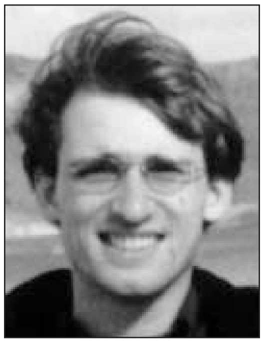
The engineering and programming of biochemical circuits, in vivo *and* in vitro, *could transform industries that use chemical and nanostructured materials.*

DNA Computing by Self-Assembly



Erik Winfree is an assistant professor in computer science and computation and neural systems at the California Institute of Technology.

Erik Winfree

Information and algorithms appear to be central to biological organization and processes, from the storage and reproduction of genetic information to the control of developmental processes to the sophisticated computations performed by the nervous system. Much as human technology uses electronic microprocessors to control electromechanical devices, biological organisms use biochemical circuits to control molecular and chemical events. The engineering and programming of biochemical circuits, *in vivo* and *in vitro*, would transform industries that use chemical and nanostructured materials. Although the construction of biochemical circuits has been explored theoretically since the birth of molecular biology, our practical experience with the capabilities and possible programming of biochemical algorithms is still very young.

In this paper, I will review a simple form of biochemical algorithm based on the molecular self-assembly of heterogeneous crystals that illustrates some aspects of programming *in vitro* biochemical systems and their potential applications. There are two complementary perspectives on molecular computation: (1) using the astounding parallelism of chemistry to solve mathematical problems, such as combinatorial search problems; and (2) using biochemical algorithms to direct and control molecular processes, such as complex fabrication tasks. The latter currently appears to be the more promising of the two.

Some major theoretical issues are common to both approaches—how algorithms can be encoded efficiently in molecules with programmable binding interactions and how these algorithms can be shown to be robust to asynchronous and unreliable molecular processes. Proof-of-principle has been experimentally demonstrated using synthetic DNA molecules; how well these techniques scale remains to be seen.

Algorithmic Self-Assembly as Generalized Crystal Growth

The idea of algorithmic self-assembly arose from the combination of DNA computing (Adleman, 1994), the theory of tilings (Grunbaum and Sheppard, 1986), and DNA nanotechnology (Seeman, 2003). Conceptually, algorithmic self-assembly naturally spans the range between maximal simplicity (crystals) and arbitrarily complex information processing. Furthermore, it is amenable to experimental investigation, so we can rigorously probe our understanding of the physical phenomena involved. This understanding may eventually result in new nanostructured materials and devices.

DNA Computing

Leonard Adleman's original paper on DNA computing contained the seed of the idea we'll pursue here—that the programmability of DNA hybridization reactions can be used to direct self-assembly according to simple rules. In the first combinatorial-generation step of Adleman's procedure, DNA molecules representing all possible paths through the target graph were assembled by DNA hybridization in a single step. The basic idea (Figure 1) is for a set of molecules with unique sequences to represent the vertices and edges of the graph, thus governing which vertices can follow which other vertices. Each possible sequence of hybridization reactions, occurring spontaneously in any order, produces a double-stranded DNA molecule whose sequence encodes a valid path through the graph. By thus generalizing one-dimensional polymerization to include programmable binding, Adleman coaxed the DNA to generate patterns that follow certain mathematical rules. This is an elegant idea—and it works! The problem is that only simple computations can be performed with linear self-assembly. Paths through graphs correspond to regular languages, which have the complexity of finite-state machines—thus more sophisticated aspects of computation cannot be reached by this technique.

