# The Crucial Role of CS in Systems and Synthetic Biology

## Biological cells, coerced to function as hardware and driven by artificial DNA, can perform such nanoscale tasks as detecting toxic substances and manufacturing new drugs.

In the coming decades the hardware-software paradigm that has been central to computer science since its inception may well be challenged—or at least complemented—by an exciting new development: synthetic biology. Biological cells will become an alternative to current hardware, and an analog of software will be engineered to direct cells to produce useful artifacts or substances. The hardware-software model will thus come increasingly close to mimicking miniature (nanosize) robotics that induce living organisms, like bacteria, which have existed in nature for billions of years, to assemble minuscule amounts of compounds. Scientists would thus be able to perform tasks that are still largely unimaginable today, like cleaning the environment, making new drugs, and detecting dangerous chemicals.

Computer science has always been in the vanguard of new scientific and engineering developments. The one I advocate here will help us move into the next frontiers. The emerging field of synthetic biology [1, 3, 4] is the engineering counterpart of designing computer-like biological machinery and its associated software, or wetware. Wetware indicates how programs that assemble the desired artifacts are constructed in a biological wet-lab, in contrast to the dry-lab used to assemble and program electronic components.

The goals of synthetic biology can be accomplished only if biologists and computer scientists fully comprehend the dynamic behavior of living cells. A discipline called systems biology [5], which encompasses synthetic biology, strives to understand dynamic cell behavior. Systems biology is essentially analytical, whereas synthetic biology deals with the engineering issues of changing known cell behavior to accomplish human goals.

My aim here is twofold: urge computer scientists to follow closely what is happening in synthetic and systems biology, participating actively in their development; and emphasize that the idealistic goals of systems and synthetic biology will not be feasible without the engaged contribution of computer scientists. Fulfilling these objectives will open inspiring new vistas for our field.

To illustrate the promise of synthetic and systems biology, consider a prototype experiment that demonstrates what is currently feasible in these new fields. A Petri dish containing a special strain of harmless bacteria sits on a laboratory bench at a research center. A slight fragrance of mint emanates from the dish. A scientist in a lab coat adds a few drops of a chemical, and the mint scent is quickly replaced by a strong odor of...bananas (web.mit.edu/newsoffice/2006/igem.html).

LISA HANEY

A comparable event occurred at the 2006 International Genetically Engineered Machines (iGEM) competition, which had brought together graduate students from 15 countries majoring in biology, computer science, and electrical engineering (parts2.mit.edu/wiki/index.php/Main_Page). The apparently innocuous test of changing a substance's fragrance by adding special ingredients has enormous practical implications, most notably in environmental remediation and pharmaceutical engineering. Imagine that the ingredient being added to the dish contains a dangerous chemical. The experiment allows investigators to detect the presence of the substance simply by exposing it to bacteria and having humans, animals, or even instrumentation detect the chemical changes generating the fragrance.

What does this have to do with computer science? The question and its answer are at the heart of synthetic biology. Its objective is to reengineer cells by changing or supplementing their DNA so the ensuing cell products are sensitive to given substances and can indicate and eventually eradicate their presence. For example, synthetic biology could potentially engineer harmless bacteria capable of detecting and absorbing oil spills or breaking down carbon dioxide in the atmosphere.

The entity that associates synthetic biology to computer science is DNA, which can be viewed as a program that remains static or dormant in a computer-like memory. Only when it is executed by processors—the equivalent of interpreters and hardware—does its dynamic behavior come to life.

From its origins in the early 1980s, bioinformatics, combining computer science and biology, has dealt mostly with the static properties of DNA and its products, like RNA and proteins. With the automation of DNA sequencing in the mid-1980s, the immediate goal was to obtain sequences of letters—A, C, G, and T—identifying the basic nucleotides that characterize all living matter, from bacteria to humans. The project of sequencing a variety of genomes is still formidable; a recent press release reported "The theoretical price of having one's personal genome sequenced just fell from the prohibitive $20 million to about $2.2 million, and the goal is to reduce the amount further—to about $1,000—to make individualized prevention and treatment realistic" [7].

Following the principles of Darwinian evolution, bioinformatics specialists have scrutinized similarities among the static DNA of various species. In fact, nearly all research in similarities to date has been done at the static level, without great concern for the dynamics triggered when DNA is processed by actors, like polymerases and ribosomes. Such actors are essentially the nanobiological machinery that processes DNA to produce proteins—the building blocks of life.

Current efforts in bioinformatics also seek to determine protein structure and function. Most research has concentrated on identifying the stable 3D shape of a static molecule, even though protein molecules have degrees of flexibility that are relevant in determining a molecule's function. The concern for static sequences and 3D structures is still amply justified, since studying the dynamics of DNA and protein interaction is nearly impossible without a thorough study of their static counterparts.

Protein function is specified through informal natural language sentences that describe the role of a protein in a living cell. This description must still be complemented with formal specifications by, for example, indicating the protein's role in a network of protein interactions. Computer scientists are needed to design these specifications.

Systems biology studies the dynamic properties of the interactions between DNA and its products. Even if this new field is viewed as a branch of bioinformatics, it is already an area of significant interest to biologists, computer scientists, control engineers, and mathematicians. In dealing with static DNA and its products, including protein sequences, fundamental computer science algorithms operate on strings of symbols. They search for approximate patterns in very long sequences, compare multiple sequences, and combine overlapping sequences. Optimization is the objective of the approximate pattern-matching of sequences; the algorithms are designed to minimize the cost of abstractly transforming one sequence into another.

