Biochemistry as a Programming Language [*]
Full Presentation

Saurabh Srivastava [†]
Tim Hsiau, Sarah Chasins, Jon Kotker, Yen-Sheng Ho, Paul Ruan, Jeff Tsui, Stephi Hamilton, Jene Li, J. Christopher Anderson, Sanjit A. Seshia, Rastislav Bodik
University of California, Berkeley
saurabhs@cs.berkeley.edu

Abstract

Synthetic biology makes biology engineerable. One objective of this engineering is to modify the chemical reactions within the cell, i.e., the biochemistry, to produce non-native compounds of commercial interest. To do this at scale, ideas from language design, verification, and synthesis will be useful. In this talk, we present our lessons learnt, future avenues and open problems, in formalizing biochemistry.

1. Introduction

Synthetic biology is a relatively new subdomain of biology. In it experimentalists attempt to bring engineering principles of modularity, component reuse, and abstraction to genetically engineer organisms (typically bacteria *Echerichia coli*, or yeast *Saccharomyces cerevisiae*) for targeted objectives. One such objective is the biological production of useful chemicals. Target chemicals include biofuels (such as butanol for replacing gasoline), therapeutic drugs (such as artemisinin for malaria), cosmetics and lubricants (such as squalene, which is normally obtained from shark liver), and polymers (such as capralactam for Nylon). Engineering bacteria to produce these targets alleviates the environmental/economical/ethical costs because the biochemical equivalent of raw materials are sugar nutrients for feeding the bacteria as opposed to fossil fuel precursors. In addition to these *microbial chemical factories* (MCF), other applications of synthetic biology include genetic logic circuits, biosensors, tumor killing bacteria, and even organisms with completely synthetic DNA.[1] Some of these applications have already found industrial uses. Amyris—a company commercializing artemisinic acid (a precursor to an anti-malarial drug) producing yeast—is providing 30 million cures this year [9]. Genomatica engineered bacteria to produce 1,4-Butanediol [6, 14] (a precursor to various plastics and elastic clothing). Unfortunately, these remarkable feats took years to decades to achieve. The focus of the community has now shifted towards reuseable plug-and-play parts [1] with a hierarchy of abstraction ranging from DNA, to parts, to devices, to pathways, and finally to cells.

While language technology is broadly applicable across the spectrum of synthetic biology applications, here we focus on MCF, where one needs to reason about reactions, both chemical (in a test-tube) and biochemical (within a cell). The language of biochemistry manipulates chemical states, i.e., compounds, through transformations, i.e., reactions. Computational techniques are part of the analysis and design of these biological systems: algorithm design (sequence assembly), machine learning and AI planning (protein folding), graphics (protein visualization), and others have been applied. Formal methods and languages, as yet underrepresented, can provide complementary solutions.

In this talk, we will discuss three specific open problems. The first is a language design problem (Section 2). There is need for a succinct, yet expressive language describing chemical reactions. The current standards for encoding compounds are designed for storage and retrieval, but for compound transformations a better language is needed. The second is a verification problem (Section 3) in the sense of checking if the system can reach a designated state. Given a set of reactions (think of them as a set of guarded commands within an infinite loop), we need techniques that infer whether there is a path to a target chemical (think of it as a final fault state). This is a reachability problem to find a trace in the infinite domain of chemical structures. The third is a synthesis problem (Section 4). A reaction can be viewed as a summary of electron/proton movements that structurally transform a molecule. Given a set of allowable movements, can one synthesize a sequence that explains a reaction?

Next, we discuss these problems in more detail.

[*] This work was supported by the the DARPA Living Foundries initiative. Additional support has been provided by the NSF Synthetic Biology Engineering Research Center (SynBERC).

[†] Corresponding author

[1] Broader introductions to the field can be found in video tutorials [3, 5, 10].

2. Language for compounds and chemical transforms

The structure manipulated in biochemistry is a molecule: a collection of atoms connected through one of four bond types. One can easily view a molecule as a graph, with the atoms the nodes and bonds the edges. While this graph structure is good for in-memory manipulation, for readability, accuracy of representation, storage/retrieval and querying, a textual representation is more appropriate. There are various proposal directed towards optimizing one or the other. SMILES [11, 13, 12] optimizes human readability and querying, IUPAC [4] optimizes standard representation and readability, InChI [8] optimizes accurate representation and storage. Figure 1(a) shows an example molecule and its SMILES[2]. Figure 1(b) shows a reaction, which is an instantiation of a transform shown in Figure 1(c). The current state-of-the-art for encoding transforms is SMARTS [2] (an extension of SMILES with wildcards); e.g., Figure 1(d). SMARTS is an stopgap solution for representing (and encoding the application of) reaction transforms, and better representations are needed.