Tiling Theory

A tiling is an arrangement of a few basic shapes (called tiles) that fit together perfectly in the infinite plane. For each tiling, the set of shapes must be finite; for example, the tile set could consist of an octagon and a square, both with unit-length sides. One motivation for studying tiling is that the tiles correspond to the periodic arrangement of atoms in crystals. A remarkable result is that all possible periodic arrangements can be classified according to their fundamental symmetries; in three dimensions there are 230 symmetries, and in two dimensions there are 17 symmetries. This suggests that, given a finite set of polygonal tiles, one should be able to determine whether they can be arranged according

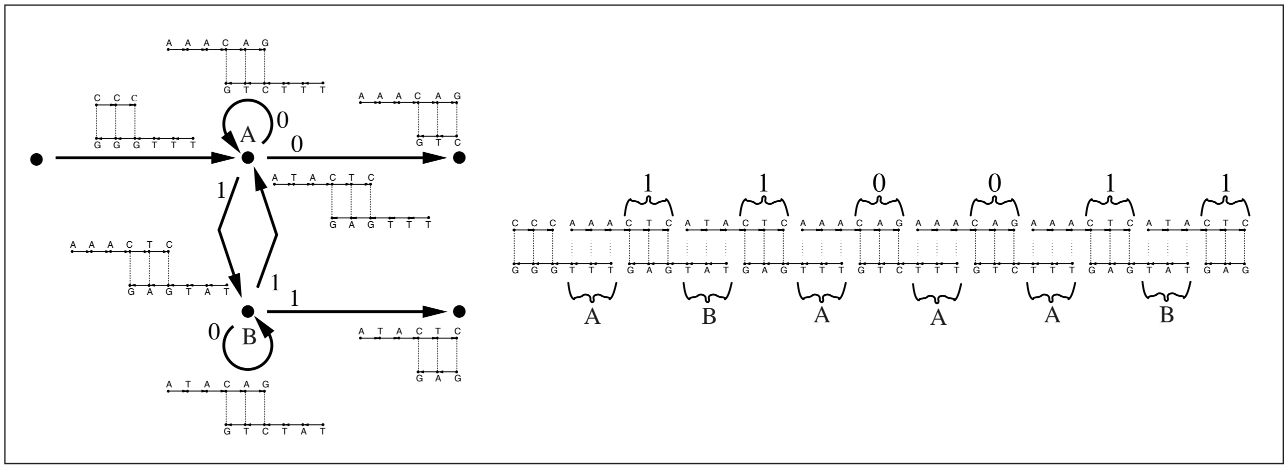


FIGURE 1 Linear self-assembly of DNA can be directed to follow valid paths through a graph. Sequences used in practice would have 15–30 nucleotides for each domain, rather than 3 nucleotides as shown here.

to one of the known symmetries, or whether there is no way to arrange them on the plane.

This is what Hao Wang thought in the 1960s, but when he looked into the question, known as the tiling problem, he discovered that it is provably unsolvable (Wang, 1963)! That is to say, aperiodic tilings are also possible. In addition, it can be incredibly difficult to determine whether a given set of tiles can tile the plane aperiodically or whether every attempt will ultimately fail. To prove this result, Wang developed a way to create a set of tiles that fit together uniquely to reproduce the space-time history of any chosen Turing[1] machine, in such a way that, if the Turing machine halts (with an output), then the attempted tiling has to get stuck; if the Turing machine continues computing forever, then a consistent global tiling is possible.

Thus, the tiling problem reduces to the halting problem, the first problem proved to be formally undecidable. This result shows that tiling is theoretically as powerful as general-purpose computers. In fact, the tiles Wang used were all essentially square, distinguished only by labels on their sides that had to match up when the tiles were juxtaposed. Thus, the complexity arises from the logical constraints in how the tiles fit together, rather than from the tiles themselves.

Given the intimate relation between crystals and tiling theory, it is natural to ask if crystal growth has the potential to compute as powerfully. To answer this question, we need two things: (1) the ability to design molecular Wang tiles; and (2) precise rules for crystal growth that can be implemented reliably.

DNA Nanotechnology

We now turn to DNA nanotechnology, the brainchild of Nadrian Seeman's vision of using DNA as an architectural element. Like RNA, DNA can make structures other than the usual double helix. These other structures include hairpins and three- and four-way branch points, which are important for biological function. Seeman, however, pictured these structures as hinges and joints, bolts and braces that could be programmed to fold and bind to each other by careful design of the DNA base sequence. Seeman and his students constructed a wide variety of amazing nanostructures: a wire-frame cube and truncated octahedron; single-stranded DNA and RNA knots, including the trefoil, the figure-eight, and Borromean rings; and rigid building-block structures, such as triangles and four-armed "bricks" known as double-crossover (DX) molecules; and more (Seeman, 2003).

