The challenge of programming molecules to manipulate themselves.

BY DAVID DOTY

Theory of Algorithmic Self-Assembly

SELF-ASSEMBLY IS THE process by which small components automatically assemble themselves into large, complex structures. Examples in nature abound: lipids self-assemble a cell's membrane, and bacteriophage virus proteins self-assemble a capsid that allows the virus to invade other bacteria. Even a phenomenon as simple as crystal formation is a process of self-assembly. How could such a process be described as "algorithmic?" The key word in the first sentence is *automatically*. Algorithms automate a series of simple computational tasks. Algorithmic self-assembly systems automate a series of simple growth tasks, in which the object being grown is simultaneously the machine controlling its own growth.

Although large tracts of the theory presented in this article are applicable to non-molecular systems, much of the motivation arises from nanoscale self-assembly (as an *engineering* field, as opposed to the study of natural self-assembly systems). The broad goal of nanoscience is to manipulate molecules with nanoscale precision. Ambitious long-term applications include microscopic, chemical-detecting robots that move toward and metabolize pollutants, or the integration of human tissue with an implanted medical device.

Why should computer science have anything to do with nanoscience, beyond the obvious role of developing software-modeling tools? Luca Cardelli, in a panel discussion at the 2011 Conference on DNA Computing and Molecular Programming, observed that while the computing revolution was about the *systematic manipulation of information*, nanoscience is about the *systematic manipulation of matter*. Nanoscience provides a novel justification for studying computation. Many of the traditional forms of manual control are simply not possible at small scales. Automating the growth of molecular structures is not merely faster or more convenient than building such structures by hand. Our hands, and the machines they operate, are simply too large to manipulate individual molecules. We must learn to program molecules to manipulate themselves.

DNA is the molecule of choice in many labs, not for its biological properties but for its information-bearing properties. It is easy to synthesize, and its physical properties are well understood. DNA origami[45] is currently the most successful laboratory technique for self-assembling DNA. A long scaffold DNA strand is folded into a shape by mixing it with hundreds of shorter staple strands, each of which binds to

» **key insights**

- The idea of "molecules that can perform computation" is transforming the way we engineer self-assembling molecular systems.

- A small number of simple types of molecules can grow into large, sophisticated nanoscale structures automatically.

- Understanding the fundamental abilities and limitations of these systems is crucial for guiding experimental work. Many known theoretical results have drawn on the theorems, techniques, and paradigms of computer science.

one region of the scaffold with its left half and another region of the scaffold with its right half, bringing the two regions together. A more experimentally challenging technique, DNA tile assembly, is the physical basis for the theoretical work discussed in this article. A DNA tile is a DNA complex with four short single-stranded "sticky" ends protruding from it (see Figure 2a), which are intended to bind the tile to other tiles having sticky ends that are Watson-Crick complementary. Figure 1 shows images of some patterns that have been experimentally assembled with DNA tiles. Figure 2 shows how a DNA tile is modeled as a square with four sides having "specific glues." DNA origami in its current incarnation is non-algorithmic self-assembly, whereas DNA tile assembly is potentially algorithmic. We now explain this distinction.

Turing[54] stated, "The 'computable' numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means." *Finite means* is formally equated with *algorithm* (for example, a Turing machine, a λ-expression, a Python program, or an appropriately initialized configuration of Conway's Game of Life). There are uncountably many real numbers but only a countable number of algorithms, so most real numbers are not computable. What if we wish to make sense of the question, which *integers* are computable? By Turing's qualitative definition, they are all computable, but it is reasonable to object that some integers (for example, $n = 10^{10,000}$) are easier to compute than others (for example, m = a random sequence of 10,000 digits), in the sense that a much smaller program suffices to compute n than to compute m. To make formal sense of such intuition, the theory of Kolmogorov complexity[33] gives a rigorous *quantitative* measure of the "algorithmic complexity" of an integer (or any other "finite object" such as a finite string or a graph): the length in bits of the shortest algorithm that prints the integer and halts.

Both of these definitions adopt the view that an object is "algorithmically constructed" when there is an algorithm constructing the object that is much smaller than the object itself (infinitely smaller in the former case). From this perspective, *algorithmic self-assembly* describes the self-assembly of a structure whose total number of components (its "size") is much greater than the number of different *types* of components. Since a component type is reused many times, it has no way of "knowing" where it is going to end up in the structure, so only local information is available to guide its attachment. DNA origami therefore does not constitute algorithmic self-assembly, since each staple strand "hardcodes" its position in the final structure. DNA tile assembly, however, has the potential to reuse a small number of tile types to create large structures.

Our combinatorial model of the dynamic behavior of these tiles is called the abstract Tile Assembly Model (aTAM), due to Winfree,[56] explained briefly in Figure 2. Figure 2c shows seven tile types that fill the entire second quadrant with a painting of the discrete Sierpinski triangle, showing that this pattern is "algorithmically self-assemblable." The main computation is done by the bottom four rule tile types using *cooperative binding*, which refers to the fact that a tile with only strength-1 glues cannot bind to an assembly unless at least two of them match. (Sometimes this binding strength threshold is called the "temperature" τ, which is usually 2 but not always.) The rule tiles in this example always bind using their south and east glues. Thus the glue labels (bits in this case) can be imagined as inputs to a function computed by the rule tile types (analogous to a transition function in a Turing machine or cellular automaton). The output of the function in this example is the XOR of the bits, which is advertised on the north and west sides.

Parallelism in Molecular Computing: The Bad News

Algorithmic self-assembly is a subfield of molecular computing, highlighted in a news article by Kirk. L. Kroeker in the December 2011 issue of *Communications*. The field was initiated in 1994 when Adleman,[1] in a landmark proof-of-concept experiment, designed DNA molecules that interact to solve a 7-vertex case of the Hamiltonian path problem: executing a "DNA algorithm" whose basic operations are well-understood chemical interactions such as hybridization.