We present this as an open challenge to the community: Design a new language for biochemical transforms. This is both a syntactic challenge, as well as a semantic one. Syntactically, the language should succintly represent guarded transforms. Semantically, ideally the language will eliminate the need for solving a subgraph isomorphism problem whenever a transform is to be applied. SMARTS very crudely addresses the syntactic challenge, but makes no attempt to solve the semantic design consideration.

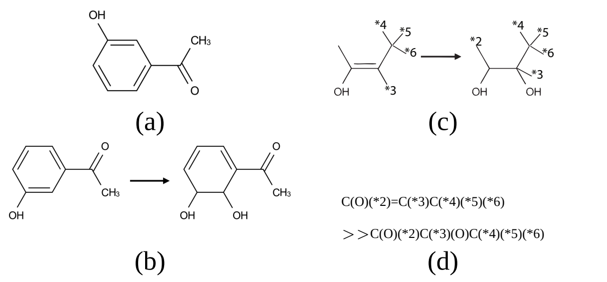
The fundamental issue making this problem non-trivial is the conflicting goals of having a potentially canonical representation of molecular graphs; and that of easily identifying a subgraph (the transform's guard) within the representation that can be syntactically modified to yield the output of the transform. The example in Figure 1 illustrates this difficulty.

3. Biochemical pathways: A reachability problem

A biochemical or chemical pathway is a chain of reactions producing sequences of chemical compounds (which optionally are described using the language in the previous section). Figure 2 shows an example pathway.

An important question in this space is the *prediction* of a pathway connecting a source compound to a desired target. The problem can be viewed as the inference of an acyclic program composed of individual statements, each of which applies a transform to its preceeding state (a compound). The additional restriction here is that the statements have precondition guards, and so are guarded commands.

[2] The SMILES linearization is trivial: Start from any atom in the molecule and read along bonds from it. If there is a branch away from it put it in parens. Ignore hydrogens. Cut rings at an arbitrary location and give it a number.



C(O)(*2)=C(*3)C(*4)(*5)(*6)

>>C(O)(*2)C(*3)(O)C(*4)(*5)(*6)

Figure 1. The problem in defining linear representations of cyclic structures: (a) An example molecular graph, whose SMILES representation can be CC(=O)C1=CC=CC(O)=C1 reading right to left and going clockwise. (b) An example reaction which is the result of transform (c) being applied to molecule (a). (d) The SMARTS string corresponding to this transform. Notice that because of the way the SMILES CC(=O)C1=CC=CC(O)=C1 was written, it is not syntactically obvious that this transform is applicable. Even ignoring the implicit 'H' that matches '*3' and '*5', there is no C(O)=CC in the SMILES representation, because the encoding choose to split the ring between the last two C's.

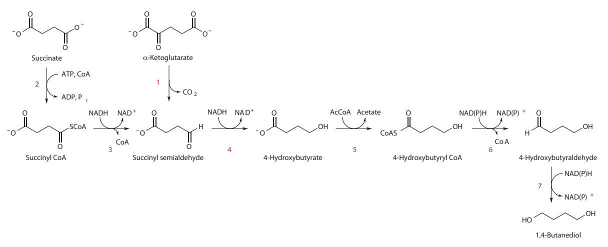
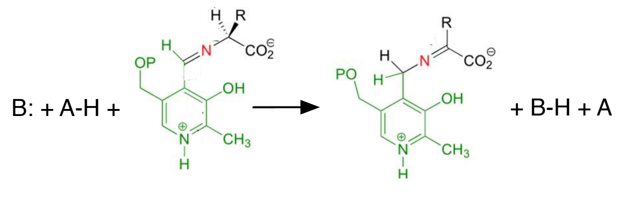


Figure 2. An example pathway [6, 14]. Note that there are two starting compounds.

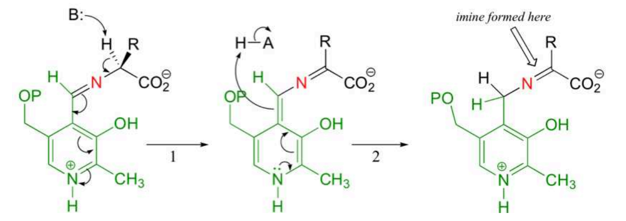
Given some starting chemicals, the search state is very large. If we restrict our attention to transforms that take only one input, then the space is finite and large, but not infinite, as the only possibilities are for the transform to rearrange the atoms and electrons within the molecule, as it cannot violate conservation of mass. If we include reactions with at least two molecules as inputs, then the search space is not only large, but potentially infinite, because a reaction could combine molecules, and thus create new yet unseen molecules ad infinitum. This problem will benefit significantly from efficient search space exploration techniques designed in the model checking and verification communities.