The goal of systems biology is to scrutinize the

dynamics of cell behavior. For example, a certain protein *P* (produced by a gene *G*) is used to prevent the production of another protein *P'* by blocking the processing of gene *G'*. The computer science analog is keeping some parts of a program from being executed once the execution reaches a certain stage. This behavior is akin to Edsger Dijkstra's semaphores in regulating the dynamic behavior of concurrent programs [2].

Both systems and synthetic biology will require expertise in computer science way beyond the expertise necessary for processing sequences. In turn, these disciplines will challenge computer scientists with problems that are at the forefront of computer science today, including how to develop nanotechnology hardware, fault-tolerant circuit design, program verification, model checking, program synthesis from data, and data mining.

Systems biology deals with gene interactions, some involving hundreds, if not thousands, of genes. When some interactions go awry, cell behavior changes dramatically, resulting in situations like the uncontrollable growth of a cancer. Thus, the computer algorithms in systems and synthetic biology are akin to those used for finding bugs or incorrect behavior in large complex programs. Debugging is one of the most arduous tasks in program development. Nonetheless, computer scientists have already developed sophisticated tools to facilitate debugging, some applicable for finding faulty configurations in biological networks.

A microarray, or genome chip, is a silicon chip that has become a common tool in systems biology. It includes tens of thousands of minute wells, each with multiple short strands of DNA material representative of each gene (www.affymetrix.com/index.affx). Each strand is matched with its counterparts obtained through a wet-lab experiment involving the genes of the cell being studied. The degree of matching is measured by special scanners linked to computers. The measurements estimate the amounts of products generated by a gene. Microarrays can also be used by biologists to dynamically record the changes in gene products over time, as a cell is subjected to some external influence (such as food starvation or the effect of a drug).

Microarrays are also expensive, each costing hundreds of dollars, and tens or possibly hundreds of them may be necessary to study the gene interactions of a single cell. But, as with sequencing costs, microarray costs are decreasing, and the volume of data being generated by microarray experiments is gigantic, possibly surpassing the size of the available DNA data. Even though results of microarray experiments are coarse and may contain laboratory errors, they represent challenging problems for data-mining experts.

The computer algorithms in systems and synthetic biology are akin to those used for finding bugs or incorrect behavior in large complex programs.

Several research groups led by prominent computer scientists have immersed themselves in fascinating research involving the amalgamation of computer science and systems biology. For example, a team at the Weizmann Institute led by Ehud Shapiro has designed nanobiological processors that function as finite-state-automata to recognize desired sequences of DNA and eventually deliver drugs capable of correcting cell behavior that could lead to disease (www.wisdom.weizmann.ac.il/math/profile/scientists/shapiro-profile.html) [8].

As mentioned earlier, synthetic biology is a notable leading-edge effort in systems biology. Its aim is to use the existing "processing" capabilities of a cell (such as yeast or *E. coli*) to perform such tasks as cleaning the environment, detecting dangerous chemicals, and manufacturing drugs. J. Craig Venter, a pioneer in sequencing the human genome, is pursuing the goal of generating (in a wet-lab) very long sequences of artificially produced DNA (www.jcvi.org). The artificial DNA is being designed to perform the tasks involved in reengineering cell behav-

ior. Computer scientists in synthetic biology teams help design the DNA fragments that must be inserted into a living cell and verify that the resulting genetic network is robust in the sense that small variations within the engineered cell mechanism are tolerated and will not result in malfunction.

The annual iGEM competition mentioned earlier is backed by a number of corporations, including Microsoft, which has established a systems biology group at its Research Center in Cambridge, England; researchers there explore the applications of Milner's π-calculus—developed to check the properties of mobile hardware, like cell phones—in systems biology. Ensuring that within a certain area a given number of calls can be handled properly by a provider's technology has counterparts in systems biology. As such, a team led by Luca Cardelli, a Microsoft computer scientist in Cambridge, who previously engaged in research aimed at ensuring the correctness of distributed programs, is today developing formal languages to describe cell behavior [6].

Even if systems and synthetic biology experiments look simplistic, they are indeed the early prototypes of major advances in the field. With the help of computer science, researchers and engineers will achieve the progress needed to transform systems and synthetic biology from pure science to industrial-scale reality. C

## References


1. Campbell, A.M. Meeting report: Synthetic biology jamboree for undergraduates. *Cell Biology Education 4,* 1 (Spring 2005), 19–23.
2. Dijkstra, E.W. Solution of a problem in concurrent programming control. *Commun. ACM 8,* 9 (Sept. 1965), 569.
3. Gardner, T.S., Cantor, C.R., and Collins, J.J. Construction of a genetic toggle switch in Escherichia coli. *Nature 403,* 6767 (Jan. 2000).
4. Hopkin, K. Life: The next generation: Engineers and biologists team up to create synthetic biological systems. *The Scientist 18,* 19 (Oct. 2004).
5. Kitano, H. Systems biology: A brief overview. *Science 295,* (Mar. 2002).
6. Phillips, A., Cardelli, L., and Castagna, G. A graphical representation for biological processes in the stochastic pi-calculus. *Transactions in Computational Systems Biology LN in CS 4230* (Nov. 2006), 123–152.
7. Pollack, A. The race to read genomes on a shoestring, relatively speaking. *New York Times* (Feb. 9, 2008).
8. Shapiro, E. and Benenson, Y. Bringing DNA computers to life. *Scientific American 294,* 5 (May 2006), 45–51.



**Jacques Cohen** (jc@cs.brandeis.edu) is the TJX/Feldberg Professor of Computer Science in the Department of Computer Science at Brandeis University, Waltham, MA.





DOI: 10.1145/1342327.1342332