Let us state an unequivocal limitation to the power of molecular computing as a model of computation, which applies to algorithmic self-assembly systems as well. Molecular computing is not a magical potion that can be ladled over *NP*-complete problems to transubstantiate them into tractable problems. In its most generic

Figure 1. Experiments with double-crossover tiles.
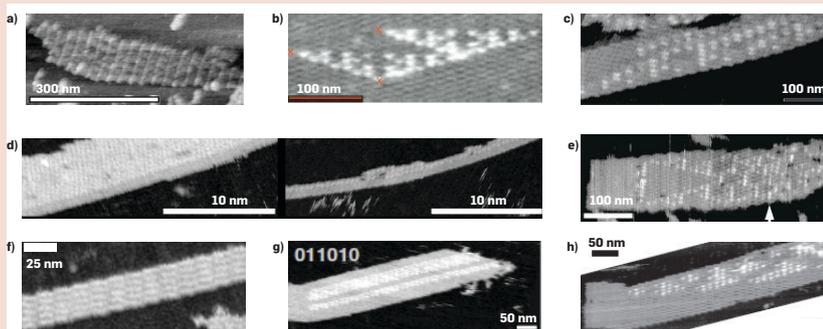
Atomic force microscopy measures the height of a structure on a surface (mica). Some tile types may have attached hairpins as "labels" (appearing white in images).
a) Lattice of tiles (single tile type).[59]
b) Sierpinski triangle in a lattice, no proofreading.[46]
c) Binary counter in a ribbon, no proofreading.[6]
d) Standard (left) versus snaked (right) proofreading for suppression of facet errors.[17]
e) Sierpinski triangle in a ribbon, no proofreading.[28]
f) Ribbons with zig-zag tiles for suppression of spurious nucleation.[49]
g) Ribbons nucleating from origami seed, copying a bit string, with proofreading.[7]
h) Ribbons nucleating from origami seed, executing a binary counter, with partial proofreading.[7]

form, the sorcery of DNA solutions to *NP*-complete problems proceeds by letting the DNA test all possible solutions in parallel in a single test tube. Assuming a modest solution length of 300 bits, squeezing 2^{300} molecules into a test tube is like installing 2^{300} processors in a parallel computer: there is not enough sand in the world to supply that much silicon.
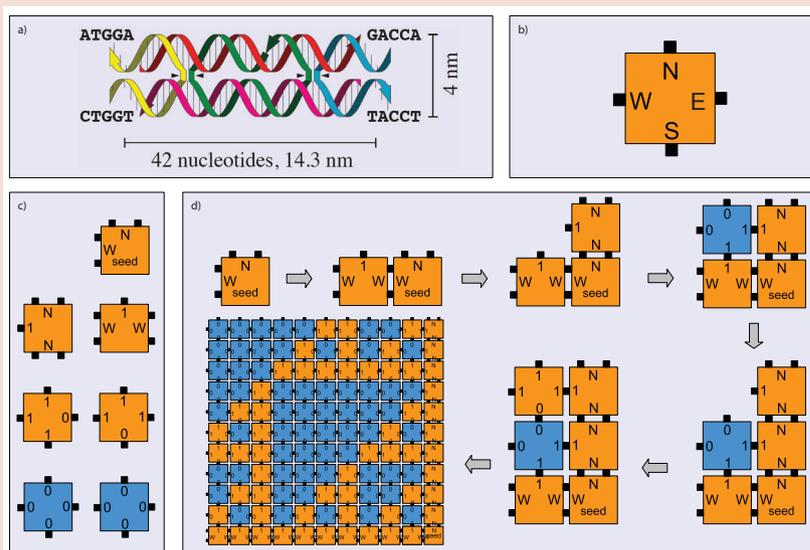
Scientific journals continue to traffic in tales of DNA's exponential search capability. As with Mjölnir, the mountain-crushing hammer of Thor, the time has come to place these tales into the dustbin of mythology, making room for the real—and far more exciting—scientific work that remains to be done in molecular computing. If there is a breakthrough in molecular engineering that will enable the solution of classically intractable problems, it will have to come from quantum mechanics or perhaps from some exotic physical theory yet to be discovered. It will not come about solely by replacing silicon with DNA. But enough pessimism; let's find out what DNA *can* do.

Computational Universality

Winfree[56] showed that the aTAM is computationally universal, that is, able to simulate any algorithm. This one fact explains the richness of the theory presented in the rest of the article. What exactly does it mean?

The model as stated lacks one feature in common with other computational models: there is no input! One potential way to program a single tile set with different inputs that result in different behaviors is to generalize the idea of a single seed tile type to a larger seed assembly. Within the model, we allow the seed assembly to be any finite stable assembly ("stable"= all cuts of the assembly that separate it into two components must break bonds of total strength at least 2). In practice, seed assemblies are not necessarily made of tiles, but must simply have a perimeter compatible with the tiles, such as a DNA origami shape appropriately augmented with sticky ends on its side (Figure 1g–h). We use the term *tile system* to refer to a finite set of tile types, together with other parameters needed to determine its behavior, such as its seed assembly σ and its temperature τ.

With this convention established, we can now state more formally what it means to claim that the aTAM is computationally universal. For every single-tape Turing machine M, there is a tile set T so that, for every input string x, there is a seed assembly σ of T so that T with seed σ uniquely assembles a space-time configuration transcript of M on input x.[a] By "uniquely assembles," we mean that although T with seed σ assembles many different partial assemblies, it has a unique terminal assembly α (terminal = no tile can attach to it). The tth row of α represents the configuration of $M(x)$ at time step t, with σ occupying the first row.

The computational universality of the aTAM implies that arbitrary algorithms may be executed by self-assembling tiles and therefore used to direct the growth of the tiles. However, the aTAM is not just another programming language. The subtle interplay of computation and geometry in self-assembly gives rise to unique characteristics such as the distinction between computable patterns and self-assembling patterns discussed later.

Modeling Errors

The aTAM is not a realistic model of how DNA tiles actually behave at the molecular level. In particular, it makes two assumptions known not to hold in practice: that tiles never detach from an assembly, and that tiles only attach when their binding strength exceeds the temperature threshold value τ. Winfree[56] introduced the kinetic Tile Assembly Model (kTAM) as a more realistic model of tile assembly that uses standard laws of chemical kinetics to relax these assumptions. Each tile type is assumed to attach to any binding site on an assembly α, producing a new assembly β, at a rate proportional to its concentration, regardless of its strength of attachment. If we assume that there is only a single seed tile, and that each other tile type is equally concentrated (and much higher count than 1, so that their rate of depletion is essentially 0), then this forward rate is equal for all tile types and approximately constant over time. Call this forward rate r_f. Each tile within β is assumed to detach at a rate proportional

---

Figure 2. Abstract Tile Assembly Model (aTAM).

a) Double-crossover tile with four sticky end.
b) Representation of a tile as a square with sides labeled by string "glues."
c) Seven tile types. Bond strengths indicated by the number of small black squares on a side: total strength 2 is required to attach a tile to a partially formed assembly of tiles. One tile type is designated as the seed, from which growth is assumed to nucleate.
d) Growth of the tiles into an assembly with the discrete Sierpinski triangle pattern.

