TASBE: A Tool-Chain to Accelerate Synthetic Biological Engineering

Jacob Beal[1] Ron Weiss[2] Douglas Densmore[3] Aaron Adler[1]
Jonathan Babb[2] Swapnil Bhatia[3] Noah Davidsohn[2] Traci Haddock[3]
Fusun Yaman[1] Richard Schantz[1] Joseph Loyall[1]
[1]BBN Technologies 10 Moulton Street Cambridge, MA, USA 02138
[2]MIT 77 Massachusetts Ave Cambridge, MA, USA 02139
[3]Boston University 8 Saint Mary's St. Boston, MA, USA 02215
{jakebeal,fyaman, aadler, rschantz, jloyall}@bbn.com,
{rweiss, ndavidso, jbabb}@mit.edu, {dougd,swapnilb,thaddock}@bu.edu

1. MOTIVATION

There is a pressing need for design automation tools for synthetic biology systems. Compared to electronic circuits, cellular information processing has more complex elementary components and a greater complexity of interactions among components. Moreover, chemical computation within a cell is strongly affected both by other computations simultaneously occurring in the cell and by the cell's native metabolic processes and its external environment. This complexity implies an engineering work-flow that is currently highly iterative, error-prone, and extremely slow—critical problems that must all be addressed in order to realize the potential of synthetic biology.

In recent years, an assortment of tools have emerged, each independently addressing various parts of the design automation challenge. For example, the RBS calculator[10] helps design ribosome binding sites, the GeneDesign suite[9] optimizes coding sequences, and TinkerCell[5] is a graphical tool for visualizing and designing regulatory networks, to name only a few. A few projects, such as Eugene[4], GenoCAD[6], GEC[8], and Proto[2], have even begun extending design up to higher level languages. The time is now right to begin connecting and organizing such efforts together into a tool-chain for integrated end-to-end design and construction of synthetic biology systems.

In the TASBE project, we are developing one such approach to factoring the problem of design and assembly into sub-problems which can be more readily solved. Practitioners using our tool-chain will be able to design organisms using high level behavior descriptions, which are automatically transformed into genetic regulatory network designs, then assembled into DNA samples ready for *in vivo* execution. The tool-chain is also free and open software, which will allow researchers to incorporate their own design tools, thereby disseminating their results to the community and enhancing the capabilities of the tool-chain.

2. PROTOTYPE TOOL-CHAIN

TASBE is a modular tool-chain with both forward and backward information flow, designed to support multiple compatible workflows and integration of additional design tools developed by the broader research community. Figure 1 shows the prototype TASBE architecture.

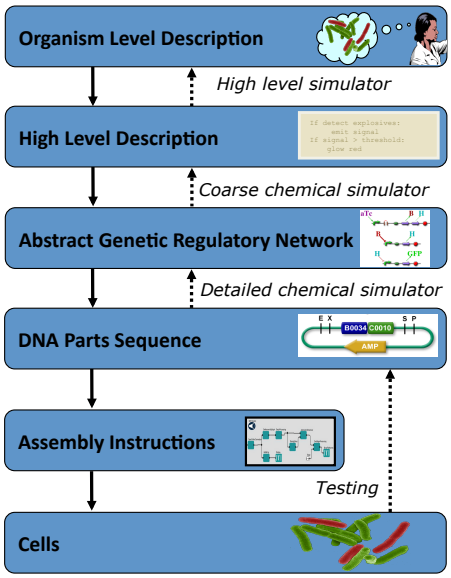

Figure 1: Prototype TASBE tool-chain architecture.

At the highest level, TASBE allows expression of the desired system function using a biologically-focused high-level programming language. TASBE then uses a compiler to transform this design systematically into an abstract genetic regulatory network design (AGRN), which specifies the types of DNA parts and interactions needed to implement the network. Optimization is performed on this network to allow a parsimonious realization under constraints of metabolic load and biological part availability. The abstract network is then instantiated by mapping the network components to existing biological parts with the help of a parts library that documents the chemical properties of such parts. The properties of these parts are established through a process of DNA device characterization, in which the input/output transfer functions of biological devices are measured in support of abstractions such as the generalized digital static discipline. Finally, the desired DNA sequence is assembled using standard laboratory automation robotics,

Work sponsored by DARPA I2O under contract HR0011-10-C-0168; the views and conclusions contained in this document are those of the authors and not DARPA or the U.S. Government.
IWBDA '11 San Diego, California, USA

which execute an automatically generated sequence of biological protocols.

At each stage, simulation and testing tools help with system debugging and provide a counter-flow of information (black dotted arrows in Figure 1) to the human designer. This approach maximizes the chance of catching design flaws early and minimizes the number of times that an actual biological system must be assembled and tested. Our prototype implementation contains the following components:

- *A biologically-focused high-level language:* A high-level programming language is a critical component of a tool-chain because it allows cell behavior to be described succinctly, allowing non-biologists to participate in the design process. We have chosen Proto [1], a spatial computing language, since it offers a unique continuous parallel dataflow semantics that is a good match for current biological computating models.

- *BioCompiler:* We have implemented a motif-based compiler[3] that transforms dataflow computations into AGRNs. The BioCompiler maps Proto primitives to genetic network motifs to produce an AGRN, then uses adapted versions of traditional compiler optimizations, such as dead code elimination and copy propagation, to reduce AGRN complexity.

