Tenure Research Statement

David Doty, University of California, Davis, Department of Computer Science

1 Introduction

The history of technology is that of honing our ability to manipulate matter using top-down principles: chopping wood, welding metal, etc. Yet many of the traditional forms of top-down manual control are simply not possible at small scales. Today, we stand on the verge of a new technological era, one that will revolutionize our ability to re-arrange matter at the molecular level, through bottom-up engineering of chemistry: designed molecules, which rely on random molecular movement to interact, yet upon interaction execute designed behavior. Automating the manipulation of molecular structures is not merely faster or more convenient than doing so by hand. Our hands, and the machines they operate, are simply too large to manipulate individual molecules. We must learn how to *program molecules to automatically manipulate themselves*, a field of study known as *molecular programming*.

My research in this area is primarily theoretical, but with one high-profile experimental paper [19] (and some ongoing unpublished experimental collaborations). Below I explain my most significant research since I was hired at UC-Davis in 2015 (occasionally referencing earlier work).

All papers are available at `https://web.cs.ucdavis.edu/~doty/papers/`.

2 DNA nanotechnology

Although molecular programming is possible in principle with a variety of substrates, in practice it has focused on DNA, because of DNA's unique ability to store digital information, to bind selectively, and (due to technological advances in the past few decades) to *process* information in a way that guides the kinetics of interaction.

2.1 Experimental algorithmic self-assembly with DNA tiles

Biology offers inspiring examples of molecules that can store and process information to construct and control the sophisticated nanoscale devices that regulate the machinery of life. Yet biology offers almost no effective *design principles* for manufacturing such molecules ourselves.[1]

Using design principles based on DNA nanotechnology, we demonstrated 21 different Boolean circuits implemented reliably by molecular self-assembly [19], many doing highly non-trivial computation, going far beyond the Sierpinski triangle and binary counter systems that had been implemented by "double-crossover" tiles in previous work. Although the structure formed was a fairly simple one—a "DNA nanotube" consisting of 16 DNA helices that elongates forever—the ultimate goal is to use this sort of computation to guide the formation of the structure itself.

[1] One exception is natural selection, which inspired an effective technique known as *in vitro evolution* for tweaking the function of proteins and other biomolecules. It is akin to modern machine learning: it sometimes works, but when it does, we don't know why. My research follows a *first principles* approach, where we understand the entire system from the ground up. This distinguishes my research from *synthetic biology*, which, at the current time, primarily uses "alien technology": natural proteins repurposed for synthetic engineering. Proteins are effective, but currently, we would not know how to design them, had nature not furnished them.

The breadth and complexity of algorithms we implemented, and the low error-rate achieved (about 0.03% per tile attachment), demonstrate that fabrication of complex, atomically precise structures, using rationally-designed, bottom-up molecular programming, is a feasible long-term goal. Nanoscience need no longer take inspiration only from biology; it can take inspiration from computer science.

DNA tiles consist of *domains*, which are short regions used to bind (though complementary base-pairing) to domains of other tiles. Domains are a DNA implementation of a *specific glue*. The key difficulty with earlier systems was short domain lengths (5 bases), limiting the number of available glues to $\frac{4^5}{2} = 512$, and in fact far fewer than that, since it is critical to avoid *spurious interactions*. For example, $5'$-TGTTT-$3'$ binds strongly to its complement $5'$-AAACA-$3'$, but it binds almost as strongly to $5'$-AAACG-$3'$, so it would cause problems to use both. Single-stranded tiles are a newer motif that naturally use domain lengths of 10 and 11 bases, giving much more leeway to choose domains to implement glues. We designed a system of 355 different tile types with over 400 different glues. The system grows from a "seed" structure encoding the input to the circuit, also made of DNA, using a non-tile technology known as DNA origami.

We defined a formal model of the sort of algorithms our tiles were able to simulate (based on some physical constraints on the sort of topology in which they grow), known as *Iterated Boolean circuits* (IBC), similar to a parallel computer architecture known as a systolic array. The IBC model is surprising powerful. Although our experimental system implemented a small IBC taking only 6 bits as input, we demonstrated a variety of nontrivial computation on these 6 bits: among others, sorting the bits, computing the parity of the number of 1s, deciding whether they represent a multiple of 3 written in binary, deciding whether the input is a palindrome (is equal to its reverse), and simulating cellular automaton Rule 110, known to be efficiently Turing universal: able to simulate any other algorithm with only a polynomial-time slowdown.