---

a   σ is a $1 \times |x|$ row encoding x in a standard way; we are not cheating by embedding the entire computation $M(x)$ into σ.

to e^{-b}, where $b \in \mathbb{N}$ is the total strength with which the tile is bound. Call this reverse rate $r_{r,b}$. The constant of proportionality depends on factors such as temperature, salinity, and sticky end length. Winfree[56] showed that by setting these factors and the non-seed tile concentrations such that $r_f = r_{r,2} + \epsilon$ then the probability that the seed grows into a correct assembly (an assembly producible under the aTAM growth rules) approaches 1 as $\epsilon \to 0$. However, the closer ϵ is to 0, the slower assembly proceeds, with $\epsilon = 0$ representing an unbiased random walk with attachment equally likely as strength 2 detachment.

There is only so much that one can do in a laboratory to precisely control experimental conditions; errors are bound to happen, just as deep-space communication and large-scale data storage inevitably corrupt some of the bits they intend to send or store. What should we do about this? Why not do the same thing that NASA and Google do about it: build in redundancy to help correct the errors! Here, I discuss algorithmic error-correction in the kTAM, showing that under certain conditions, the probability of correct growth can be boosted to enable correct growth under more flexible experimental conditions.

Winfree and Bekbolatov[58] devised a scheme known as proofreading. It helps correct *growth errors*, shown in Figure 3a, in which an incorrect tile binds with strength 1 where a correct tile could bind with strength 2. A tile bound with only strength 1 may temporarily stick and be secured in place by a subsequent tile attachment. The basic proofreading scheme is shown in Figure 3b. Each tile type t is replaced by a $k \times k$ block of tile types, with glues internal to the block unique to t. This enforces that errors cannot happen in isolation: if there are any errors in the block, then there are $\geq k$ errors. Intuitively, errors happen only slowly (since insufficiently bound tiles fall off more quickly than new tiles attach), so if one error occurs, it is more likely for the erroneous tiles to detach before the block completes than for an additional $k-1$ errors to occur and secure the block in place. To a very rough approximation, $k \times k$ proofreading changes a base error probability of ϵ to be ϵ^k. This scheme incurs two costs: the number of tile types increases by factor k^2, and the system suffers a resolution loss since $k \times k$ blocks in the new system represent individual tiles in the original system. Reif, Sahu, and Yin[44] demonstrated a compact proofreading scheme, in which there is no resolution loss, but in general the tile complexity blowup is exponential. To reduce error probability of ϵ to ϵ^k for tile system T, the new tile system may have up to $|T|^{k^2}$ tile types. Soloveichik and Winfree[51] showed that for all but a restricted class of tile systems, the tile complexity blowup of any compact proofreading scheme is necessarily exponential.

A *facet error* is the attachment of a tile with strength 1 where no tile should go because there is only one adjacent strength 1 glue. Both the proofreading constructions cited[44,58] correct growth errors but not facet errors. In practice, facet errors tend to dominate the overall behavior. Chen and Goel[14] showed a proofreading scheme, also using a $k \times k$ block replacement as in Winfree-Bekbolatov,[58] which is resistant to both growth and facet errors, and which achieves error of $O(\epsilon^k)$ on a base error probability of ϵ. In fact, this scheme was experimentally demonstrated[17] to reduce facet errors better than either no error-correction or the Winfree-Bekbolatov proofreading scheme. Chen, Goel, and Luhrs[15] showed that two-dimensional tile systems may be proofread with no resolution loss and only polynomial increase in tile complexity by using the third dimension. It grows a larger structure than the original system, but when projected onto the plane there is no resolution loss. They also introduce combinatorial criteria to help in proving that a proofreading scheme works without needing to carry out the cumbersome Markov process analysis required to analyze kTAM systems.

Doty et al.[26] showed a stronger form of error correction using the hierarchical model of self-assembly, in which there is no seed and two assemblies that have assembled independently in parallel are allowed to aggregate together, as opposed to the standard seeded aTAM in which tiles attach one at a time to a growing assembly. In this model, they use the ability of geometry to enforce binding constraints (preventing two assemblies with matching glues from attaching if their shapes are not compatible to allow the glues to touch) to show how to assemble an $n \times n$ square from $O(\log n)$ tile types while guaranteeing the following constraint. Arbitrary strength 1 growth is allowed; however, any assembly that grows sufficiently to become stable at temperature 2 is guaranteed to assemble into the correct final assembly. Thus errors are prevented absolutely, rather than only with high probability.

The errors modeled so far are those of "insufficient attachment:" a tile binds with strength 1 and sticks around long enough to cause prob-

Figure 3. a) Growth error. b) 2 × 2 Proofreading. Each tile type t is replaced by a $k \times k$ block of tile types, with glues internal to the block unique to t.
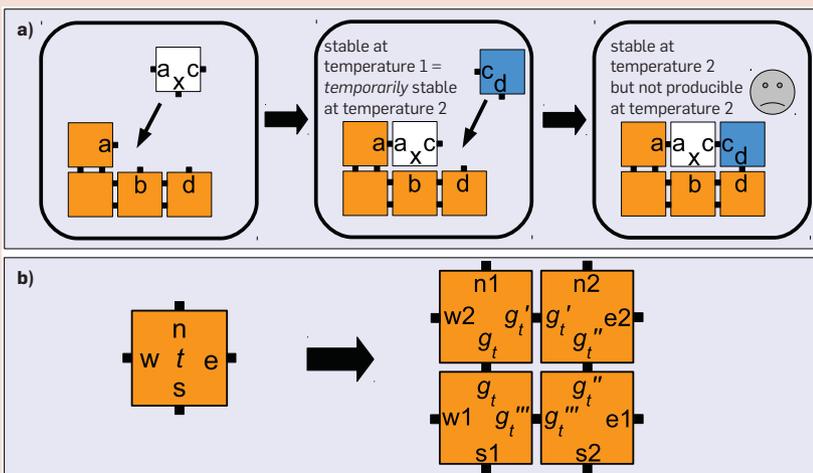
lems. We can consider other types of errors. For instance, we consider some tile types to have input and output sides, although this is not enforced from within the model. Consider the following error: after some time, a portion of the assembly is removed to make a hole. This could result in "backward" growth filling in the hole using output glues, possibly resulting in the wrong tiles attaching (this would happen in Figure 2d since XOR is not a 1-1 function). Winfree[57] introduced a 3×3 block transformation scheme called self-healing that enables the assembly to regrow correctly even when holes are blown out of the assembly, so long as the seed remains present. Soloveichik, Cook, and Winfree[50] showed how to combine self-healing simultaneously with proofreading. Chen et al.[16] showed that a more extreme form of self-healing was possible for the particular task of assembling an $n \times n$ square from $O(\log n)$ tile types: the entire square is able to grow from any subassembly of width or height $2 \log n$, without requiring the seed. They also generalize the idea to arbitrary finite shapes (with some resolution loss).