The idea, then, is to use these "bricks" as molecular Wang tiles (Winfree et al., 1998a). The four arms of the DX molecules can be given sequences corresponding to the labels on the four sides of the Wang tiles. Thus, any chosen Wang tile can be implemented as a DNA molecule. Appropriate design of the molecule will encourage assembly into two-dimensional sheets.

In the 1960s, Hao Wang discovered that the tiling problem is provably unsolvable.

The problem, then, is to ensure that the growth process results in tile arrangements in which all tiles match with their neighbors. It is easy, however, to envision ways of putting the tiles together so that the tiles match at each step but soon create a configuration for which there is no way to proceed without creating a mismatch or having to remove offending tiles. This situation is analogous to the distinction between uncontrolled precipitation, which occurs rapidly when there is a strong thermodynamic advantage to aggregation, and quality crystal growth, which occurs slowly when there is a slight thermodynamic advantage for molecules that bind in the preferred orientation, but other possible ways to bind are disadvantageous.

A formalization of this notion for Wang tiles, the Tile Assembly Model, supposes that each label on a Wang tile binds with a certain strength (typically, 0, 1, or 2) and that tiles will only stick to a growing assembly if they bind (possibly via multiple bonds) with a total strength greater than some threshold τ (typically 1 or 2); tiles that bind with a weaker strength immediately fall off (Winfree, 1998). Under these rules, growth from a "seed tile" can result in a unique, well defined pattern. Because Turing machines and cellular automata can be simulated by this process, the Turing-universality of tiling is retained.

As an example, consider the seven tiles shown in Figure 2 assembling at $\tau = 2$. These tiles perform a simple computation—they count in binary. Starting with the seed tile, labeled *S*, the tiles with strength-2 bonds polymerize to form a V-shaped boundary for the computation. There is a unique tile that can fit into the nook of the V; because it makes *two* strength-1 bonds, it can in fact be added. Two new nooks are created, and again a unique tile can be added in each location. The assembly thus grows forever, counting and counting with unabated madness.

Tiles can be added in any order, but the resulting pattern is the same. The same basic self-assembly mechanisms used here are sufficient to perform more sophisticated computations. No new ideas or mechanisms are necessary to obtain fully programmable Turing-universal behavior.

Experimental Advances

The first demonstration of these ideas—two-dimensional, periodic arrays of DNA tiles—could hardly be called "algorithmic," but it did show that the sequences given to the tiles' sticky ends could be used to program different periodic arrangements of tiles (Winfree et al., 1998a). The encoding of tiles as DNA DX molecules is illustrated in Figure 3; Figure 4 shows small crystals of DX molecules adsorbed on mica, as they appear in the atomic force microscope. Subsequent studies have shown that DNA tiles can be made from a variety of different molecular structures. Thus, the principle that the arrangement of two-dimensional tiles can be directed by programmable, sticky-end interactions appears to be quite robust.

The goal of creating three-dimensional, periodic arrays of DNA tiles, originally formulated by Seeman more than 20 years ago, remains an open problem in the field. Once solved, it will allow for more sophisticated information-processing techniques in algorithmic self-assembly, roughly analogous to the increase in power from one-dimensional to two-dimensional cellular automata or Turing machines.

For the time being, experimental demonstration of algorithmic self-assembly has been confined to one- and two-dimensional assemblies. The first use of one-dimensional algorithmic self-assembly appeared as the first step in Adleman's original DNA-based computing demonstration; this process formally corresponds to the generation of languages by finite-state machines. Furthermore, using one-dimensional, tile-based assembly, it is possible to read an input string (encoded as a one-dimensional tile assembly) and generate an output string consisting of the cumulative[2] exclusive-OR (XOR) of the input string (Mao et al., 2000); this formally corresponds to a finite-state transducer.

