DNA Computing by Self-Assembly

ERIK WINFREE
Departments of Computer Science and Computation & Neural Systems
California Institute of Technology
Pasadena, California

INFORMATION AND ALGORITHMS IN BIOCHEMISTRY

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 electro-mechanical devices, biological organisms use biochemical circuits to control molecular and chemical events. The ability to engineer and program biochemical circuits, *in vivo* and *in vitro*, is poised to transform industries that make use of chemical and nano-structured materials. Although the possibility of constructing biochemical circuits has been explored theoretically since the birth of molecular biology, our practical experience with what biochemical algorithms are capable of and how they can be programmed is very young.

In this paper, I review a simple form of biochemical algorithm, based on molecular self-assembly of heterogeneous crystals, that illustrates some aspects of programming *in vitro* biochemical systems and the potential applications. Two complementary perspectives on molecular computation can be considered: using the astounding parallelism of chemistry to solve mathematical problems, such as combinatorial search problems; and using the biochemical algorithms to direct and control molecular processes, such as complex fabrication tasks. The latter currently appears to be more promising. Major theoretical issues are common to both approaches: how algorithms can be efficiently encoded 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 will 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 & Sheppard, 1986), and DNA nanotechnology (Seeman, 1982; Seeman, 2003). Conceptually, algorithmic self-assembly naturally spans the range between maximal simplicity – the periodic order of crystals – and arbitrarily complex information processing. Furthermore, it is amenable to experimental investigations, allowing us to rigorously probe our understanding of the physical phenomena involved. This understanding may eventually result in new nano-structured materials and devices.

DNA Computing

Leonard Adleman's original paper on DNA computing already 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 desired 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. As shown in Figure 1, the basic idea is to have a set of molecules with unique sequences representing the vertices and edges of the graph, thus governing which vertices can follow which other vertices. Each possible sequence of hybridization reactions produces a double-stranded DNA molecule whose sequence encodes a valid path through the graph. By thus generalizing one-dimensional (1D) 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 done with linear self-assembly: paths through graphs correspond to regular languages, which are recognized by finite-state machines – and consequently most interesting computations cannot be performed.

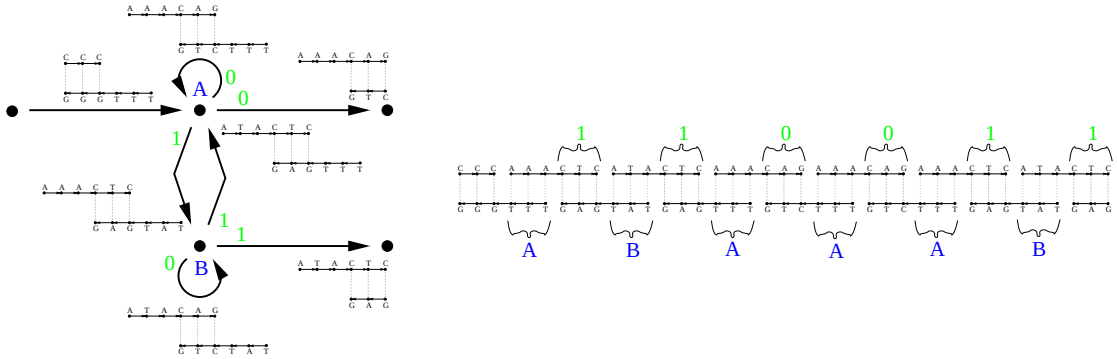

FIGURE 1 Linear self-assembly of DNA can be directed to follow valid paths through a graph. Sequences used in practice would have 15-30 nt for each domain, rather than 3 nt as shown here. Adapted with permission from Winfree, et al, 1998c.