Another type of error is *spurious nucleation*, in which tiles attach to each other without a seed, using strength 1 growth to form a stable assembly from which an incorrect assembly could grow. Schulman and Winfree showed both theoretically[48] and experimentally[49] that a certain class of tile systems can be made resistant to nucleation errors, in the sense that with the addition of $O(k)$ extra tile types, in the absence of the seed, the probability is at most $O(2^{-k})$ that sufficiently many tiles aggregate to create a stable assembly from which further growth can occur.

The conclusion to draw from these error correction techniques is that, even though the aTAM does not accurately describe the behavior of DNA tiles, under certain assumptions, it is an implementable "programming language" for tile self-assembly.

Tile Complexity
We have identified the algorithmic aspect of self-assembly with the ability of a small number of tile types to assemble a large structure. This ability will prove crucial to programming large,

> Even though the aTAM does not accurately describe the behavior of DNA tiles, under certain assumptions, it is an implementable "programming language" for tile self-assembly.

complex molecular self-assembling systems, since the total number of types of molecular components dominates the cost (in money and time) of implementation.

To quantitatively formalize this idea, Rothemund and Winfree[47] defined the *tile complexity* of a shape S (a finite, connected subset of $\mathbb{Z}^2$) to be the minimum number of tile types in any tile system—with a single seed tile—that uniquely assembles a single terminal assembly with shape S. The requirement of a size-1 seed avoids cheating by simply letting the seed assembly have the desired shape. They studied the tile complexity of $n \times n$ squares, which has since become a canonical benchmark problem for studying various other aspects of self-assembly, since $n \times n$ squares are in a sense the simplest shape with non-trivial tile complexity. The simplest shape is a single point, whose tile complexity is clearly 1. The next simplest shape might be a $1 \times n$ line. An easy argument shows that its tile complexity is n. Since it is one-dimensional, any stable assembly with that shape must use entirely double-strength glues to hold it together. If fewer than n tile types are used, then one must repeat, but this tile system would then be able to grow an infinite line by repeating the segment between the repetitions. Only in two dimensions is tile complexity nontrivial.

Rothemund and Winfree showed that for most values of n (all algorithmically random n), the tile complexity of an $n \times n$ square is $\Omega(\frac{\log n}{\log \log n})$. Why does this hold? A tile system of size k can be described using $O(k \log k)$ bits ($O(\log k)$ bits per tile type). If the tile system uniquely self-assembles an $n \times n$ square, then that description, together with a constant size aTAM simulator, constitute a program of length $O(k \log k)$ that outputs n. Since for most n, the shortest program outputting n is $\log n$ bits long, we must have $k \log k = \Omega(\log n)$, that is, $k = \Omega(\frac{\log n}{\log \log n})$. They then show that the tile complexity of all $n \times n$ squares is $O(\log n)$, in which $\log n$ tile types each represent a particular bit of n that assemble to form a $1 \times \log n$ row whose north glues represent n in binary. From this assembly, a $O(1)$ tile types attach to assemble an $n \times n$ square, first by growing a counter—essentially a binary-to-unary converter—that counts

from n down to 0, extending the $1 \times \log n$ row into a $n \times \log n$ rectangle, together with $O(1)$ tile types that extend this rectangle into the full square.

The gap between these upper and lower bounds arises from the fact that encoding one bit of n per tile type is wasteful, in an information-theoretic sense. In principle, a set of size k can encode $\log k$ bits per element. Letting $k = \frac{\log n}{\log \log n}$, Adleman et al.[2] showed how to encode $\approx \log k$ bits of n per tile type, showing the tile complexity of any $n \times n$ square is $O(\frac{\log n}{\log \log n})$. The trick is to encode n in a larger base b. Letting b be the unique power of two satisfying $k \le b < 2k$, k tile types are required to encode n in base b. An additional $O(k)$ tile types then convert n to base 2, from which the constant set of tile types of Rothemund and Winfree[47] self-assemble an $n \times n$ square.

Intuitively, most of the complexity of an $n \times n$ square is captured by its width; geometry does not play much of a role. What about shapes with more complicated geometry? One would expect that even if such a shape has a compact algorithmic description, self-assembly that simulates this algorithm may not be possible to execute *within* the shape. Soloveichik and Winfree[52] showed that in fact, if we ignore scaling factors, then the tile complexity of *every* shape S is closely related to its Kolmogorov complexity $K(S)$, the length in bits of the shortest program outputting the points in the shape. More formally, given a finite shape $S \subset \mathbb{Z}^2$ and positive integer $c \in \mathbb{Z}^+$, let $S^c = \{ (x,y) \mid (\lfloor \frac{x}{c} \rfloor, \lfloor \frac{y}{c} \rfloor) \in S \}$ be S scaled up by factor c. They showed that for every finite shape S, there is a scaling factor c such that the tile complexity of S^c is $\Theta(\frac{K(S)}{\log K(S)})$. This is a tight bound, since no smaller tile system could uniquely self-assemble any scaling of S without contradicting the Kolmogorov complexity of S. The scaling factor c is proportional to the running time of the shortest program for S. A constant set of tile types T produces S^c from a $1 \times \frac{K(S)}{\log K(S)}$ row encoding the program for S. Thus, T is a single tile set that is universally programmable (by seeding with an appropriate program) for building any finite shape S, if scaling factors are ignored.

What is the lesson behind of all these bounds? The ability of tiles to execute algorithms translates directly into

The ability of tiles to execute algorithms translates directly into the ability to assemble large, complex shapes from a very small number of different types of molecular components.

the ability to assemble large, complex shapes from a very small number of different types of molecular components.

Tile complexity has also been studied in many interesting variants of the standard aTAM. Each of these models allows a *single* tile system to be reused for assembling different structures by programming it with different environmental conditions affecting the behavior of the tiles, thus showing that $O(1)$ tile complexity suffices for assembly of complex structures if additional laboratory steps are used.