The first two-dimensional, algorithmic self-assembly process to be experimentally demonstrated with DNA is a generalization of the one-dimensional XOR example (Rothemund and Winfree, in preparation). Beginning with an input row consisting of a single 1 in a sea of 0's, the next layer grows by placing a 0 where both neighbors in the layer below are the same and a 1 where they are different. This process, an example of a one-dimensional cellular automaton, generates a fractal pattern known as the Sierpinski gasket.

In addition to the DNA required to construct the input, only four DNA tiles are required (in principle) to grow arbitrarily large Sierpinski triangles. Experimentally, error-free Sierpinski

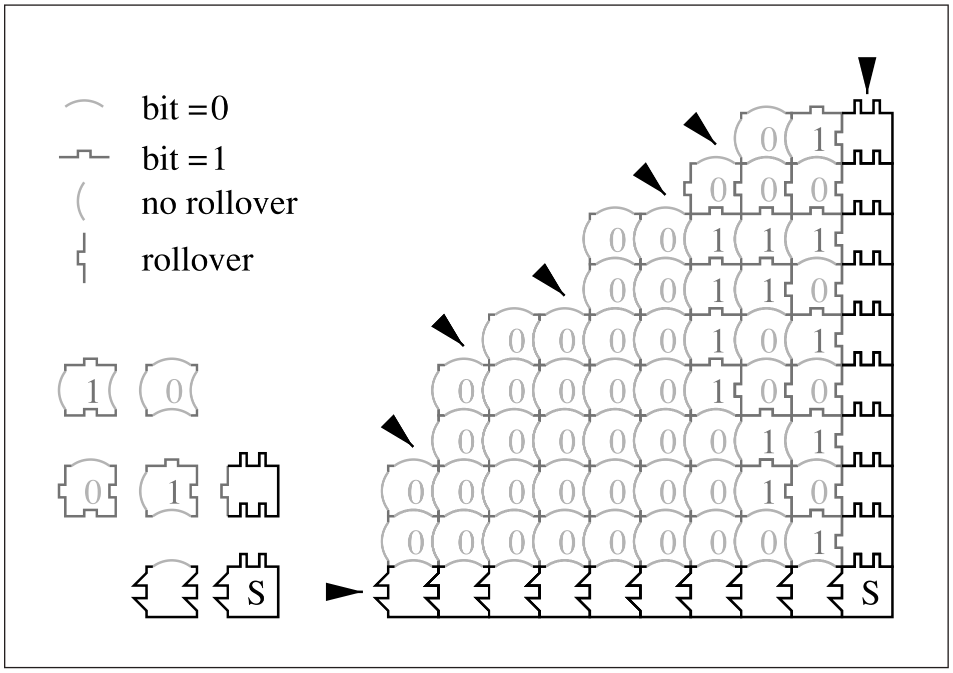


FIGURE 2 A set of seven tiles that implement a binary counter when started with the seed tile *S*. Strength-2 bonds are indicated by tile sides with *two* projections (or indentations); other bonds have strength 1. Arrows indicate sites where a tile may be added at $\tau = 2$.

triangles as large as 8 x 16 have been observed by atomic force microscopy. However, error rates (the frequency with which the wrong tile was incorporated into the crystal) ranged from 1 to 10 percent, and many fragments appeared to have grown independently of the input structure. It is clear that controlling nucleation and finding mechanisms to reduce the error rates are critical challenges for making algorithmic self-assembly practical.

