113] J. Kirby and J. D. Keasling. Metabolic engineering of microorganisms for isoprenoid production. *Nat Prod Rep*, 25(4):656–61, Aug 2008.

[114] B. Knudsen and J. Hein. Rna secondary structure prediction using stochastic context-free grammars and evolutionary history. *Bioinformatics*, 15(6):446–454, Jan 1999.

[115] B. Knudsen and J. Hein. Pfold: Rna secondary structure prediction using stochastic context-free grammars. *Nucleic Acids Res*, 31(13):3423–8, Jul 2003.

[116] D. Knuth. Semantics of context-free languages. *Theory of Computing Systems*, Jan 1968.

[117] D. Knuth. The genesis of attribute grammars. *Lecture Notes in Computer Science*, 461:1–12, Jan 1990.

[118] H. Kobayashi, M. Kaern, M. Araki, K. Chung, T. S. Gardner, C. R. Cantor, and J. J. Collins. Programmable cells: interfacing natural and engineered gene networks. *Proc Natl Acad Sci USA*, 101(22):8414–9, Jun 2004.

[119] D. S. Kong, P. A. Carr, L. Chen, S. Zhang, and J. M. Jacobson. Parallel gene synthesis in a microfluidic device. *Nucleic Acids Res*, 35(8):e61, Jan 2007.

[120] B. P. Kramer, A. U. Viretta, M. Daoud-El-Baba, D. Aubel, W. Weber, and M. Fussenegger. An engineered epigenetic transgene switch in mammalian cells. *Nat Biotechnol*, 22(7):867–70, Jul 2004.

[121] G. Kudla, A. W. Murray, D. Tollervey, and J. B. Plotkin. Coding-sequence determinants of gene expression in escherichia coli. *Science*, 324(5924):255–8, Apr 2009.

[122] M. Landthaler and D. A. Shub. Unexpected abundance of self-splicing introns in the genome of bacteriophage twort: introns in multiple genes, a single gene with three introns, and exon skipping by group i ribozymes. *Proc Natl Acad Sci USA*, 96(12):7005–10, Jun 1999.

[123] C. Lartigue, J. I. Glass, N. Alperovich, R. Pieper, P. P. Parmar, C. A. Hutchison, H. O. Smith, and J. C. Venter. Genome transplantation in bacteria: changing one species to another. *Science*, 317(5838):632–8, Aug 2007.

[124] K. Lee. Principles of cad/cam/cae systems. *Prentice Hall*, page 582, Jan 1999.

[125] S. K. Lee, H. Chou, T. S. Ham, T. S. Lee, and J. D. Keasling. Metabolic engineering of microorganisms for biofuels production: from bugs to synthetic biology to fuels. *Curr Opin Biotech*, 19(6):556–63, Dec 2008.

[126] S. Leung, C. Mellish, and D. Robertson. Basic gene grammars and dna-chartparser for language processing of escherichia coli promoter dna sequences. *Bioinformatics*, 17(3):226–36, Mar 2001.

[127] A. Levskaya, A. A. Chevalier, J. J. Tabor, Z. B. Simpson, L. A. Lavery, M. Levy, E. A. Davidson, A. Scouras, A. D. Ellington, E. M. Marcotte, and C. A. Voigt. Synthetic biology: engineering escherichia coli to see light. *Nature*, 438(7067):441–2, Nov 2005.

[128] R. C. Lewontin. The triple helix: gene, organism, and environment. *Harvard University Press*, page 136, Jan 2001.

[129] P. Linz. An introduction to formal languages and automata. *Jones & Bartlett Pub*, Jan 2006.

[130] C. M. Lloyd, M. Halstead, and P. Nielsen. Cellml: its future, present and past. *Progress in Biophysics and Molecular Biology*, Jan 2004.

[131] C. M. Lloyd, J. R. Lawson, P. J. Hunter, and P. F. Nielsen. The cellml model repository. *Bioinformatics*, 24(18):2122–2123, Jan 2008.