In the *step assembly* model introduced by Reif,[45] tile types are added to a test tube in a series of steps, with each step assumed to run to completion before washing away remaining unbound tiles and nonterminal assemblies, before adding new tile types for the next step. Maňuch, Stacho, and Stoll[37] showed that for a large class of shapes, including arbitrary shapes scaled by factor 2 or any other shape with a Hamiltonian path, $O(1)$ tile types can assemble the shape in the step assembly model. There is a natural generalization of step assembly known as staged assembly, in which the order of test tube mixing is not monotonic and hierarchical assembly (attachment of two large assemblies to each other) is allowed. A directed graph describes the order of test tube mixing. Under this model, Demaine et al.[20] showed that $O(1)$ tile types can be used to assemble any shape, and that certain classes of shapes require only $O(\log n)$ parallel mixing stages (source-to-sink distance in the mixing graph). They also showed a number of trade-offs between this measure (stage complexity) and bin complexity, the total number of test tubes required. In the same model in one dimension, Demaine et al.[21] showed that for each alphabet Σ, there is a constant set of tile types—each labeled by a symbol from Σ —so that, given any string $x \in \Sigma^*$, the tiles can be mixed to uniquely assemble a linear assembly spelling x. Furthermore, the number of mixing stages required is within a constant of the size of the smallest context-free grammar that produces the singleton language $\{x\}$, if each intermediate stage is required to produce a unique terminal assembly (if not, then there are strings for which more efficient mixings exist).

Another model relaxes the assumption of constant temperature over time: *temperature programming*, in which the temperature is raised and lowered over many stages, with each stage assumed to reach a terminal state before changing the temperature for the next stage. Aggarwal et al.[5] proved that with one temperature change, a *thin rectangle*—an $n \times k$ rectangle where $k < \frac{\log n}{\log \log n - \log \log \log n}$—can be assembled from $O(\frac{\log n}{\log \log n})$ tile types, compared to a provable lower bound of $\Omega(\frac{n^{1/k}}{k})$ in the standard aTAM. Kao and Schweller[31] showed that $O(1)$ tile types suffice to assemble any $n \times n$ square using $O(\log n)$ temperature changes. Summers[53] showed that $O(1)$ tile types suffices to assemble any shape (scaled up) using temperature programming, using two constructions. In the first, the number of temperature changes is $O(|S|)$, and the scaling factor is $O(1)$. In the second, the number of temperature changes is $O(K(S))$ (the Kolmogorov complexity of S), and the scaling factor is proportional to the running time of the shortest program for S.

Randomized self-assembly uses the inherent nondeterminism in the aTAM in which multiple tile types compete to bind to a single binding site, and tile type concentrations determine the probability of a tile type attaching to a binding site when it competes with others sharing the same input glues. Chandran, Gopalkrishnan, and Reif[12] showed that the tile complexity of a $1 \times n$ line is $\Theta(\log n)$ in the randomized model, that is, $\Theta(\log n)$ tile types are sufficient and necessary to assemble a line of expected length n (compared to n tile types that are provably necessary in the deterministic aTAM). In fact, their tile system is equimolar: all tile types have equal concentrations. Becker, Rapaport, and Rémila[9] introduced the model of *concentration programming*, in which concentrations of tile types are used to program input to the system. Kao and Schweller[32] showed that for each $\delta, \varepsilon > 0$, there is a single tile system T so that, for every $n \in \mathbb{Z}^+$, there is a setting of concentrations that cause T to assemble an $n' \times n'$ square, where $(1 - \varepsilon) n \leq n' \leq (1 + \varepsilon)n$ with probability at least $1 - \delta$. That is, the square is probably approximately assembled. Doty[23] improved this result, showing that for each $\delta > 0$, a single tile system can be used to assemble an exactly $n \times n$ square for any $n \in \mathbb{Z}^+$ with probability at least $1 - \delta$ using concentration programming.

Demaine et al.[22] studied the RNase model (first suggested by Rothemund and Winfree[47]), in which some tile types are assumed to be made of RNA, which is compatible to bind with DNA but has a different chemical structure. An enzyme called ribonuclease—abbreviated RNase—can dissolve RNA but not DNA, so adding RNase after assembly completes allows one to controllably separate the assembly into parts that can now interact with each other. They use this model to improve on the Soloveichik/Winfree[52] result, showing that for every shape S, the optimal $O(\frac{K(S)}{\log K(S)})$ tile types suffice to assemble a scaled version S^c of S, but with scaling factor $c = O(\log |S|)$, rather than c depending on the running time of S's shortest program.

In the *flexible glue* model studied by Aggarwal et al.,[5] unequal glues are allowed to interact with non-zero strength. One could imagine this being implemented, for instance, by multiple sticky ends on a tile's side, in which the strength of interaction depends on the number of sticky ends that match. They showed that with flexible glues, $O(\sqrt{(\log n)})$ tile types are necessary and sufficient to assemble an $n \times n$ square.

Computational Complexity
Rather than manually analyzing each new shape or class of shapes to determine its tile complexity, it would be beneficial to develop an algorithm that automates the task: given a finite shape S, it outputs the smallest tile system that uniquely assembles S. Unfortunately, an efficient algorithm is unlikely to exist: Adleman et al.[3] showed that the problem is *NP*-complete. Here, "uniquely assembles S" means, assembles one unique terminal assembly, whose shape is S. One could object that a tile system deterministically assembles S if it produces many terminal assemblies that all have the shape S. Under this definition, Bryans et al.[10] showed that the problem, in a sense, is even harder: NP^{NP}-complete, that is, *NP*-complete for algorithms that have access to a constant-time subroutine for an *NP*-complete problem such as SAT.

In addition to optimization prob-

lems, there are important verification problems of interest in tile assembly. One easy problem is this: given a tile system T and a (possibly nonterminal) assembly α, can T produce α? One can simply add tiles to the seed that are consistent with α and have sufficient binding strength until either α is complete, or until no more tiles can be added; the answer is "yes" in the former case and "no" in the latter. A more challenging question: Is α the unique terminal assembly produced by T? A tile system could produce α through an exponential number of different pathways (orders of tile addition), so it would naïvely seem to require that we check all of them to ensure none of them create a different terminal assembly. However, Adleman et al.[3] showed this problem is solvable in quadratic time. On the other hand, in the hierarchical aTAM (in which two large assemblies are allowed to attach), the problem of determining whether α is uniquely assembled is *coNP*-complete.[11] This is discouraging, since such decision problems are a formalization of the practically important task of "write a simulator for the hierarchical aTAM."

Shape building is one goal of self-assembly; another is pattern painting. Briefly, we paint some tile types black (for example, by attaching a streptavidin marker molecule) and say the pattern assembled is the set of positions in the final assembly with a black tile. Such a definition is appropriate for modeling practical goals such as self-assembled circuit layouts, such as those studied by Cook, Winfree, and Rothemund.[19] Although the proof techniques for these verification problems generalize easily to pattern assembly, it remains open to prove or disprove pattern assembly analogues of the optimization results noted here. The optimization problems are uncomputable if the tile system is allowed to grow arbitrarily far outside the pattern, using the same argument that shows Kolmogorov complexity is uncomputable. If the minimum tile system to assemble a given finite pattern could be computed, then this computation could be simulated in the second quadrant by a tile system of size $O(1) + \log n$ searching for the first pattern on the positive x-axis that requires more than n tile

types to assemble, at which point the tile system could grow back to the x-axis assemble that pattern, a contradiction. So to establish (for instance) *NP*-completeness requires also specifying a bounding box as part of the input and requiring that the assembled shape is a subset of the bounding box. Göös and Orponen[30] and Lempiäinen, Czeizler, and Orponen[35] developed branch-and-bound algorithms for a variant of this problem, but these algorithms require exponential time.

