# Programming with Chemical Reaction Networks: Mathematical Foundations

Anne Condon, U. British Columbia
Chris Thachuk, U. Oxford
David Doty, Caltech

June 8, 2014–July 13, 2014

## 1 Overview

Here we report on the "Programming with Chemical Reaction Networks: Mathematical Foundations" workshop, held at BIRS in July 2014. Our workshop brought together mathematicians, computer scientists, physicists, and chemists who may not normally cross paths, to answer foundational questions on the capabilities of programs that are described abstractly as chemical reaction networks. What can such programs compute? How energy efficient can they be? How would we verify that such programs are correct?

We'll first give a short overview of chemical reaction networks, the motivation for studying them, and how the workshop was structured for this purpose. In the next section we'll describe the research questions that were the focus of the workshop, and progress on some of these questions.

### 1.1 What are CRNs?

Stochastic chemical reactions networks (CRNs) describe how certain species of molecules within a solution, such as DNA strands, can react to produce new species.

CRNs are specified as a set of reactions along with initial counts of molecular species. An example is a CRN with three reactions

$$\begin{aligned} X + Y &\overset{k_1}{\rightleftharpoons} B + B \\ X + B &\overset{k_2}{\rightleftharpoons} X + X \\ Y + B &\overset{k_3}{\rightleftharpoons} Y + Y, \end{aligned}$$

with initial counts of being $\#X = 10$, $\#Y = 5$ and $\#B = 0$. The likelihood of a given reaction depends on the relative counts of its species and on the number of reactants, and the reaction rate constants [17].

Traditionally used to describe extant chemical systems, CRN's are increasingly being used for molecular programming. Molecular programming is the process of designing molecules that store information and execute instructions in purposeful ways. Nucleic acids—DNA and RNA molecules—are interesting to program because of their digital sequences and dynamic structural properties, and because they extend the reach of computational devices by naturally interacting with "wet" biological systems.

## 1.2 Motivation for the study of CRNs

Because of their simplicity, CRNs are natural language for specifying and reasoning about molecular programs—systems of interacting molecules. Much recent progress in molecular programming has been aided by the use of CRNs, without getting bogged down in low-level details.

CRNs can in principle be "compiled" to produce physical embodiments in the form of so-called DNA strand displacement systems (DSDs). DSDs have been experimentally demonstrated to simulate arbitrary logic circuits [17], compute the square root of a number [15], or implement a tiny neural network [16].

Chemical reaction networks provide a means of exploring intriguing new research directions at the intersection of computer science and biochemistry, such as how to compute in an energy efficient manner in the sense envisioned by Bennett [3], and how to store information at unprecedented scales in a medium that can persist for centuries.

CRNs can also describe programs that execute in biological processes. For example, it has been shown that the cell of eukaryotic organisms computes an approximate majority algorithm to determine when to begin the process of mitosis [4]—partitioning into two daughter cells. In fact, the CRN with three reactions described in section 1.1 computes approximate majority.

Longer term, molecular programs show promise for the development of biosensors capable of "point-of-care" diagnostics (*e.g.,* malaria testing). Molecular programs inserted into cells might act as smart drugs, comprised of logic circuitry that can detect the disease state of a cell and respond with the appropriate treatment, sparing healthy cells from unnecessary dosage.

## 1.3 Interdisciplinary nature of the topic

Some of the challenges faced when designing CRNs are akin to those in distributed computing, and research on CRNs draws naturally from the theory of population protocols. As CRNs are by nature probabilistic and may be applied to problems in human health, verifying their correctness is of paramount concern and builds on formal verification and model checking techniques. Understanding the potential for energy-efficient computation requires the expertise of mathematicians, physicists and complexity theorists.

Although theoretical in nature, this line of research is informed directly by experimental endeavors in programming the behavior of matter at the molecular level, using DNA, enzymes, and other biochemical molecules as engineering tools repurposed from their biological function to serve as computational primitives. To ensure that theoretical models remain grounded in reality, research on this topic benefits from the perspectives of leading experimental scientists.

## 1.4 Structure and outcomes of the workshop

The workshop was structured to provide ample opportunity for those present to learn from those with complementary expertise. Early in the workshop there was a strong focus on identifying important research questions, drawing on the broad perspectives of those present, and facilited by a wiki page. The workshop then included a mix of informal problem-solving sessions and expository talks, with progress reports on the last day.