4. Mechanistic explanations of reactions: A synthesis problem

Where do the transforms we talk of in the previous two sections come from? The transforms encode fundamental organic chemistry knowledge and are patterns biochemists see periodically. The transforms are higher-level constructs, or abstractions, of underlying fundamental atomic transformations within the molecules. A transformation within a molecule is achieved by electron or proton movements. Se-



(a)



(b)

Figure 3. (a) A reaction that is summarization of atomic operations (b) A mechanism, i.e., series of electron movements, that explains the reaction in (a).

quences of these atomic changes *explain* reactions and are called *mechanisms*. Mechanisms describe in formal detail, all the electron and proton movements, and collisions between two reacting molecules. Bonds are shared electrons, so bond changes are electron movements as well. An example of a reaction mechanism is shown in Figure 3(b). A reaction that summarizes this mechanism is shown in Figure 3(a). What is abstracted in this summary is the particular sequence, and there could be alternative mechanisms to the same reaction. In the figure, each curved arrow indicates a movement of electrons (either part of a bond movement, or sitting unbound over atoms–indicated with dots–moving.)

The synthesis challenge then is to *infer mechanisms* given input and output compounds of reactions, i.e., from Figure 3(a) derive a series of electron movements (the arrow pushes) that yield Figure 3(b).

5. Other avenues

While we have discussed three specific problems that would benefit from language techniques, many more avenues remain undescribed. Some sample problems include sythesizing models explaining protein expression data (a synthesis problem for inferring a causal logic between the input sequence and the output expression), or verification of synthetic biology devices that would ideally take a model, as yet unavailable, of the cell as the environment.

A key enabler for the use of language techniques is the recognition that a probabilistic approach—employed in most current computational approaches to biology—is not the only choice for modeling, as we show in previous work [7]. Alternative symbolic descriptions are at times appropriate.

6. Conclusions

In this paper, we have skimmed the surface of the solutions to synthetic biology problems that can come from programming languages technology. Synthetic biologists these days routinely talk of design, compile, execute, test cycles [5](29:00). The two very critical phases of design and compile will benefit tremendously from our approaches, and additionally will require programming languages researchers to invent novel domain-specific changes to their techniques. For the specific case of biochemical needs within the field, we describe three fundamental problems that will inevitably see language solutions.

References

[1] `http://partsregistry.org/`.

[2] `http://www.daylight.com/dayhtml/doc/theory/theory.smarts.html`.

[3] Synthetic biology explained. `http://www.youtube.com/watch?v=rD5uNAMbDaQ`.

[4] *Nomenclature of organic chemistry*. Number v. 1 in Nomenclature of Organic Chemistry. International Union of Pure and Applied Chemistry, 1965.

[5] A. Hessel. Introduction to synthetic biology. `http://www.youtube.com/watch?v=niQ0kkgPxJk`.

[6] G. Inc. `http://www.genomatica.com/products/bdo/`.

[7] A. S. Koksal, Y. Pu, S. Srivastava, R. Bodk, J. Fisher, and N. Piterman. Synthesis of biological models from mutation experiments. In *POPL '13*, 2013.

[8] A. McNaught. The IUPAC intl chem identifier : InChI - a new std for molecular informatics. *Chem. Intl.*, 28(6):12–15.

[9] 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. 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):940–943, Apr 2006.

[10] C. Venter. On the verge of creating synthetic life. `http://www.youtube.com/watch?v=nKZ-GjSaqgo`.

[11] D. Weininger. SMILES, a chemical language and information system. 1. introduction to methodology and encoding rules. *J. Chem. Inf. Comput. Sci.*, 28(1):31–36, Feb. 1988.

[12] D. Weininger. SMILES, 3. DEPICT. graphical depiction of chemical structures. *J. of Chem. Inf. and Comp. Sc.*, 30(3):237–243, 1990.

[13] D. Weininger, A. Weininger, and J. L. Weininger. Smiles. 2. algorithm for generation of unique smiles notation. *J. of Chem. Inf. and Comp. Sc.*, 29(2):97–101, 1989.

[14] H. Yim, R. Haselbeck, W. Niu, C. Pujol-Baxley, A. Burgard, J. Boldt, J. Khandurina, J. D. Trawick, R. E. Osterhout, R. Stephen, J. Estadilla, S. Teisan, H. B. Schreyer, S. Andrae, T. H. Yang, S. Y. Lee, M. J. Burk, and S. Van Dien. Metabolic engineering of Escherichia coli for direct production of 1,4-butanediol. *Nat. Chem. Biol.*, 7(7):445–452, Jul 2011.