[132] C. Loose, K. Jensen, I. Rigoutsos, and G. Stephanopoulos. A linguistic model for the rational design of antimicrobial peptides. *Nature*, 443(7113):867–9, Oct 2006.

[133] C. Lou, X. Liu, M. Ni, Y. Huang, Q. Huang, L. Huang, L. Jiang, D. Lu, M. Wang, C. Liu, D. Chen, C. Chen, X. Chen, L. Yang, H. Ma, J. Chen, and Q. Ouyang. Synthesizing a novel genetic sequential logic circuit: a push-on push-off switch. *Mol Syst Biol*, 6:350, Jan 2010.

[134] T. K. Lu and J. J. Collins. Engineered bacteriophage targeting gene networks as adjuvants for antibiotic therapy. *Proc Natl Acad Sci USA*, 106(12):4629–34, Feb 2009.

[135] T. K. Lu, A. S. Khalil, and J. J. Collins. Next-generation synthetic gene networks. *Nat Biotechnol*, 27(12):1139–50, Dec 2009.

[136] M. Lynch and B. Walsh. Genetics and analysis of quantitative traits. *Sinauer Associates*, page 980, Jan 1998.

[137] M. A. Marchisio and J. Stelling. Computational design of synthetic gene circuits with composable parts. *Bioinformatics*, 24(17):1903–1910, Jan 2008.

[138] M. A. Marchisio and J. Stelling. Computational design tools for synthetic biology. *Curr Opin Biotech*, Sep 2009.

[139] H. Matsui, K. Sato, and Y. Sakakibara. Pair stochastic tree adjoining grammars for aligning and predicting pseudoknot rna structures. *Bioinformatics*, 21(11):2611–7, Jun 2005.

[140] R. McDaniel and R. Weiss. Advances in synthetic biology: on the path from prototypes to applications. *Curr Opin Biotechnol*, 16(4):476–83, Aug 2005.

[141] C. Mead and L. Conway. Introduction to vlsi systems. *Addison-Wesley Pub*, page 396, Jan 1980.

[142] I. M. Meyer and I. Miklós. Simulfold: simultaneously inferring rna structures including pseudoknots, alignments, and trees using a bayesian mcmc framework. *PLoS Comput Biol*, 3(8):e149, Aug 2007.

[143] M. L. Meyer-Ficca, R. G. Meyer, H. Kaiser, A. R. Brack, R. Kandolf, and J.-H. Küpper. Comparative analysis of inducible expression systems in transient transfection studies. *Anal Biochem*, 334(1):9–19, Nov 2004.

[144] E. S. Miller, E. Kutter, G. Mosig, F. Arisaka, T. Kunisawa, and W. Rüger. Bacteriophage t4 genome. *Microbiol Mol Biol Rev*, 67(1):86–156, Mar 2003.

[145] R. Moore. Removing left recursion from context-free grammars. *ACM International Conference Proceeding Series*, Jan 2000.

[146] M. Mueller, L. Martens, and R. Apweiler. Annotating the human proteome: beyond establishing a parts list. *Biochim Biophys Acta*, 1774(2):175–91, Feb 2007.

[147] S. Mueller, J. R. Coleman, and E. Wimmer. Putting synthesis into biology: a viral view of genetic engineering through de novo gene and genome synthesis. *Chem Biol*, 16(3):337–47, Mar 2009.

[148] S. H. Muggleton, C. H. Bryant, A. Srinivasan, A. Whittaker, S. Topp, and C. Rawlings. Are grammatical representations useful for learning from biological sequence data?–a case study. *J Comput Biol*, 8(5):493–521, Jan 2001.

[149] R. Münch, K. Hiller, H. Barg, D. Heldt, S. Linz, E. Wingender, and D. Jahn. Prodoric: prokaryotic database of gene regulation. *Nucleic Acids Res*, 31(1):266–9, Jan 2003.