The workshop was highly interactive, with strong communication among all attendees, particularly between students and more senior researchers. Furthermore, the informal nature of the discussions allowed younger researchers to witness research in its earliest stages, before concrete problems have even been formulated, teaching them by example the creative process of research. This unique environment permitted early stage researchers to learn the main results, techniques, and perspectives of programming chemical reaction networks, to identify unifying conceptual principles

underlying the different projects, and to generate novel research directions for the future based on this interdisciplinary interaction.

There have already been some publications as an outgrowth of the workshop, and more to follow. One open problem discussed at the workshop, concerning a fundamental distributed computing problem known as "leader election" that has been found to be of importance in chemical reaction networks, was resolved by two of the workshop participants, David Doty and David Soloveichik, both of whom were postdocs. Their paper, entitled "Stable leader election in population protocols requires linear time", appeared at the 29th International Symposium on Distributed Computing. Seung Woo Shin, a graudate student funded by the grant, presented his masters thesis research on verifying chemical reaction networks at the workshop. This work was later published at the "2014 Verification of Engineered Molecular Devices and Programs" workshop.

# 2 Research Themes and Open Problems

Here we describe several topics and open problems contributed by workshop participants. For some of the topics, we also describe progress that was made by participants during the workshop; for other topics, progress was largely a better understanding and sharpening of the problem.

## 2.1 Leader election

**Overview.** Biological organisms are exemplars of self-organisation. Mathematical approaches like reaction-diffusion systems attempt to capture geometrical self-organization — how complex patterns can arise from uniform initial conditions. Analogously we ask how well-mixed chemical systems can transform uniform initial conditions to controlled amounts of desired species. Having certain species present in precise counts (eg a species LL with exactly 1 molecule) is a prerequisite for a number of sophisticated CRNs in the literature. For example, constructions for polynomial-time Turing machine simulation requires a "leader": a species with initial count 1, which is used to coordinate interaction among the other species in a controlled way [1].

If a CRN starts with uniform initial conditions where all initially-present species have count proportional to the total volume, then it is possible to "elect" a leader through two reactions:

$$\begin{aligned} X &\to L \\ L + L &\to L \end{aligned}$$

However, this takes expected time $O(n)$, which is slow compared with other basic forms of CRN computation. For example, for example, the reaction $X \to Y + Y$ "quickly" computes multiplication by 2 in the sense that it takes expected time $O(\log(n)$ to convert $n$ copies of $X$ into $2n$ copies of $Y$.

Algorithms for leader election have been extensively studied in the distributed computing literature. The possibility of fast leader election is a long standing open problem in "population protocols," a model which is formally equivalent to a subclass of CRNs. Angluin et al. [2], developed an algorithm which appears to elect a leader in sublinear time with high probability based on extensive numerical simulation. Although the complexity of the population protocol has so far hindered efforts to prove correctness, it informs our understanding of what may or may not be possible, and suggests new techniques for a positive result. Proving either a positive or negative result would be an important development outside of the CRN literature and would engage the larger CS theory community.

**Research questions and challenges.** A central question is: Is there a CRN with an initial state with count $n$ of a species $X$ in volume $n$ and count 0 of everything else, that in expected time $\log^k(n)$ for some constant $k$ reaches a state $\bar{c}$ with 1 copy of $L$ (and arbitrary counts of everything else), such that every state reachable from $\bar{c}$also has count 1 of $L$? There are also many interesting variants of the problem:

- Deterministic versus small probability of error uniform initial state ($n$ copies of $X$ in volume $n$) versus the CRN must handle all initial configurations with $n$ molecules (the latter proved to be impossible to handle) population protocols (2-reactants, 2-products) versus allowing any reactions (the latter can be done by cheating with the reaction $2X \to 3X$)
- How to count the expected time until done: Is it the last time the leader changes (from a copy count of 0 or 2 to 1)? The time at which the CRN enters a state from which it is impossible to change the count of $L$ (which may happen much later than the last time $L$ changes)? Or the first time at which the leader gets count 1?
- Is there only 1 leader species, or do we allow a set $\mathcal{L} = \{L_1, \ldots, L_k\}$ of leader species, so long as eventually exactly one $L_i$ is present?