The system is capable of randomized computation, where a gate has multiple outputs possible for a given input. This is implemented by having two different tile types compete to bind to the same binding site, with their relative concentrations determining the probability that each wins. We used this ability to implement a nontrivial randomized algorithm: von Neumann's scheme for using a biased coin (represented by competing tiles with unequal concentrations) to simulate a fair coin: flip the coin twice, if it comes up HT, report H (via an certain repeating pattern of 1's in the circuit), if it comes up TH, report T (via the circuit going to a fixed point of all 0's), and if it comes up HH or TT, repeat. In chemical terms, it is a crystal that forms one of two patterns randomly due to random monomer binding, but the two patterns are each exactly equally likely *no matter the relative monomer concentrations*.

More information (e.g., media coverage): `https://web.cs.ucdavis.edu/~doty/papers/#drmaurdsa`

2.2 scadnano web app

This is a browser-based, scriptable tool for designing DNA nanostructures [13].[2] It duplicates most features of cadnano, the most popular tool for designing DNA origami, while addressing many of cadnano's shortcomings (e.g., difficulty of installation, lack of well-documented scripting support).

Since scadnano runs in the browser, no installation is necessary. It is accompanied by a well-documented Python scripting library, installable from PyPI.[3] Extensive documentation is available

[2] `https://scadnano.org/`

[3] `https://pypi.org/project/scadnano/`

for the web interface[4] and the Python package,[5,6] as well as tutorials for each.[7,8]

Many features have been added (and are currently being added) that were lacking in cadnano. These include 1) copy and pasting of strands, 2) altering the "roll" of a DNA helix when visualizing its strands' backbone angles (to visualize angles between helices implied the existing crossovers), 3) hiding a subset of helices (helpful when designing 3D structures, which are a mess to view when projected into a 2D view), 4) visualization of common DNA modifications (such as biotin and fluorophores), and 5) export to file formats recognized by DNA synthesis companies.

Developed in the past year, I'm hoping this tool will be useful and impactful in the DNA nanotechnology community. It's the tool I wish I had when designing DNA origami for my experimental project [19].

2.3 Design of specific molecular bonds

Woo and Rothemund, as well as Gerling, Wagenbauer, Neuner, and Dietz, implemented specific molecular "macrobonds" in DNA, not with DNA base-pairing as is usually done, but with rigid geometrical placement of *non-specific* bonds (specifically something called *blunt-end stacking*). The specificity is enforced by the fact that two macrobonds not intended to bind have their smaller non-specific blunt ends limited in how much they can overlap, even when translated. We investigated the idea of creating many such specific macrobonds, using few non-specific bonds for each, and with limited placement resolution [16]. We borrowed ideas from coding theory, mainly the use of finite-field polynomials, using the fact that two polynomials, if both are low degree, have low overlap. Our main result shows that placing n non-specific bonds on an $n \times n$ grid suffices to create $\approx n^{\lambda-1}$ specific macrobonds, where λ is the desired limit on overlap allowed. (For example, we can create 1210 macrobonds each represented by 11 smaller non-specific bonds each, defined on an 11×11 grid, with overlap of ≤ 4 non-specific bonds between different macrobonds.)

3 Programming molecules with thermodynamics alone

I helped develop an abstract model of molecular binding called *thermodynamic binding networks*. The model captures the fundamentally thermodynamic idea that in any well-mixed chemical system, the spontaneous co-localization of k different molecules is exponentially unlikely in k. We showed that this principle can be used to create Boolean circuits whose correctness derives entirely from thermodynamic forces of entropy and enthalpy, instead of substrate-specific details [14].

In a followup paper [3], we showed that a theoretical system can be constructed implementing a catalytic reaction $C+X \rightleftharpoons C+Y$ such that the "leak" reaction $X \rightleftharpoons Y$, which necessarily occurs at some slower rate, can be made *arbitrarily* slower. A long-term experimental goal is to construct an effectively leakless autocatalytic reaction $C + X \rightleftharpoons 2C$, as a tool for low-false-positive detection of very low molecular count analytes. My Applied Math Ph.D. student David Haley, a key theoretical contributor to this project, has embarked on an ambitious experimental implementation of these ideas in David Soloveichik's wet lab in Austin. Unfortunately David H. had to return to Davis early due to the COVID-19 pandemic. We hope to complete the experiments once travel is safe.

[4] https://github.com/UC-Davis-molecular-computing/scadnano/blob/master/README.md

[5] https://github.com/UC-Davis-molecular-computing/scadnano-python-package/blob/master/README.md

[6] https://scadnano-python-package.readthedocs.io/

[7] https://github.com/UC-Davis-molecular-computing/scadnano/blob/master/tutorial/tutorial.md

[8] https://github.com/UC-Davis-molecular-computing/scadnano-python-package/blob/master/tutorial/tutorial.md

4 Distributed computation with chemical reaction networks

Chemical reaction networks (CRNs) consist of reactions among abstract chemical species such as $A + B \to C$ and $X \to 2Y$. Though a useful tool for modeling natural chemical systems since the mid 19th century, the ability of CRNs to *do computation* was poorly understood until the last decade, when Soloveichik, Cook, Bruck, and Winfree showed that they are able to efficiently simulate arbitrary computation with low probability of error, while also proving that for arbitrary computation the probability of error cannot be made 0. (The result was obtained independently, and earlier, by Angluin, Aspnes, Diamadi, Fischer, and Peralta studying *population protocols*, a special type of CRN where all reactions have two inputs and two outputs.)