[150] K. F. Murphy, G. Balázsi, and J. J. Collins. Combinatorial promoter design for engineering noisy gene expression. *Proc Natl Acad Sci USA*, 104(31):12726–31, Jul 2007.

[151] J. Paakki. Attribute grammar paradigms—a high-level methodology in language implementation. *Computing Surveys CSUR*, 27(2), Jun 1995.

[152] M. Padidam. Chemically regulated gene expression in plants. *Curr Opin Plant Biol*, 6(2):169–77, Apr 2003.

[153] T. Parr. The definitive antlr reference: building domain-specific languages. *Pragmatic Bookshelf*, page 361, Jan 2007.

[154] A. Pavesi. Origin and evolution of overlapping genes in the family microviridae. *J Gen Virol*, 87(Pt 4):1013–7, Apr 2006.

[155] J. Peccoud, M. F. Blauvelt, Y. Cai, K. L. Cooper, O. Crasta, E. C. DeLalla, C. Evans, O. Folkerts, B. M. Lyons, S. P. Mane, R. Shelton, M. A. Sweede, and S. A. Waldon. Targeted development of registries of biological parts. *PLoS ONE*, 3(7):e2671, Jan 2008.

[156] J. Peccoud and Y. Cai. Software for design and verification of synthetic genetic constructs. *US Patent App. 12/058,712*, Jan 2008.

[157] J. Peccoud and L. Coulombel. [a competition of synthetic biology or how to create the "water of e. coli" and nano-barbies]. *Med Sci (Paris)*, 23(5):551–2, May 2007.

[158] M. Pedersen and A. Phillips. Towards programming languages for genetic engineering of living cells. *Journal of the Royal Society, Interface / the Royal Society*, Apr 2009.

[159] R. C. Périer, T. Junier, and P. Bucher. The eukaryotic promoter database epd. *Nucleic Acids Res*, 26(1):353–7, Jan 1998.

[160] S. Picataggio. Potential impact of synthetic biology on the development of microbial systems for the production of renewable fuels and chemicals. *Curr Opin Biotech*, May 2009.

[161] P. E. M. Purnick and R. Weiss. The second wave of synthetic biology: from modules to systems. *Nat Rev Mol Cell Biol*, 10(6):410–22, Jun 2009.

[162] D. Quest, W. Tapprich, and H. Ali. A grammar based methodology for structural motif finding in ncrna database search. *Computational systems bioinformatics / Life Sciences Society Computational Systems Bioinformatics Conference*, 6:215–25, Jan 2007.

[163] M. Rabani, M. Kertesz, and E. Segal. Computational prediction of rna structural motifs involved in posttranscriptional regulatory processes. *Proc Natl Acad Sci USA*, 105(39):14885–90, Sep 2008.

[164] A. Rai and J. Boyle. Synthetic biology: caught between property rights, the public domain, and the commons. *PLoS Biol*, 5(3):e58, Mar 2007.

[165] A. Raj and A. V. Oudenaarden. Nature, nurture, or chance: stochastic gene expression and its consequences. *Cell*, 135(2):216–26, Oct 2008.

[166] S. A. Ramsey, J. J. Smith, D. Orrell, M. Marelli, T. W. Petersen, P. de Atauri, H. Bolouri, and J. D. Aitchison. Dual feedback loops in the gal regulon suppress cellular heterogeneity in yeast. *Nat Genet*, 38(9):1082–7, Sep 2006.

[167] A. Rashtchian, G. W. Buchman, D. M. Schuster, and M. S. Berninger. Uracil dna glycosylase-mediated cloning of polymerase chain reaction-amplified dna: application to genomic and cdna cloning. *Anal Biochem*, 206(1):91–7, Oct 1992.

[168] S. M. Richardson, S. J. Wheelan, R. M. Yarrington, and J. D. Boeke. Genedesign: rapid, automated design of multikilobase synthetic genes. *Genome Res*, 16(4):550–6, Apr 2006.