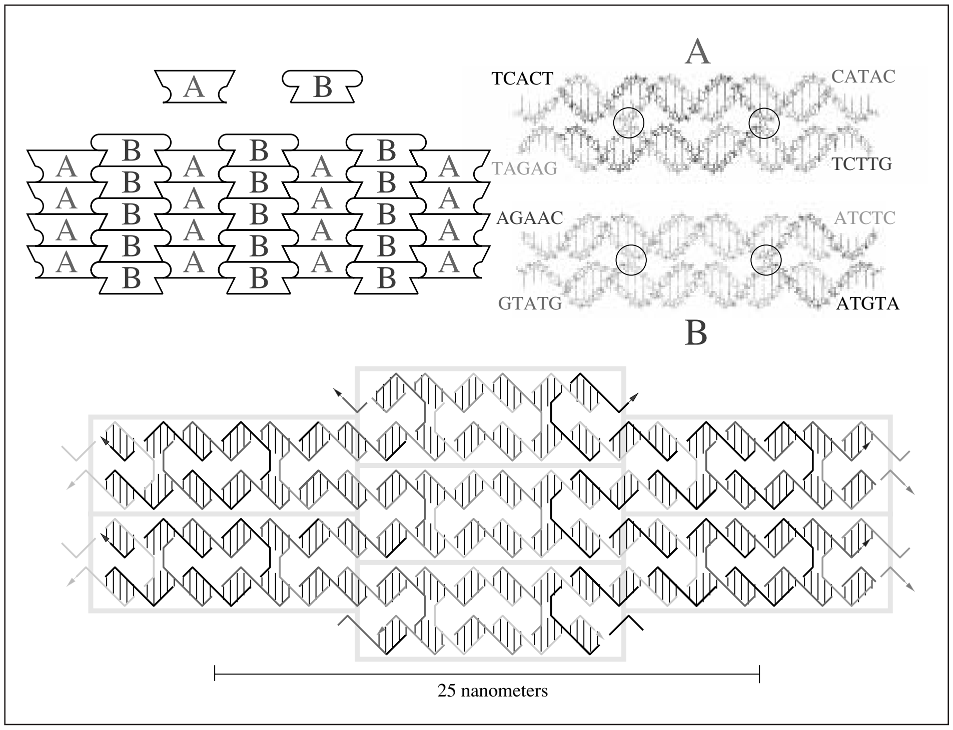


FIGURE 3 DNA double-crossover molecules can implement abstract Wang tiles, producing a two-dimensional lattice of DNA with binding interactions dictated by the DNA sticky ends.

Potential Technological Applications

Combinatorial Optimization Problems

Solving combinatorial optimization problems, in the spirit of Adleman's original paper, was the first application considered for algorithmic self-assembly. Adleman's essential insight is based on the fact that a class of hard computational problems, the NP-complete problems, share a common generate-and-test form—does a sequence exist that satisfies easy-to-check properties X, Y, …, and Z. All known algorithms for NP-complete problems require exponential[3] time or exponential parallelism. The basic idea is to use combinatorial chemistry techniques to simultaneously generate all potential solutions and then to filter them, based on chemical properties related to the information they encode, leaving at the end possibly only a single molecule that has all of the desired properties. If the final solution to the problem is defined by satisfying a small number of simple properties—as is the case for all NP-complete problems—then this approach can be used to find the solution in a short amount of time, if the parallelism is sufficient. That a single cc of DNA in solution at reasonable concentrations can contain 2^{60} bits of information—which can be acted on simultaneously by chemical operations—gives us hope that the parallelism could be sufficient.

By exploiting the situation in which multiple different tiles could be added at a given location—much like Adleman's assembly step that produced all possible paths through a graph—self-assembly can generate a combinatorial set of possible assemblies and then continue growing according to a process that tests the information to see if it has the desired properties. Theoretical schemes have been worked out that use a single self-assembly step to solve the Hamiltonian path problem (HPP) (Winfree et al., 1998b), solve the Boolean formula satisfiability problem (SAT) (Lagoudakis and LaBean, 2000), and perform other math calculations (Reif, 1997). How much computation could be done this way? If assembly were to proceed with few errors, solving a 40-variable SAT problem would require 30 milliliters of DNA at a tile concentration of 1 micromolar and might be completed in a few hours. This "best possible" estimate corresponds to 10^{12} bit operations per second—not bad for chemistry but still low compared to electronic computers.