When multiple chemical reactions are possible, their relative *rates* influence the probability of which one occurs next. Thus probability-1 computation in CRNs can be thought of as *rate-independent* computation: as long as reactions keep happening, even if far out of proportion to their expected rates (e.g., due to violations of the well-mixed assumption, or imprecisely controlled rate constants in engineered reactions, both known to afflict experimental projects in engineering synthetic reactions), then the computation will eventually stabilize to the correct output. This led Chen, Soloveichik, and I (along with Hajiaghayi on a related project) to develop and characterize exactly the class of functions computable by rate-independent CRNs, in both the settings of discrete integer counts [7, 12] and continuous real-valued concentrations [8].

4.1 Time inefficiency of population protocols/CRNs.

In the discrete setting, certain CRNs stabilize much faster than others. For example, the CRN with one reaction $X \to 2Y$ converts n molecules of X into $2n$ molecules of Y, computing the function $f(n) = 2n$, in expected time $\Theta(\log n)$. The similar CRN $2X \to Y$, computing $f(n) = \lfloor n/2 \rfloor$, requires *exponentially* longer expected time: $\Theta(n)$. Known CRNs for computing many other functions, such as subtraction, minimum, and the constant 1 (also known in distributed computing as the *leader election* problem), also seemed to require $\Theta(n)$ time. This raised the question: are there faster chemical algorithms computing these functions?

We showed that no ("constant-state") population protocol (nor any "reasonable" CRN) can perform leader election faster than the obvious $\Theta(n)$-time algorithm [15] implemented by the reaction $2L \to L$. This paper has been my most influential in the field of distributed computing, having been cited in several other papers as a "breakthrough". This extended techniques used in two earlier papers [6, 9], one of which won Best Paper Award at DISC 2014 [6]. We refined our techniques to show "most" functions computable with probability 1 by CRNs (including division by 2, as well as equality, subtraction, minimum, maximum, and many others) require $\Theta(n)$ time to compute [2], though some need only logarithmic time (specifically, linear functions with positive integer coefficients, e.g., $2x_1 + 3x_2$, via reactions $X_1 \to 2Y$ and $X_2 \to 3Y$). The techniques have been used by others to show stronger lower bounds on the time and memory complexity of leader election and other distributed computing problems such as computing majority.

4.2 Efficient counting and termination with population protocols.

With my Ph.D. student Mahsa Eftekhari, we studied the fundamental distributed computing problem of *counting*: determining the number n of agents are in a network, finding efficient protocols for computing n exactly [11] and approximating n [10]. We broke with convention in recent population protocols in requiring the algorithms to be *uniform*: agents lack any prior estimate on n.

(One goal is to help make those protocols uniform by removing the need to pre-program an estimate of n into the algorithm.)

We also showed that uniform protocols cannot properly "terminate" if all agents start in the same state. Briefly, this means that if all agents have a Boolean field done in their memory, which starts as false, but will eventually be true for all agents, then some agent will set it to true almost immediately (in time approaching 0 as $n \to \infty$). Thus, in these conditions it is impossible to "time" another computation, ruling out one common method of composition where a downstream process is delayed from starting until the upstream process has finished. This lemma has been useful in on other projects, helping rule out ideas that, if they worked, would imply a done signal could be delayed for longer than constant time, a contradiction.

4.3 Composable computation with CRNs.

Composition in CRNs is not straightforward to implement, since it is not clear how to detect when the upstream computation is finished, and its output correct, before allowing the downstream computation to proceed. For example, $X_1 \to Y$ and $X_2 + Y \to \emptyset$ compute the function $f(x_1, x_2) = x_1 - x_2$ whenever $x_1 \geq x_2$, but this cannot be straightforwardly composed with the multiply-by-2 reaction $Y \to 2Z$ to produce $2(x_1 - x_2)$ copies of Z. This is because the first and third reactions can produce $2x_1$ copies (too many) of Z before the second reaction can consume any Y. Y is overproduced by the first reaction, and must be consumed by the second, to compute $x_1 - x_2$, but the third reaction interferes with this required consumption.

With my Ph.D. students Eric Severson and David Haley, we studied CRNs that are *naturally* composable and tolerant to fluctuations in the upstream output, exactly characterizing the class of functions computable by such networks [18], if a "leader" is allowed: a molecule present initially with count 1. This answered an open question (for the leader-driven case) of Chalk, Kornerup, Reeves, and Soloveichik in CMSB 2018.