[169] I. Rigoutsos and A. Floratos. Combinatorial pattern discovery in biological sequences: The teiresias algorithm. *Bioinformatics*, 14(1):55–67, Jan 1998.

[170] I. Rigoutsos, A. Floratos, L. Parida, Y. Gao, and D. Platt. The emergence of pattern discovery techniques in computational biology. *Metab Eng*, 2(3):159–77, Jul 2000.

[171] E. Rivas and S. R. Eddy. A dynamic programming algorithm for rna structure prediction including pseudoknots. *J Mol Biol*, 285(5):2053–68, Feb 1999.

[172] E. Rivas and S. R. Eddy. The language of rna: a formal grammar that includes pseudoknots. *Bioinformatics*, 16(4):334–40, Apr 2000.

[173] D.-K. Ro, E. M. Paradise, M. Ouellet, K. J. Fisher, K. L. Newman, J. M. Ndungu, K. A. Ho, R. A. Eachus, T. S. Ham, J. Kirby, M. C. Y. Chang, S. T. Withers, Y. Shiba, R. Sarpong, and J. D. Keasling. Production of the antimalarial drug precursor artemisinic acid in engineered yeast. *Nature*, 440(7086):940–3, Apr 2006.

[174] S. H. Rodger and T. W. Finley. Jflap-an interactive formal languages and automata package. *Jones & Bartlett Pub*, Jan 2006.

[175] G. Rodrigo, J. Carrera, and A. Jaramillo. Asmparts: assembly of biological model parts. *Systems and synthetic biology*, 1(4):167–70, Dec 2007.

[176] D. A. Rosenblueth, D. Thieffry, A. M. Huerta, H. Salgado, and J. Collado-Vides. Syntactic recognition of regulatory regions in escherichia coli. *Comput Appl Biosci*, 12(5):415–22, Oct 1996.

[177] Y. Sakakibara. Recent advances of grammatical inference. *Theoretical Computer Science*, 185(1):15–45, 1997.

[178] Y. Sakakibara. Grammatical inference in bioinformatics. *Ieee Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1051–62, Jul 2005.

[179] Y. Sakakibara. Learning context-free grammars using tabular representations. *Pattern Recognition*, 38(9):1372–1383, 2005.

[180] Y. Sakakibara, M. Brown, R. Hughey, I. S. Mian, K. Sjölander, R. C. Underwood, and D. Haussler. Stochastic context-free grammars for trna modeling. *Nucleic Acids Res*, 22(23):5112–20, Nov 1994.

[181] Y. Sakakibara and H. Muramatsu. Learning context-free grammars from partially structured examples. *Lecture Notes in Computer Science*, 1891:229–240, 2000.

[182] H. M. Salis, E. A. Mirsky, and C. A. Voigt. Automated design of synthetic ribosome binding sites to control protein expression. *Nat Biotechnol*, 27(10):946–50, Oct 2009.

[183] D. B. Searls. The linguistics of dna. *American Scientist*, Jan 1992.

[184] D. B. Searls. The computational linguistics of biological sequences. *Artificial Intelligence and Molecular Biology*, Jan 1993.

[185] D. B. Searls. Sequence alignment through pictures. *Trends Genet*, 12(1):35–7, Jan 1996.

[186] D. B. Searls. Linguistic approaches to biological sequences. *Comput Appl Biosci*, 13(4):333–44, Aug 1997.

[187] D. B. Searls. Reading the book of life. *Bioinformatics*, 17(7):579–80, Jul 2001.

[188] D. B. Searls. The language of genes. *Nature*, 420(6912):211–7, Nov 2002.

[189] D. B. Searls. Linguistics: trees of life and of language. *Nature*, 426(6965):391–2, Nov 2003.