Tiling Theory

Let's now turn to the geometer's theory of tiling. A tiling is an arrangement of basic shapes (such as an octagon and a square) such that they fit together perfectly in the infinite plane. One of the motivations for studying tiling is that some tilings 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 to one of the known symmetries, or whether there is no way to arrange them on the plane. So thought Hao Wang in the 1960's, but upon looking into the question, known as the tiling problem, he discovered that the tiling problem is provably unsolvable (Wang, 1963)! This derives from the fact

that aperiodic tilings are also possible, and it can be incredibly difficult to determine whether a given set of tiles can tile the plane aperiodically, or whether every such attempt ultimately fails. 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 machine[1]. If the Turing machine halts (with an output) then the attempted tiling has to get stuck, whereas 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 markings on their sides that are required to match when tiles are juxtaposed (see Figure 2). Thus, the complexity arises from the logical constraints inherent in how the tiles fit together rather than in the tiles themselves.

Given the intimate relation between crystals and tiling theory, it is natural to ask, does crystal growth have the potential to compute as powerfully? In order to find an answer, what we need is (a) the ability to design molecular Wang tiles, (b) precise rules for crystal growth that can be implemented reliably.

DNA Nanotechnology

We now turn to DNA nanotechnology, the brain-child of Nadrian Seeman's vision of using DNA as an architectural element (Seeman, 1982). Like RNA, DNA can make structures other than the usual double helix. Structures such as hairpins and 3- and 4-way branch points are important for biological function. In Seeman's mind, however, such structures were 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 went on to construct a wide variety of amazing nanostructures: a wire-frame cube and truncated octohedron; single-stranded DNA and RNA knots including the trefoil, figure-8, and Borromean rings; 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, 1998b). 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 (2D) sheets. The problem, then, is how to ensure that the growth process results in tile arrangements where all tiles match with their neighbors. Unfortunately, for most tile sets there are many conceivable assembly sequences that create no mismatch, but soon reach a configuration from which there is no way to proceed without creating a mismatch or removing offending tiles. This situation, in which tiles stick anywhere they match at all, is analogous to chemical precipitation – rapid uncontrolled growth when

[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 can be performed by any other mechanical device – there is no fundamental need to use a more complicated kind of computer!

there is a strong thermodynamic advantage to aggregation. We are therefore interested in 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 (Winfree, 1998a) supposes that each label on a Wang tile binds with a certain strength (typically, 0, 1, or 2), and 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. 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.

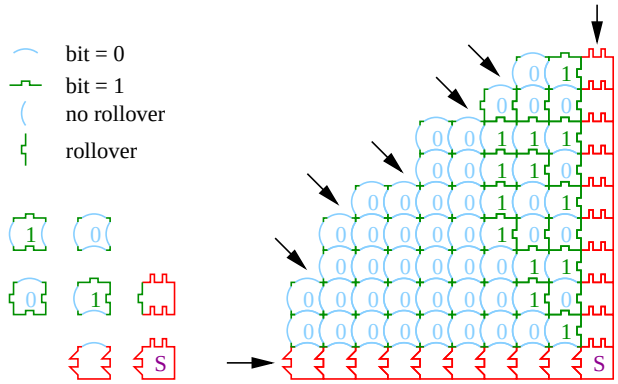

FIGURE 2 A set of seven tiles that implement a binary counter when started with the seed tile S. Arrows indicate sites where a tile may be added at τ=2. Adapted with permission from Winfree, 2000.

As an example, consider the seven tiles shown in Figure 2 assembling at τ=2. These tiles perform a simple computation: they count in binary. Starting with the seed tile, labeled "S", the red tiles make use of their strength-2 bonds to polymerize a V-shaped boundary for the computation. There is a unique tile that can fit in 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. Continuing, the assembly grows forever, counting and counting with unabated madness. It should be pointed out that tiles can be added in any order, but the resulting pattern is the same. Performing more sophisticated computations in the Tile Assembly Model is not fundamentally different.

EXPERIMENTAL ADVANCES

The first demonstration of these ideas (Winfree et al, 1998b) – 2D periodic arrays of DNA tiles – could hardly be called "algorithmic," but it did demonstrate that the sequences given to the tiles' sticky ends could be used to program different periodic arrangements of tiles. 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 (AFM). Subsequent studies have shown that crystal-forming DNA tiles can be made from a variety of different molecular structures; the principle that the arrangement of tiles can be directed by programmable sticky-end interactions appears to be quite robust.