In each variant, one option makes the leader election problem easier to solve, so that is the option to choose if trying to show leader election is possible, and the other option should be chosen if the goal is to show leader election is impossible. The following observations are good to keep in mind:

- Leader election is not speed fault-free [9].
- If stabilization must happen simultaneously with the last time the leader count changes, that reaction must take $\Omega(n)$ time (assuming there is exactly one leader species; this does not apply if we allow a set $\mathcal{L}$ of several leader species). The lesson is that to prove leader election is impossible, one should focus not on the last reaction to change $L$, but instead on the reaction that stabilizes $\#L$ (or focus on something else).
- It seems leader election can be done if one allows the number of species to grow with $n$ ($O(\log n)$ species seem to suffice).
- Leader election can be done in expected time $\approx O(\sqrt{n})$ with a non-uniform initial state with $nF_1$ and $\sqrt{n}/n^{1/4}$ $F_2$. This violates the hypothesis of a uniform initial state, but it served as a counter-example to several ideas for how to prove leader election is impossible, since those ideas if correct would work on this CRN as well.

## 2.2 CRN Equivalences

**Overview.** Researchers are attempting to use DNA strand displacement cascades to implement a wide variety of chemical reaction networks with interesting algorithmic behaviors. This raises the general question of when we can say such an implementation is correct.

In this context there is a 'formal' CRN, the one being implemented, together with an 'implementation' CRN. The implementation CRN often contains many more chemical species in addition to species corresponding to those in the formal CRN. We call these additional species intermediate species and call those corresponding to species in the formal CRN formal species. Although every reaction that we are trying to implement is a one-step reaction involving only formal species, in the implementation CRN they are often implemented via a multi-step pathway which goes through states containing intermediate species.

There are a number of concepts of equivalence between the formal and implementation CRN, which seek to make precise the notion that the implementation is correct. The first notion is widespread, the second was known in the state-transition system literature but introduced into CRN theory by Qing Dong in 2012, and the third was introduced by Seung Woo Shin in 2011:

- Reachability equivalence. A necessary condition for an implementation to be correct is to require that given any two states containing only formal species, the second is reachable from the first in the formal CRN if and only if it is reachable in the implementation CRN.

- Bisimulation equivalence. A stronger notion is bisimulation equivalence. This is based on the idea of interpreting any intermediate species as a combination of formal species. Bisimulation equivalence requires that the implementation CRN, under such an interpretation, contains exactly the same reactions as the formal CRN. In particular, this ensures that any reaction pathway that exists in one CRN has a corresponding pathway in the other.

- Pathway decomposition equivalence. Pathway decomposition equivalence is a stronger notion than reachability equivalence, but neither implying nor implied by bisimulation equivalence. The main observation that gives rise to this notion is that most pathways of the implementation CRN can be formed by composing smaller pathways. We call those pathways that cannot be thus decomposed "prime pathways". The implementation CRN is pathway decomposition equivalent to the formal CRN if they have the same set of prime pathways.

**Research questions and challenges.** How do these notions compare to those that naturally arise in category theory? The key observation, due to Jose Meseguer, Ugo Montanari and Vladimiro Sassone in the early 1990s, is that any Petri net gives rise to a "symmetric monoidal category": a category where there is a way of adding objects and adding morphisms. However, the formalism of Petri nets is just another language for talking about chemical reaction networks, so any chemical reaction network also gives a symmetric monoidal category. The objects are the formal linear combinations of species, e.g. $A + B + 2C$ where $A, B$, and $C$are chemical species. Addition of these is defined in the obvious way. The morphisms are the reaction pathways, built from the basic reactions in the CRN by composition and addition.

There are various kinds of maps and equivalences between symmetric monoidal categories:

- Symmetric monoidal functor - this is the simplest kind of map between symmetric monoidal categories; it sends objects to objects and morphisms to morphisms in a way that preserves composition and addition.

- Symmetric monoidal equivalence - this is a symmetric monoidal functor $F : C \to D$ equipped with a symmetric monoidal functor G:DCG:DC that serves as an inverse. Two symmetric monoidal categories that have an equivalence of this sort between them are the same for all practical purposes.

- Symmetric monoidal adjunction - this concept is intermediate between the previous two. This is a symmetric monoidal functor $F : C \to D$ equipped with a symmetric monoidal functor $G : D \to C$ that obeys conditions weaker than those of an inverse. In category theory adjunctions are considered very important and much more interesting than equivalences.

It is natural to ask how these concepts relate to the concepts of equivalence for CRNs. Suppose $C$ and $D$ are two symmetric monoidal categories coming from CRNs: $C$ coming from the formal CRN and $D$ coming from the implementation CRN. The existence of a symmetric monoidal functor

$F : CD$ implies that given formal states $S$ and $S'$, if $S'$ is reachable from $S$ in the formal CRN then it is reachable in the implementation CRN. However, the converse may not hold. Thus, reachability equivalence may not hold.