[190] D. B. Searls and K. P. Murphy. Automata-theoretic models of mutation and alignment. *Proceedings / International Conference on Intelligent Systems for Molecular Biology ; ISMB International Conference on Intelligent Systems for Molecular Biology*, 3:341–9, Jan 1995.

[191] D. B. Searls and M. Noordewier. Pattern-matching search of dna sequences using logic grammars. *Artificial Intelligence Applications, 1991. Proceedings., Seventh IEEE Conference on*, i:3 – 9, Jan 1991.

[192] E. Segal, Y. Fondufe-Mittendorf, L. Chen, A. Thåström, Y. Field, I. K. Moore, J.-P. Z. Wang, and J. Widom. A genomic code for nucleosome positioning. *Nature*, 442(7104):772–8, Aug 2006.

[193] L. Serrano. Synthetic biology: promises and challenges. *Mol Syst Biol*, 3:158, Jan 2007.

[194] P. Shannon, A. Markiel, O. Ozier, N. S. Baliga, J. T. Wang, D. Ramage, N. Amin, B. Schwikowski, and T. Ideker. Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome research*, 13(11):2498–504, Nov 2003.

[195] R. P. Shetty, D. Endy, and T. F. Knight. Engineering biobrick vectors from biobrick parts. *Journal of biological engineering*, 2:5, Jan 2008.

[196] R. K. Shultzaberger, R. E. Bucheimer, K. E. Rudd, and T. D. Schneider. Anatomy of escherichia coli ribosome binding sites. *J Mol Biol*, 313(1):215–28, Oct 2001.

[197] M. Singer and P. Berg. George beadle: from genes to proteins. *Nat Rev Genet*, 5(12):949–54, Dec 2004.

[198] K. Slonneger and B. L. Kurtz. Formal syntax and semantics of programming languages: A laboratory based approach. *Addison Wesley Longman*, page 637, Jan 1995.

[199] C. Smith, P. J. Day, and M. R. Walker. Generation of cohesive ends on pcr products by udg-mediated excision of du, and application for cloning into restriction digest-linearized vectors. *PCR Methods Appl*, 2(4):328–32, May 1993.

[200] H. O. Smith, C. A. Hutchison, C. Pfannkoch, and J. C. Venter. Generating a synthetic genome by whole genome assembly: phix174 bacteriophage from synthetic oligonucleotides. *Proc Natl Acad Sci USA*, 100(26):15440–5, Dec 2003.

[201] C. D. Smolke. Building outside of the box: igem and the biobricks foundation. *Nat Biotechnol*, 27(12):1099–102, Dec 2009.

[202] Z. Solan, D. Horn, E. Ruppin, and S. Edelman. Unsupervised learning of natural languages. *Proc Natl Acad Sci USA*, 102(33):11629–34, Aug 2005.

[203] C. N. Stewart. Plant functional genomics: beyond the parts list. *Trends in Plant Science*, 10(12):561–2, Dec 2005.

[204] J. E. Stoy. Denotational semantics: The scott-strachey approach to programming language theory. *The MIT Press*, page 414, Jan 1981.

[205] J. Stricker, S. Cookson, M. R. Bennett, W. H. Mather, L. S. Tsimring, and J. Hasty. A fast, robust and tunable synthetic gene oscillator. *Nature*, 456(7221):516–U39, Jan 2008.

[206] A. H. Sturtevant. A history of genetics. *Cold Spring Harbor Laboratory Press*, page 174, Jan 2001.

[207] M. Suárez and A. Jaramillo. Challenges in the computational design of proteins. *Journal of the Royal Society, Interface / the Royal Society*, 6 Suppl 4:S477–91, Aug 2009.

[208] T. A. Sudkamp. Languages and machines: an introduction to the theory of computer science. *Addison Wesley*, page 569, Jan 1997.

[209] Y. Tanouchi, A. Pai, and L. You. Decoding biological principles using gene circuits. *Mol Biosyst*, 5(7):695–703, Jul 2009.