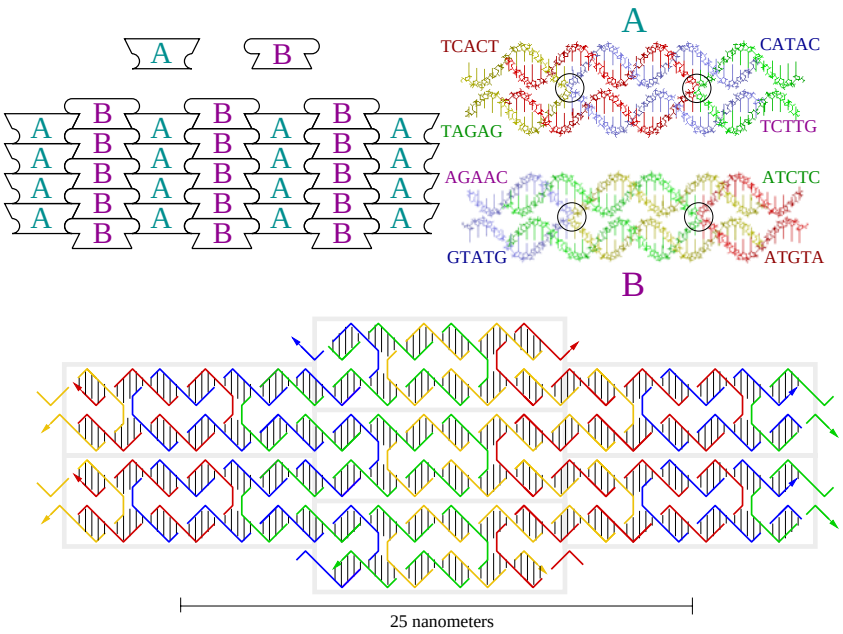

FIGURE 3 DNA double-crossover molecules can implement abstract Wang tiles, producing a 2D lattice of DNA with binding dictated by the DNA sticky ends. Adapted with permission from Winfree, 2000.

The goal of creating three-dimensional periodic arrays of DNA tiles, originally formulated by Seeman over 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, an advance analogous to the increase in computational efficiency gained by moving from 1D to 2D cellular automata or Turing machines.

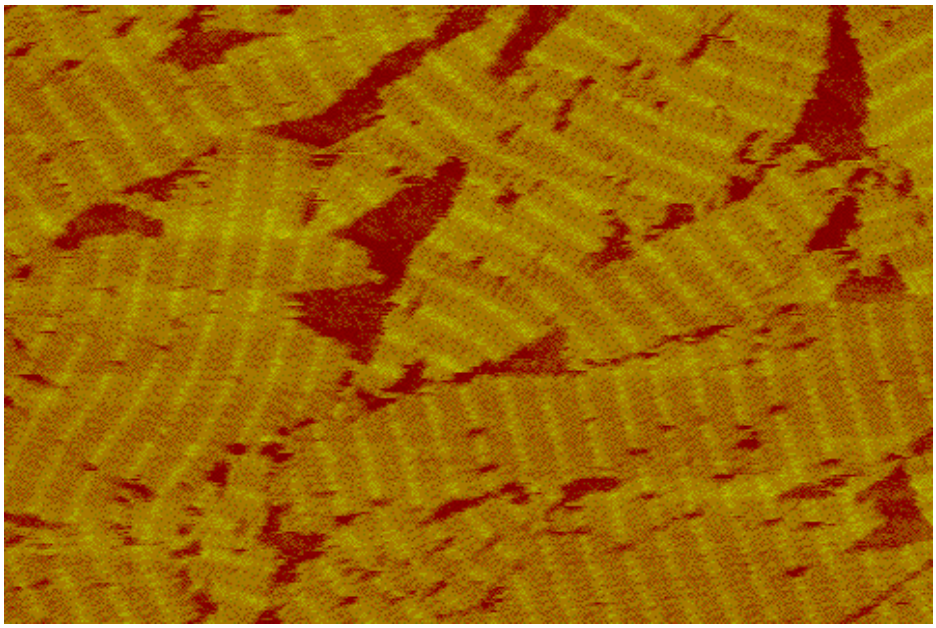

FIGURE 4 AFM image of DNA double-crossover crystals. Stripes are spaced 25nm; individual 2x4x13nm tiles are visible. Image taken by Nick Papadakis, Winfree lab.