One can also consider the assembly of infinite patterns (as in Figure 2d) as the "computability-theoretic" version of self-assembly. Lathrop et al.[34] showed that not every computable pattern (subset of $\mathbb{Z}^2$) can be self-assembled, since each time step of computation with tiles irreversibly occupies a point in space, and for some patterns, it requires strictly more time to compute whether that point is part of the pattern. However, with regard to one-dimensional patterns, Patitz and Summers[42] showed that every computable subset of $\mathbb{N}$ can be self-assembled on the x-axis (crucially using the fact that the assembly occupies a large area outside the x-axis where the actual computation happens). Lathrop et al.[34] showed a weaker result for computably enumerable sets: there is a simple quadratically bounded function $f: \mathbb{N} \to \mathbb{N}$ such that, for every computably enumerable $A \subseteq \mathbb{N}$, $f(A)$ can be self-assembled on the x-axis.

Assembly Time Complexity
Another resource bound to consider is the time required to build a shape (as determined by stochastic chemical kinetics). There is a linear amount of parallelism available for assembling a shape of size N in fewer than N steps. This is because the frontier (set of potential binding sites) of an assembly of size N is potentially as large as $O(N)$, and in a given unit of time, a constant fraction of the frontier will have tiles attach if tile types are sufficiently concentrated. Adleman et al.[2] showed that any shape of diameter D requires time $\Omega(D)$ to assemble from a deterministic tile system in the aTAM, later shown by Chen and Doty[13] to hold for nondeterministic systems. Adleman et al.[2] showed that this lower bound is tight for the assembly of $n \times n$ squares, that is, they can be assem-

bled in time $O(n)$, achievable using the information-theoretically optimal $O(\log n / \log \log n)$ tile types. Reif[43] also studied local parallelism within the seeded aTAM, for the purpose of computation rather than shape-building. Reif showed that for many problems such as prefix sum and bitonic merge, it is possible to design a tile system that solves the problem using the optimal parallel speedup. Reif used a generalization of the model known as step assembly, in which tile types are added to a test tube in a series of steps, with each step assumed to run to completion before washing away remaining unbound tiles, before adding new tile types.

In a variant of the model known as the hierarchical aTAM, two assemblies that have assembled independently in parallel are allowed to aggregate together, as opposed to the standard seeded aTAM in which tiles attach one at a time to a growing assembly. The distributed parallelism of the hierarchical aTAM dwarfs the linear local parallelism of the seeded aTAM, so a natural conjecture is that hierarchical assembly could be *much* faster than seeded assembly. Chen and Doty[13] showed there is a hierarchical tile system with the optimal $O(\frac{\log n}{\log \log n})$ tile types that assembles an $n \times n$ square using $O(\log^2 n)$ parallel stages, which is close to the optimal $\log n$ stages, forming the final $n \times n$ square from four $n/2 \times n/2$ squares, which are themselves recursively formed from $n/4 \times n/4$ squares, and so on. However, despite this nearly maximal parallelism, the system actually requires superlinear time to assemble the square. They extended the definition of *partial order tile systems* studied by Adleman et al. in a natural way to hierarchical assembly and show that no hierarchical partial order tile system can build any shape with diameter N in less than time $\Omega(N)$. Intuitively, although attaching an assembly β of size k to another assembly α increases the size of α by k tiles in one step, a conservation of mass argument shows that the concentration of β is at most $1/k$, so we expect to wait at least k time steps for this attachment to occur. However, Chen and Doty also showed the partial order hypothesis to be necessary, since for infinitely many n, a tile

system can assemble a diameter-$\Theta(n)$ rectangle in time $O(n^{4/5} \log n)$, breaking the linear-time lower bound that applies to all seeded systems and partial order hierarchical systems. It is an open question to determine precisely the extent to which hierarchical assembly imparts a speedup over single-tile assembly.

Intrinsic Universality
Doty et al.[24] showed a different notion of universality, with respect to *growth* rather than computation, constructing a single tile set U that simulates any tile system T, by choosing an appropriate seed assembly to encode T using tiles from U. The simulation is intrinsic in the sense that the self-assembly process carried out by U is exactly that carried out by T, with each tile of T represented by an $m \times m$ block of U tiles. Therefore there is a single tile set that can (modulo rescaling) carry out any self-assembly process that can be achieved by any tile system. The challenge is to program all this activity without ever blocking a path that may later be needed for intra-block communication, since no block representing a tile is necessarily aware of the identities of its neighboring blocks, whether they are even present, and if not, whether they will ever arrive.

Power of Cooperative Binding
Cooperative binding is used crucially in all non-trivial constructions in the aTAM. It is also the most challenging aspect of the model to enforce experimentally, so it is reasonable to question its necessity in algorithmic self-assembly. Doty, Patitz, and Summers[27] investigated the power of temperature 1 self-assembly in the aTAM, in which all positive glue strengths have sufficient energy to bind a tile, which effectively inhibits the use of cooperative binding. They show that a wide class of deterministic tile systems, satisfying a condition known as *pumpability*, are incapable of universal computation in the sense that they self-assemble only "periodic" (semilinear) sets. They conjecture that all deterministic, temperature 1 tile systems are pumpable. Maňuch, Stacho, and Stoll[38] obtained similar results studying temperature 1 tile systems that assemble finite assemblies and

guarantee no glue mismatches, showing that $\Omega(n)$ tile types are required to assemble shapes of diameter n.

However, with slight changes to the model, universal computation becomes possible. Adleman et al.[4] devised a nondeterministic temperature 1 tile system that simulates a Turing machine in the following sense: the system has many terminal assemblies, but the unique largest terminal assembly represents the computation. Cook, Fu, and Schweller[18] showed that deterministic tile systems can achieve universal computation if allowed to grow in three dimensions. Patitz, Schweller, and Summers[41] showed that if negative glue strengths are allowed then universal computation is possible, essentially using negative-strength glues to enforce cooperative binding.

But the original question stands: what is the computational power of deterministic, planar, positive-strength, temperature 1 self-assembly? Is cooperative binding truly necessary to compute by planar self-assembly?