- *Mapping between abstract and available DNA parts* To realize an AGRN, each AGRN element must be mapped to an available DNA part, and this mapping must be chosen such that the resulting network will function correctly within the bounds of chemical, physical, and metabolic constraints. Our MatchMaker system uses characterization data and a generalized version of the digital static discipline to guide a heuristic search for a correct implementation of an AGRN, then performs a greedy linearization of this network to yield a sequence of available DNA parts to be assembled.

- *DNA Assembly Planning* Laboratory automation equipment can execute DNA assembly protocols faster and more reliably than human technicians. Our Assembly Planner system, implemented in the Clotho framework[7], inputs a sequence of DNA parts and a protocol type and produces a set of assembly instructions for the laboratory robot to execute the protocol and produce a DNA sample ready for *in vivo* execution. By accessing laboratory inventory, this system can also optimize the number of assembly steps and the reusability of assembly mid-products.

- *Robotic Assembly* Finally, DNA assembly protocols are executed on a laboratory automation robot. At present, we are using a Tecan Evo 150, and two assembly protocols: BioBricks assembly for *E. coli* and a modified Gibson/Gateway protocol that uses magnetic beads and magnetic blocks to automate manual steps in DNA assembly.

Figure 2 shows a simple program, "If the cell detects the Dox molecule it fluoresces cyan, otherwise it fluoresces yellow," as it passes through the stages of the tool-chain. We have achieved an end-to-end integration of the software tools, and have used it to produce part sequence designs equivalent to known good designs executing *in vivo* in the Weiss laboratory. Once our current work on assembly protocol implementation is complete, these designs will be transformed into DNA samples, providing the first end-to-end compilation of high-level programs into DNA samples.

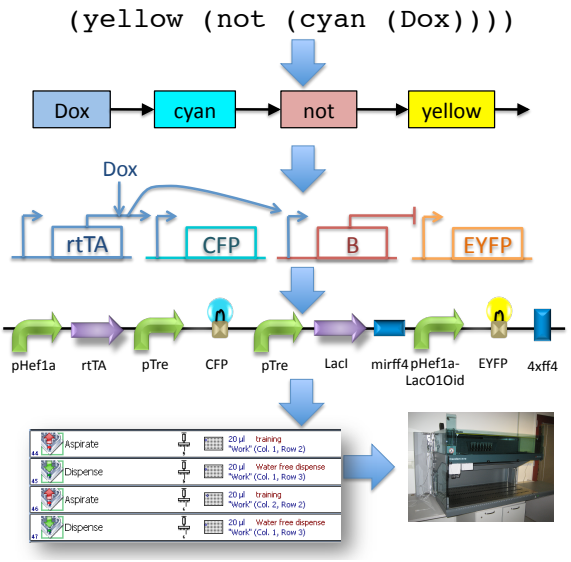

Figure 2: A simple program, to fluoresce cyan in the presence of Dox and yellow in its absence, passing through the stages of the tool-chain.

Note that although the prototype architecture is mostly linear, linearity is not an assumption of the architecture, but merely reflects the limited set of components in the initial prototype. Additionally, we have chosen Clotho[7] for an implementation framework for many of these components as its "app store" API and data management model facilitate integration of these tools and potentially many others. Finally, the components of our tool-chain are all designed for extensibility: Proto and the BioCompiler accept new primitive and motif definitions, MatchMaker can be augmented with additional design constraints, the Assembly Planner can be extended for new protocols, and the robotic automation systems are being controlled with a generalized API that should be portable to other robotic systems.

3. CONTRIBUTIONS

TASBE is an initial step towards achieving automation in synthetic biology. We have developed an initial set of prototype tools that can serve as a backbone for developing a larger, more comprehensive tool-chain. Our choice of decomposition attempts to minimize interaction between subtasks, thereby simplifying a potentially intractable problem of design at a cost of possible increased cost and inability to find solutions at the edge of design viability. Initial results point toward likely success of the TASBE approach. We intend to offer this tool-chain as an open platform for the research community, potentially multiplying the impact of each new design tool and significantly speeding the progress of synthetic biology research.

4. REFERENCES

[1] J. Beal and J. Bachrach. Infrastructure for engineered emergence in sensor/actuator networks. *IEEE Intelligent Systems*, pages 10–19, March/April 2006.

[2] J. Beal and J. Bachrach. Cells are plausible targets for high-level spatial languages. In *Spatial Computing Workshop*, 2008.

[3] J. Beal, T. Lu, and R. Weiss. Automatic compilation from high-level languages to genetic regulatory networks. In *Proceedings of IWBDA: International Workshop on Bio-Design Automation at DAC*, June 2010.

[4] Berkeley Software 2009 iGem Team. Eugene. http://2009.igem.org/Team:Berkeley_Software/Eugene, October 2009.

[5] D. Chandran, F. Bergmann, and H. Sauro. Tinkercell: modular cad tool for synthetic biology. *Journal of Biological Engineering*, 3(1):19, 2009.

[6] M. Czar, Y. Cai, and J. Peccoud. Writing DNA with GenoCAD. *Nucleic Acids Research*, 37(W40-7), 2009.

[7] D. Densmore, A. V. Devender, M. Johnson, and N. Sritanyaratana. A platform-based design environment for synthetic biological systems. In *TAPIA '09*, pages 24–29. ACM, 2009.

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

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

[10] H. Salis, E. Mirsky, and C. Voigt. Automated design of synthetic ribosome binding sites to control protein expression. *Nature Biotechnology*, 27:946–950, 2009.