The sheer speed and flexibility of silicon-based electronic computers make them preferable to DNA computing, even if self-assembly were to proceed without errors. We can conclude, then, that the low-hanging fruit are not to be found in the field of combinatorial search. But the ability of self-assembly to perform sophisticated computations suggests that we are making progress toward our goal of understanding (and

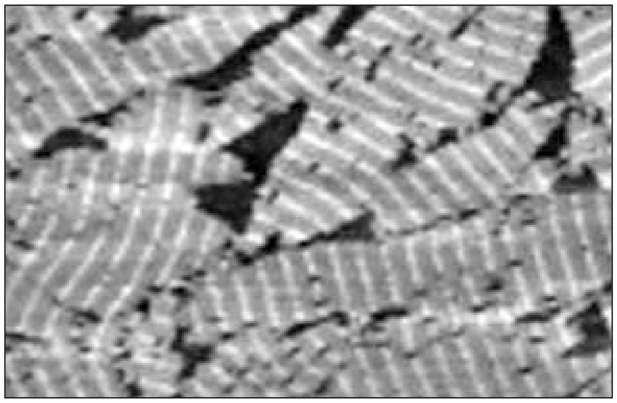


FIGURE 4 Atomic force microscope image of DNA double-crossover crystals. Stripes are spaced at 25 nm; individual 2 x 4 x 13 nm tiles are visible.

potentially exploiting) autonomous biochemical algorithms. A more promising application is suggested by examining how self-assembly is used in biology.

Programmable Nanofabrication

Biology uses algorithmically controlled growth processes to produce nanoscale and hierarchically structured materials with properties far beyond the capability of today's human technology. Does DNA-based algorithmic self-assembly give us access to new and useful technological capabilities? The simplest applications would make use of self-assembled DNA as a template or scaffold for arranging other molecular components into a desired pattern. This could be used for biochemical assays, novel materials, or devices. Seeman has envisioned, for example, using periodic three-dimensional DNA lattices to assist with difficult protein crystallization or to direct construction of molecular electronic components into a memory (Robinson and Seeman, 1987).

The potential of self-assembly for fabricating molecular electronic circuits is intriguing, given the limitations of conventional silicon-circuit fabrication techniques. Photolithography is unable to create features significantly smaller than the wavelength of light, and even if it could, for several-nanometer line widths the unspecified atomic positions within the silicon substrate would lead to large stochastic fluctuations in device function. For these reasons, many researchers are investigating electrical computing devices created from molecular structures, such as carbon nanotubes, in which the location of every atom is well defined. However, an outstanding problem is how to arrange these chemical components into a desired pattern.

DNA self-assembly could be used in a variety of ways to solve this problem: molecular components (e.g., AND, OR, and NOT gates, crossbars, routing elements) could be chemically attached to DNA tiles at specific chemical moieties, and subsequent self-assembly would proceed to place the tiles (and hence circuit elements) into the appropriate locations. Alternatively, DNA tiles with attachment moieties could self-assemble into the desired pattern, and subsequent chemical processing would create functional devices at the positions specified by the DNA tiles. None of these approaches has yet been convincingly demonstrated, but it is plausible that any of them could eventually succeed to produce two- or three-dimensional circuits with nanometer resolution and precise control of chemical structure.

Using self-assembly to direct the construction of circuits as large and complex as those found in modern microprocessors is daunting. The question arises, therefore, of whether there are useful circuit patterns that can be generated by a feasibly small number of tiles. Any circuit pattern that has a concise algorithmic description is a potential target for this approach. Small tile sets have been designed for demultiplexers, such as the ones necessary to access a RAM memory (shown in Figure 5), and for signal-processing primitives, such as the Hadamard matrix transform (Cook et al., in press). Regular gate arrays, such as those used in cellular automata and field programmable gate arrays (FPGAs), are another natural target for algorithmic self-assembly of circuits.

Many technical hurdles will have to be overcome before algorithmic self-assembly can be developed into a practical commercial technology. It is not clear if real circuits will ever be built this way, but the sheer range of possibilities opened up by algorithmic growth processes suggests that algorithmic self-assembly will be used in the future for technologies that place molecular components in a precisely defined complex organization.