The Future

In the early history of computability theory, the optimistic term "elementary" was applied to problems computable in time bounded by a tower of exponentials such as $2^{2^{2^n}}$. It was not until the 1960s, once faster and faster computers had been built to execute actual algorithms, that polynomial time came to be seen as a more realistic notion of feasibly implementable. Today we confront a similar situation in algorithmic self-assembly, where experimental work has not yet caught up to theory. When it does, the theory will doubtlessly take unanticipated turns. One concrete way that theory can help experiment today is to further develop error correction, such as reducing tile complexity of error-correction constructions or handling broader classes of tile systems than previous constructions.

The complexity of today's operating systems and other software is made possible only by the development of software engineering techniques to help manage the complexity. Eventually molecular self-assembly systems will scale to the point that no one person could manually keep track of all the types of molecular compo-

> **Eventually molecular self-assembly systems will scale to the point that no one person could manually keep track of all the types of molecular components and how they are all supposed to fit together.**

nents and how they are all supposed to fit together. Software simulators for the aTAM exist[40,55] but require the user to think at the level of individual tile types, which quickly begins to feel like assembly code programming with a nightmarish instruction set. There is some preliminary work[8,25] developing higher-level languages for describing aTAM tile systems, but we are far from having a "Python of self-assembly."

DNA tile assembly is an example of passive self-assembly, in which the tiles are mobile but otherwise stateless. Active self-assembly, using components with a changeable state, may enable sophisticated structures to be built at a lower cost in terms of time, tile complexity, or error-resistance. Majumder, LaBean, and Reif,[36] as well as Fujibayashi et al.,[29] and Padilla, Liu, and Seeman,[39] have proposed theoretical suggestions to augment tiles with a physical mechanism known as strand displacement to enable active signaling across tiles. It remains to see what will actually work in the laboratory, but whatever works will provide a rich theory to explore. As in any nascent field, the most exciting aspect is not knowing what the future holds.

References
1. Adleman, L.M. Molecular computation of solutions to combinatorial problems. *Science 266*, 5187 (1994), 1021.
2. Adleman, L.M., Cheng, Q., Goel, A. and Huang, M-D. Running time and program size for self-assembled squares. In *Proceedings of STOC* (2001), 40–748.
3. Adleman, L.M., Cheng, Q., Goel, A. and Huang, M-D., Kempe, D., de Espanés, P.M. and Rothemund, P.W.K. Combinatorial optimization problems in self-assembly. In *Proceedings of STOC* (2002), 23–32.
4. Adleman, L.M, Kari, J., Kari, L., Reishus, D. and Sosík, P. The undecidability of the infinite ribbon problem: Implications for computing by self-assembly. *SIAM Journal on Computing 38*, 6 (2009), 2356–2381. Preliminary version appeared in FOCS 2002.
5. Aggarwal, G. Cheng, Q., Goldwasser, M.H., Kao, M-Y., de Espanés, P.M. and Schweller, R.T. Complexities for generalized models of self-assembly. *SIAM Journal on Computing 34* (2005), 1493–1515. Preliminary version appeared in In *Proceedings of SODA* 2004.
6. Barish, R.D., Rothemund, P.W.K. and Winfree, E. Two computational primitives for algorithmic self-assembly: Copying and counting. *Nano Letters 5* (2005), 2586–2592.
7. Barish, R.D, Schulman, R., Rothemund, P.W.K. and Winfree, E. An information-bearing seed for nucleating