For the time being, experimental demonstration of algorithmic self-assembly has been confined to 1D and 2D assemblies. The first use of 1D 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 1D tile-based assembly, it is possible to read an input string (encoded as a 1D tile assembly) and generate an output string consisting of the cumulative XOR of the input string (Mao et al, 2000); generalized, this formally corresponds to a finite-state transducer.

The first 2D algorithmic self-assembly process to be experimentally demonstrated with DNA (Rothemund & Winfree, in preparation) is a generalization of the 1D XOR example. 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 instance of a 1D cellular automaton, generates a triangular 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 triangles as large as 8x16 have been observed by atomic force microscopy. However, error rates (the frequency with which the wrong tile was incorporated into the crystal) ranged between 1 to 10%, and many fragments were observed that 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.

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 rests on the fact that a class of hard computational problems, the NP-complete problems, all share a common generate-and-test form: "Does there exist a sequence that satisfies easy-to-check properties X, Y, …, and Z?" All known algorithms for NP-complete problems require exponential[2] 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 the 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 milliliters 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.

[2] Exponential in the length of the problem description, in bits.

By exploiting the situation where there are multiple different tiles that could be added at a given location – much like Adleman's assembly step that produced all possible paths through a graph – self-assembly of tiles can generate a combinatorial set of possible assemblies. Subsequent growth processes test the information to see if it has desired properties. Theoretical schemes have been worked out that use a single self-assembly step to solve HPP (Winfree et al, 1998c), SAT (Lagoudakis & 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.

As with other DNA computing approaches, however, the sheer speed and flexibility of silicon-based electronic computers make them the preferred method, 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 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 nano-scale 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 such 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 crystallizations or to direct construction of molecular electronic components into a memory (Robinson & Seeman, 1987).

The potential of self-assembly for fabricating molecular electronic circuits is particularly 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 will 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, where every atom's location 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., NAND 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.