[210] E. Tatum. A case history in biological research. *Science*, 129(3365):1711–5, Jun 1959.

[211] J. Tian, H. Gong, N. Sheng, X. Zhou, E. Gulari, X. Gao, and G. Church. Accurate multiplex gene synthesis from programmable dna microchips. *Nature*, 432(7020):1050–4, Dec 2004.

[212] M. Tigges, T. T. Marquez-Lago, J. Stelling, and M. Fussenegger. A tunable synthetic mammalian oscillator. *Nature*, 457(7227):309–12, Jan 2009.

[213] T. M. Tumpey, C. F. Basler, P. V. Aguilar, H. Zeng, A. Solórzano, D. E. Swayne, N. J. Cox, J. M. Katz, J. K. Taubenberger, P. Palese, and A. García-Sastre. Characterization of the reconstructed 1918 spanish influenza pandemic virus. *Science*, 310(5745):77–80, Oct 2005.

[214] R. L. Vellanoweth and J. C. Rabinowitz. The influence of ribosome-binding-site elements on translational efficiency in bacillus subtilis and escherichia coli in vivo. *Mol Microbiol*, 6(9):1105–14, May 1992.

[215] A. Villalobos, J. E. Ness, C. Gustafsson, J. Minshull, and S. Govindarajan. Gene designer: a synthetic biology tool for constructing artificial dna segments. *BMC Bioinformatics*, 7:285, Jan 2006.

[216] C. A. Voigt. Genetic parts to program bacteria. *Curr Opin Biotechnol*, 17(5):548–57, Oct 2006.

[217] C. A. Voigt and J. D. Keasling. Programming cellular function. *Nat Chem Biol*, 1(6):304–7, Nov 2005.

[218] G. von Dassow, E. Meir, E. M. Munro, and G. M. Odell. The segment polarity network is a robust developmental module. *Nature*, 406(6792):188–92, Jul 2000.

[219] R. Wang, X. Zhou, and X. Wang. Chemically regulated expression systems and their applications in transgenic plants. *Transgenic Res*, 12(5):529–40, Oct 2003.

[220] S. Wang, Y. Zhang, and Q. Ouyang. Stochastic model of coliphage lambda regulatory network. *Phys Rev E Stat Nonlin Soft Matter Phys*, 73(4 Pt 1):041922, Apr 2006.

[221] J. D. Watson and F. H. Crick. The structure of dna. *Cold Spring Harb Symp Quant Biol*, 18:123–31, Jan 1953.

[222] R. Weiss and S. Panke. Synthetic biology-paths to moving forward. *Curr Opin Biotechnol*, 20(4):447–8, Aug 2009.

[223] S. Weng, Q. Dong, R. Balakrishnan, K. Christie, M. Costanzo, K. Dolinski, S. S. Dwight, S. Engel, D. G. Fisk, E. Hong, L. Issel-Tarver, A. Sethuraman, C. Theesfeld, R. Andrada, G. Binkley, C. Lane, M. Schroeder, D. Botstein, and J. M. Cherry. Saccharomyces genome database (sgd) provides biochemical and structural information for budding yeast proteins. *Nucleic Acids Res*, 31(1):216–8, Jan 2003.

[224] G. Wu, J. B. Wolf, A. F. Ibrahim, S. Vadasz, M. Gunasinghe, and S. J. Freeland. Simplified gene synthesis: a one-step approach to pcr-based gene construction. *J Biotechnol*, 124(3):496–503, Jul 2006.

[225] B. J. Yeh and W. A. Lim. Synthetic biology: lessons from the history of synthetic organic chemistry. *Nat Chem Biol*, 3(9):521–5, Sep 2007.

[226] J. Zhu and M. Q. Zhang. Scpd: a promoter database of the yeast saccharomyces cerevisiae. *Bioinformatics*, 15(7-8):607–11, Jan 1999.