4.4 Other projects on computation with CRNs

Other projects studied "atomic" CRNs that are interpretable as the rearrangement of indivisible atoms [17] (unlike, say $X \to 2X$), how the output convention in a CRN affects its computational ability [4], as well as papers in submission on efficient fault-tolerant leader election [5] and distributed computing with low communication bandwidth [1].

References

[1] Talley Amir, James Aspnes, David Doty, Mahsa Eftekhari, and Eric Severson. Message complexity of population protocols. In *DISC 2020: Proceedings of the 28th International Symposium on DIStributed Computing, Freiburg, Germany*, 2020.

[2] Amanda Belleville, David Doty, and David Soloveichik. Hardness of computing and approximating predicates and functions with leaderless population protocols. In *ICALP 2017: 44th International Colloquium on Automata, Languages, and Programming*, volume 80, pages 141:1–141:14, 2017.

[3] Keenan Breik, Cameron Chalk, David Doty, David Haley, and David Soloveichik. Programming substrate-independent kinetic barriers with thermodynamic binding networks. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*. to appear. Special issue of invited papers from CMSB 2018.

[4] Robert Brijder, David Doty, and David Soloveichik. Democratic, existential, and consensus-based output conventions in stable computation by chemical reaction networks. *Natural Computing*, 17(1):97–108, 2018. Special issue of invited papers from DNA 2016.

[5] Janna Burman, David Doty, Thomas Nowak, Eric E Severson, and Chuan Xu. Efficient self-stabilizing leader election in population protocols. *arXiv preprint 1907.06068*, 2019.

[6] Ho-Lin Chen, Rachel Cummings, David Doty, and David Soloveichik. Speed faults in computation by chemical reaction networks. In *DISC 2014: Proceedings of the 28th International Symposium on Distributed Computing, Austin, TX, USA, October 12-15, 2014*, volume 8784, pages 16–30, 2014.

[7] Ho-Lin Chen, David Doty, and David Soloveichik. Deterministic function computation with chemical reaction networks. *Natural Computing*, 13(4):517–534, 2014. Special issue of invited papers from DNA 2012.

[8] Ho-Lin Chen, David Doty, and David Soloveichik. Rate-independent computation in continuous chemical reaction networks. In *ITCS 2014: Proceedings of the 5th Innovations in Theoretical Computer Science Conference*, pages 313–326, 2014.

[9] David Doty. Timing in chemical reaction networks. In *SODA 2014: Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 772–784. SIAM, 2014.

[10] David Doty and Mahsa Eftekhari. Efficient size estimation and impossibility of termination in uniform dense population protocols. In *PODC 2019: Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, pages 34–42, 2019.

[11] David Doty, Mahsa Eftekhari, Othon Michail, Paul G. Spirakis, and Michail Theofilatos. Brief announcement: Exact size counting in uniform population protocols in nearly logarithmic time. In *DISC 2018: 32nd International Symposium on Distributed Computing*, 2018.

[12] David Doty and Monir Hajiaghayi. Leaderless deterministic chemical reaction networks. *Natural Computing*, 14(2):213–223, 2015. Special issue of invited papers from DNA 2013.

[13] David Doty, Benjamin L Lee, and Tristan Stérin. scadnano: A browser-based, scriptable tool for designing DNA nanostructures. In *DNA 2020: Proceedings of the 26th International Meeting on DNA Computing and Molecular Programming*, 2020.

[14] David Doty, Trent A. Rogers, David Soloveichik, Chris Thachuk, and Damien Woods. Thermodynamic binding networks. In *DNA 2017: Proceedings of the 23rd International Meeting on DNA Computing and Molecular Programming*, pages 249–266, 2017.

[15] David Doty and David Soloveichik. Stable leader election in population protocols requires linear time. *Distributed Computing*, 31(4):257–271, 2018. Special issue of invited papers from DISC 2015.

[16] David Doty and Andrew Winslow. Design of geometric molecular bonds. *T-MBMC: IEEE Transactions on Molecular, Biological, and Multi-Scale Communications*, 3(1):13–23, 2017. Preliminary version in ISIT 2016.

[17] David Doty and Shaopeng Zhu. Computational complexity of atomic chemical reaction networks. *Natural Computing*, 17(4):677–691, Dec 2018.

[18] Eric E Severson, David Haley, and David Doty. Composable computation in discrete chemical reaction networks. *Distributed Computing*. to appear. Special issue of invited papers from PODC 2019.

[19] Damien Woods†, David Doty†, Cameron Myhrvold, Joy Hui, Felix Zhou, Peng Yin, and Erik Winfree. Diverse and robust molecular algorithms using reprogrammable DNA self-assembly. *Nature*, 567(7748):366–372, 2019. †joint first authors.