Summary and Prospects

DNA-based self-assembly appears to be a robust, readily programmable phenomenon. Periodic two-dimensional crystals have been demonstrated for tens of distinct types of DNA tiles, illustrating that in these systems the sticky ends drive the interactions between tiles. Several factors limit immediate applications, however. Unlike high-quality crystals, current DNA tile lattices are often slightly distorted, with the relative position of adjacent tiles jittered by a nanometer and lattice defect rates of 1 percent or more. Some DNA tiles designed to form two-dimensional sheets appear to prefer tubes, for better or worse. Furthermore, procedures have yet to be worked out for reliably growing large (greater than 10 micron) crystals and depositing them nondestructively on the substrate of choice.

Although one- and two-dimensional algorithmic self-assembly has been demonstrated, per-step error rates between 1 and 10 percent preclude the execution of complex algorithms. Recent theoretical work has suggested the possibility of error-correcting tile sets for self-assembly, which, if demonstrated experimentally, would significantly increase the feasibility of interesting applications. A second prevalent source of algorithmic errors is undesired nucleation (analogous to programs starting by themselves with random input). Thus controlling nucleation, through careful exploitation of supersaturation and tile design, is another active topic of research. Learning how to obtain robustness to other natural sources of variation—lattice defects, ill-formed tiles, poorly matched sticky-end strengths, changes of tile concentrations, temperature, and buffers—will also be necessary.

Presuming that algorithmic self-assembly of DNA can be made more reliable, it then becomes important that we understand the logical structure of self-assembly programs and how that structure relates to and differs from existing models of computation. At the coarse scale of what can be computed—at all—by self-assembly of DNA tiles, there is a natural parallel to the Chomsky hierarchy of formal language theory. Recent theoretical work by Adleman, Goel, Reif, and others, has focused on two issues of efficiency: (1) the kinds of shapes and patterns that can be assembled using a small number of tiles; and/or (2) the kinds of shapes and patterns that can be assembled with rapid assembly kinetics.

To what extent has this investigation enlightened us about how information and algorithms can be encoded in biochemical systems? First, it is intrinsically interesting that self-assembly can support general-purpose computation, although it looks very different from conventional electronic computational circuits. At first glance, other biochemical systems, such as *in vivo* genetic regulatory circuits, appear to have a structure more similar to conventional electronic circuits. But we should be prepared for differences that dramatically alter how the system can be efficiently programmed. Ever-present randomness, pervasive feedback, and a tendency toward energy minimization are unfamiliar factors for computer scientists to consider. Nevertheless, functional computation can be hidden in many places!

Thus, DNA self-assembly can be seen as one step in the quest to harness biochemistry in the same way we have harnessed the electron. Electronic computers are good at (and pervasive at) embedded control of