algorithmic self-assembly. In *Proceedings of the National Academy of Sciences 106*, 15 (Mar. 2009), 6054–6059.
8. Becker, F. Pictures worth a thousand tiles, a geometrical programming language for self-assembly. *Theoretical Computer Science 410*, 16 (2009), 1495–1515.
9. Becker, F., Rapaport, I. and Rémila, E. Self-assembling classes of shapes with a minimum number of tiles, and in optimal time. In *Proceedings of FSTTCS* (2006), 45–56.
10. Bryans, N., Chiniforooshan, E., Doty, D., Kari, L. and Seki, S. The power of nondeterminism in self-assembly. *Theory of Computing.* To appear. Preliminary version appeared in *Proceedings of SODA* (2011), 590–602.
11. Cannon, S., Demaine, E.D., Demaine, M.L., Eisenstat, S., Patitz, M.J., Schweller, R.T., Summers, S.M. and Winslow, A. Two hands are better than one (up to constant factors). Technical Report 1201.1650, Computing Research Repository, 2012.
12. Chandran, H., Gopalkrishnan, N. and Reif, J.H. The tile complexity of linear assemblies. In *SIAM Journal on Computing 41*, 4 (2012) 1051–1073. Preliminarily version appeared in *ICALP* (2009).
13. Chen, H.-L. and Doty, D. Parallelism and time in hierarchical self-assembly. In *Proceedings of SODA* (2012), 1163–1182.
14. Chen, H.-L. and Goel, A. Error free self-assembly with error prone tiles. *DNA* 2004.
15. Chen, H.-L., Goel, A. and Luhrs, C. Dimension augmentation and combinatorial criteria for efficient error-resistant DNA self-assembly. In *Proceedings of SODA* (2008), 409–418.
16. Chen, H.-L., Goel, A. and Luhrs, C. and Winfree, E. Self-assembling tile systems that heal from small fragments. *DNA* (2007), 30–46.
17. Chen, H.-L., Schulman, R., Goel, A. and Winfree, E. Reducing facet nucleation during algorithmic self-assembly. *Nano Letters 7*, 9 (Sept. 2007), 2913–2919.
18. Cook, M., Fu, Y. and Schweller, R.T. Temperature 1 self-assembly: Deterministic assembly in 3D and probabilistic assembly in 2D. In *Proceedings of SODA* (2011), 570–589.
19. Cook, M., Rothemund, P. and Winfree, E. Self-assembled circuit patterns. *DNA* (2004), 1979–1979.
20. Demaine, E.D., Demaine, M.L., Fekete, S.P., Ishaque, M. Rafalin, E., Schweller, R.T. and Souvaine, D.L. Staged self-assembly: Nanomanufacture of arbitrary shapes with O(1) glues. *Natural Computing 7*, 3 (2008), 347–370. Preliminary version appeared in *DNA* (2007).
21. Demaine, E.D., Eisenstat, S., Ishaque, M. and Winslow, A. One-dimensional staged self-assembly. *DNA* (2011), 100–114.
22. Demaine, E.D., Patitz, M.J., Schweller, R.T. and Summers, S.M. Self-assembly of arbitrary shapes using RNAse enzymes: Meeting the Kolmogorov bound with small scale factor. In *Proceedings of STACS* (2011).
23. Doty, D. Randomized self-assembly for exact shapes. *SIAM Journal on Computing, 39*, 8 (2010), 3521–3552. Preliminary version appeared in *Proceedings of FOCS* (2009).
24. Doty, D., Lutz, J.H., Patitz, M.J., Schweller, R.T., Summers, M. and Woods, D. The tile assembly model is intrinsically universal. In *Proceedings of FOCS* (2012), to appear. IEEE.
25. Doty, D. and Patitz, M.J. A domain-specific language for programming in the tile assembly model. *Proceedings of DNA (2009)*, 25–34.
26. Doty, D. and Patitz, M.J., Reishus, D., Schweller, R.T. and Summers, S.M. Strong fault-tolerance for self-assembly with fuzzy temperature. In *Proceedings of FOCS* (2010), IEEE, 417–426.
27. Doty, D. and Patitz, M.J. and Summers, S.M. Limitations of self-assembly at Temperature 1. *Theoretical Computer Science 412*, 1–2 (Jan. 2011), 145–158. Preliminary version appeared in *DNA* (2009).
28. Fujibayashi, K., Hariadi, R., Park, S.H., Winfree, E. and Murata, S. Toward reliable algorithmic self-assembly of DNA tiles: A fixed-width cellular automaton pattern. *Nano Letters 8*, 7 (2007), 1791–1797.
29. Fujibayashi, K., Zhang, D., Winfree, E. and Murata, S. Error suppression mechanisms for DNA tile self-assembly and their simulation. *Natural Computing 8*, 3 (2009), 589–612.
30. Göös, M. and Orponen, P. Synthesizing minimal tile sets for patterned DNA self-assembly. *DNA* (2010), 71–82.
31. Kao, M.-Y. and Schweller, R.T. Reducing tile complexity for self-assembly through temperature programming. In *Proceedings of SODA* (2006), 571–580.
32. Kao, M.-Y. and Schweller, R.T. Randomized self-assembly for approximate shapes. In *Proceedings of ICALP* (2008), 370–384.
33. Kolmogorov, A.N. Three approaches to the quantitative definition of 'information.' *Problems of Information Transmission 1:1* (1965), 7.
34. Lathrop, J. Lutz, J., Patitz, M. and Summers, S. Computability and complexity in self-assembly. *Theory of Computing Systems 48* (2011), 617–647. Preliminary version appeared in *CiE* (2008).
35. Lempiäinen, T., Czeizler, E. and Orponen, P. Synthesizing small and reliable tile sets for patterned DNA self-assembly. *DNA* (2011), 145–159.
36. Majumder, U., LaBean, T.H. and Reif, J.H. Activatable tiles for compact error-resilient directional assembly. *DNA* (2007), 15–25.
37. Maňuch, J., Stacho, L. and Stoll, C. Step-assembly with a constant number of tile types. In *Proceedings of ISAAC* (2009), 954–963.
38. Maňuch, J., Stacho, L. and Stoll, C. Two lower bounds for self-assemblies at Temperature 1. *Journal of Computational Biology 17*, 6 (2010), 841–852.
39. Padilla, J.E., Liu, W. and Seeman, N.C. Hierarchical self-assembly of patterns from the Robinson tilings: DNA tile design in an enhanced tile assembly model. *Natural Computing 11*, 2 (2012), 323–338.
40. Patitz, M.J. Simulation of self-assembly in the abstract tile assembly model with ISU TAS. *FNANO* (2009), 209–219.
41. Patitz, M.J., Schweller, R.T. and Summers, S.M. Exact shapes and Turing universality at Temperature 1 with a single negative glue. *DNA* (2011), 175–189.
42. Patitz, M.J. and Summers, S.M. Self-assembly of decidable sets. *Natural Computing 10* (2011), 853–877. Preliminary version appeared in UC 2008.
43. Reif, J.H. Local parallel biomolecular computation. *DNA 3* (1999), 217–254.
44. Reif, J. Sahu, S. and Yin, P. Compact error-resilient computational DNA tiling assemblies. *DNA* 2004.
45. Rothemund, P.W.K. Folding DNA to create nanoscale shapes and patterns. *Nature 440*, 7082 (2006), 297–302.
46. Rothemund, P.W.K., Papadakis, N. and Winfree, E. Algorithmic self-assembly of DNA Sierpinski triangles. *PLoS Biology 2*, 12 (2004), 2041–2053.
47. Rothemund, P.W.K. and Winfree, E. The program-size complexity of self-assembled squares (extended abstract). In *Proceedings of STOC* (2000), 459–468.
48. Schulman, R. and Winfree, E. Programmable control of nucleation for algorithmic self-assembly. *SIAM Journal on Computing 39*, 4 (2009), 1581–1616. Preliminary version appeared in DNA (2004).
49. Schulman, R. and Winfree, E. Synthesis of crystals with a programmable kinetic barrier to nucleation. In *Proceedings of the National Academy of Sciences 104*, 39 (2007), 15236–15241.
50. Soloveichik, D., Cook, M. and Winfree, E. Combining self-healing and proofreading in self-assembly. *Natural Computing 7*, 2 (2008), 203–218.
51. Soloveichik, D. and Winfree, E. Complexity of compact proofreading for self-assembled patterns. *DNA* (2005).
52. Soloveichik, D. and Winfree, E. Complexity of self-assembled shapes. *SIAM Journal on Computing 36*, 6 (2007), 1544–1569. Preliminary version appeared in *DNA* (2004).
53. Summers, S.M. Reducing tile complexity for the self-assembly of scaled shapes through temperature programming. *Algorithmica 63*, 1 (2012) 117–136.
54. Turing, A.M. On computable numbers, with an application to the Entscheidungsproblem. In *Proceedings of the London Mathematical Society* (1936), 230–265.
55. Winfree, E. Simulations of computing by self-assembly. Technical Report Caltech CSTR:1998.22. California Institute of Technology, 1998.
56. Winfree, E. Algorithmic Self-Assembly of DNA. Ph.D. thesis, California Institute of Technology, June 1998.
57. Winfree, E. Self-healing tile sets. *Nanotechnology: Science and Computation, Natural Computing Series*. J. Chen, N. Jonoska, and G. Rozenberg, eds. Springer, 2006, 55–78.
58. Winfree, E. and Bekbolatov, R. Proofreading tile sets: Error correction for algorithmic self-assembly. *DNA* (2003), 126–144.
59. Winfree, E., Liu, F., Wenzler, L.A. and Seeman, N.C. Design and self-assembly of two-dimensional DNA crystals. *Nature 394*, 6693 (1998), 539–544.

David Doty (ddoty@caltech.edu) is a Computing Innovation Fellow and Postdoctoral Scholar in the Department of Computing and Mathematical Sciences at the California Institute of Technology, Pasadena.