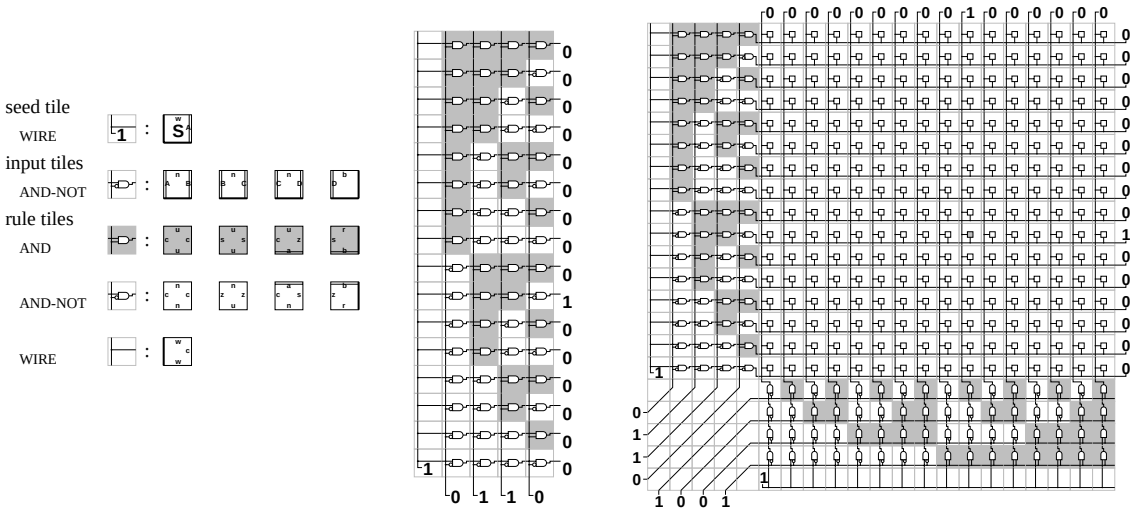

FIGURE 5 Using self-assembly of DNA tiles to create a molecular-scale pattern for a RAM memory with demultiplexed addressing. Labels indicate matching constraints. The tileset is closelly related to the binary counter. Adapted with permission from Cook, et al, 2003.

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 exist 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 are needed to access a RAM memory (shown in Figure 5), and for signal-processing primitives such as the Hadamard matrix transform (Cook et al, 2003). 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.

There are many technical hurdles that will have to be overcome before algorithmic self-assembly can develop into a practical commercial technology. It is not clear whether 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 by technologies that need to place molecular components in a precisely defined complex organization.

SUMMARY AND PROSPECTS

DNA-based self-assembly appears to be a robust and readily programmable phenomenon.

Periodic two-dimensional crystals have now 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. Unlike high-quality crystals, current DNA tile lattices are often slightly distorted, with the relative position of adjacent tiles jittered by a nanometer or more and with lattice defect rates of 1% or more. Some DNA tiles designed to form two-dimensional sheets appear to prefer to form 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 non-destructively on the substrate of choice.

Although 1D and 2D algorithmic self-assembly has been demonstrated, per-step error rates between 1 and 10% preclude execution of complex algorithms. Recent theoretical work has pointed to 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, buffers – will also be necessary.

Presuming algorithmic self-assembly of DNA can be made more reliable, it becomes important to understand the logical structure of self-assembly programs, and how it 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: what kinds of shapes and patterns can be assembled using a small number of tiles, and/or how quickly can they be assembled?

To what extent has this investigation enlightened us about how information and algorithms can be encoded in biochemical systems? It is interesting of itself 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 for energy minimization are unfamiliar factors for computer scientists to consider – but functional computation can be hidden in many places!

Thus, DNA self-assembly can be seen as one step in our quest to harness biochemistry in the same way we have harnessed the electron. Electronic computers are good at (and pervasive for) embedded control of macroscopic and microscopic electro-mechanical systems. We don't yet have embedded control for chemical and nano-scale systems. Programmable, algorithmic biochemical systems may be our best bet.

REFERENCES

Adleman, L.M. 1994. Molecular computation of solutions to combinatorial problems. Science 266: 1021—1024.

Cook, M., P.W.K. Rothemund, and E. Winfree. (2003, to appear). Self-assembled circuit patterns. DNA Based Computers 9, Lecture Notes in Computer Science, Springer-Verlag.

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

Lagoudakis, M.G., and T.H. LaBean. 2000. 2D DNA self-assembly for satisfiability. In DNA Based Computers V (eds. E. Winfree and D.K. Gifford). American Mathematical Society, Providence, RI.

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: 493—496.

Reif, J. 1997. Local parallel biomolecular computing. In DNA Based Computers III (eds. H. Rubin and D.H. Wood). American Mathematical Society, Providence, RI.

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. 1982. Nucleic-acid junctions and lattices. Journal of Theoretical Biology 99: 237—247.

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. In Mathematica Theory of Automata (ed. J. Fox). Polytechnic Press, New York.

Winfree, E. 1998a. Simulations of computing by self-assembly. Caltech CS Technical Report 1998.22.

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

Winfree, E., X. Yang, and N.C. Seeman. 1998c. Universal computation via self-assembly of DNA: some theory and experiments. In DNA Based Computers II (eds. L.F. Landweber and E.B. Baum). American Mathematical Society, Providence, RI.

Winfree, E. 2000. Algorithmic Self-Assembly of DNA: Theoretical Motivations and 2D Assembly Experiments. Journal of Biomolecular Structure and Dynamics 11: 263—270.