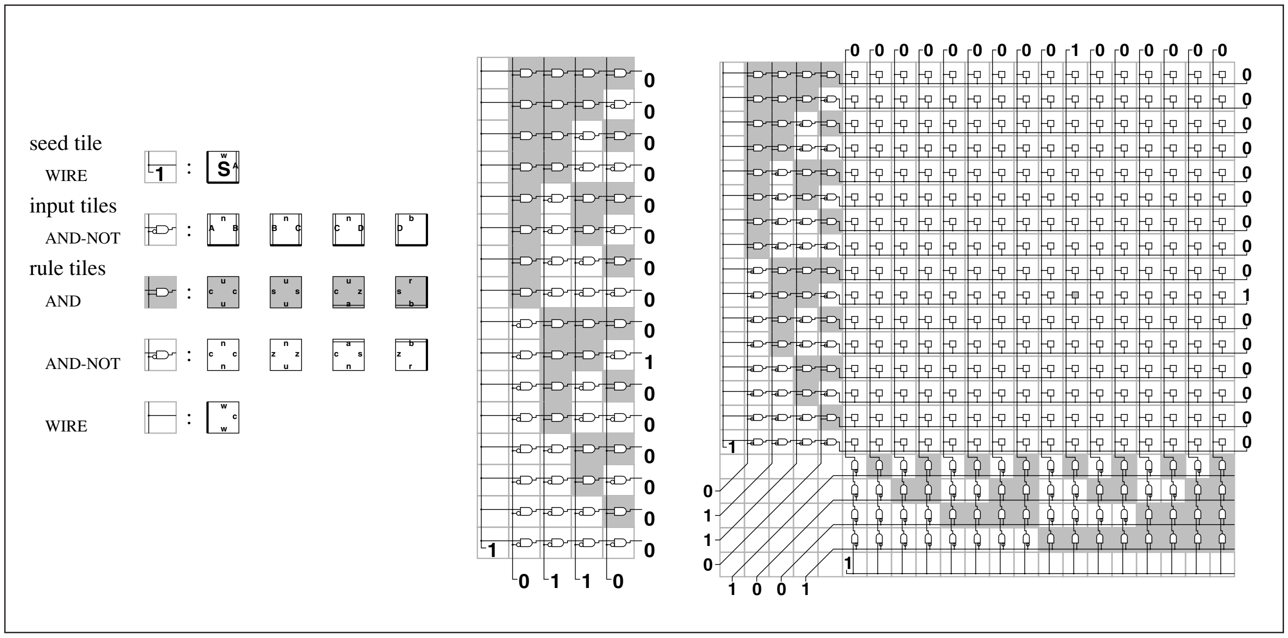


FIGURE 5 Using self-assembly of DNA tiles to create a molecular-scale pattern for a RAM memory with demultiplexed addressing. The tile set is closely related to the binary counter.

macroscopic and microscopic electromechanical systems. We don't yet have embedded control for chemical and nanoscale systems. Programmable, algorithmic biochemical systems may be our best bet.

References

Adleman, L.M. 1994. Molecular computation of solutions to combinatorial problems. Science 266(5187): 1021–1024.

Cook, M., P.W.K. Rothemund, and E. Winfree. In press. Self-assembled circuit patterns. DNA Based Computers 9.

Grunbaum, B., and G.C. Shephard. 1986. Tilings and Patterns. New York: Freeman.

Lagoudakis, M.G., and T.H. LaBean. 2000. 2D DNA Self-Assembly for Satisfiability. Pp. 141–154 in DNA Based Computers V, E. Winfree and D.K. Gifford, eds. Providence, R.I.: American Mathematical Society.

Mao, C., T.H. LaBean, J.H. Reif, and N.C. Seeman. 2000. Logical computation using algorithmic self-assembly of DNA triple-crossover molecules. Nature 407(6803): 493–496.

Reif, J. 1997. Local Parallel Biomolecular Computing. Pp. 217–254 in DNA Based Computers III, H. Rubin and D.H. Wood, eds. Providence, R.I.: American Mathematical Society.

Robinson, B.H., and N.C. Seeman. 1987. The design of a biochip: a self-assembling molecular-scale memory device. Protein Engineering 1(4): 295–300.

Seeman, N.C. 2003. Biochemistry and structural DNA nanotechnology: an evolving symbiotic relationship. Biochemistry 42(24): 7259–7269.

Wang, H. 1963. Dominoes and the AEA Case of the Decision Problem. Pp. 23–55 in Mathematical Theory of Automata, J. Fox, ed. Brooklyn, N.Y.: Polytechnic Press.

Winfree, E. 1998. Simulations of Computing by Self-Assembly. Caltech Computer Science Technical Report 1998.22. Pasadena, Calif.: California Institute of Technology.

Winfree, E., F. Liu, L.A. Wenzler, and N.C. Seeman. 1998a. Design and self-assembly of two-dimensional DNA crystals. Nature 394(6693): 539–544.

Winfree, E., X. Yang, and N.C. Seeman. 1998b. Universal Computation via Self-Assembly of DNA: Some Theory and Experiments. Pp. 191–214 in DNA Based Computers II, L.F. Landweber and E.B. Baum, eds. Providence, R.I.: American Mathematical Society.

Endnotes

1 Turing machines, invented by Alan Turing in 1936, are extremely simple computers that consist of a finite-state compute head that can move back and forth on an infinite one-dimensional memory tape. Turing showed that these machines are universal in the sense that they can perform any computation that can be performed by any other mechanical device—there is no fundamental need to use a more complicated kind of computer!

2 The n^{th} bit of the cumulative XOR gives the parity of the first n bits of the input sequence.

3 Exponential in the length of the problem description, in bits.