If there is a symmetric monoidal equivalence $F : C \to D$, then reachability equivalence holds, as well as bisimulation equivalence. However, symmetric monoidal equivalence is too strong to hold in most examples found in practice. We successfully applied the more flexible notion of symmetric monoidal adjunction to one realistic example, but we proved that bisimulation equivalence does not imply the existence of a symmetric monoidal adjunction. An important open question is thus to find additional conditions on a symmetric monoidal adjunction that imply bisimulation equivalence.

## 2.3 Robust CRNs: An experimentally motivated questions about CRN-to-DNA technologies

**Overview.** One can write down formal CRNs which are interesting in many ways, such as: exhibiting complex temporal dynamics under the mass action model (oscillations, chaos, ...) or performing computation with counts under the stochastic model. Can we implement any interesting CRN we can write down in a real test tube with real molecules?

David Soloveichik et al. showed that, in theory, it is possible to use DNA strand displacement to approximate the dynamics of any arbitrary formal CRN, with arbitrary accuracy, up to scaling rate constants (and in certain concentration regimes.) Luca Cardelli proposed another CRN to DNA compilation scheme using DNA strand displacement with nicked double stranded DNA complexes.

This inspired many researchers to work on actually engineering complex chemical reaction networks in the wet-lab - in order to understand the kinds of real world problems or issues we would need to solve to really get this technology to work, and these efforts are leading to significant breakthroughs. Following are some questions about CRNs and CRN-to-DNA technologies that are directly experimentally motivated based on experimental experience over the last few years.

**Research questions and challenges.** How can we design or program robust or fault tolerant CRNs? Experimentally, we find that not all our molecules are perfect - some may have synthesis errors - and therefore our implementations are not perfect. So, if we set out to implement the reaction $X \to Y$, it is possible that we really are implementing $X \to (1\delta)Y$, for some $\delta$ that depends on the extent of synthesis errors. Alternatively this could be viewed as "every so often, we consume the reactant(s) without releasing the product(s)". Could we build CRNs that are somehow robust to errors of this kind?

Strand displacement rate can be controlled over 5 to 6 orders of magnitude based on toehold strength, but blunt end or zero toehold strand displacement still happens at a non-zero rate. This means, from an experimental standpoint, that there are no pure implementations of any reactions. In particular, if one wants to implement a bimolecular reaction $A + B \to C$ with rate constant $k_1$, one is really implementing that reaction *and* another, which looks like $B \to C$ with rate constant $k_2$. Ideally, $k_2 << k_1$, but getting this to work in practice can be tricky. Can we build CRNs robust to this kind of more difficult error?

## 2.4 CRNs with limited geometry

**Overview.** Chemical reaction networks are known to be Turing-universal with probability $1 - \epsilon$ (for any $\epsilon > 0$. CRNs augmented with a constant number of stacks are Turing-universal. This was demonstrated using a DNA strand displacement system construction that exploited the fact that DNA can easily form polymers. The stacks in that construction behave as one would expect:

a molecule can be added to the top of the stack, can be removed from the top of the stack, and molecules not on the top are not accessible to other reactions. A stack of type $i$ initiates from a special subunit molecule called $\perp_i$. Simplifying details, this leads to CRN reactions of the form:

$$\begin{array}{rcl} \perp_i \ldots X] + Y & \rightarrow & [\perp_i \ldots XY] \\ [\perp_i \ldots XY] + \text{Pop}_i & \rightarrow & [\perp_i \ldots X] + Y \end{array}$$

As with many results that demonstrate CRNs capable of complex computation the stack machine construction assumes that certain molecular species are present in exact initial quantities, in particular there is exactly one copy of each stack.

**Research questions and challenges.** What is the power of CRNs augmented with stacks, without the initial context assumption (i.e., that the concentrations of stack subunit species, $\perp_i$, are given as part of the input)? The inspiration of this question can be found in the ability for DNA strand displacement systems to form polymers—-complexes consisting of multiple DNA strands bound via hybridization of complementary domains. While polymers can simulate stacks, they are more general and open up new possibilities for programming CRNs. Interesting thought experiments in this model may include:

- Beginning from a population of input $X$, can stacks be used to output $\log(|X|)$ copies of an output species $Y$?
- Could consensus of two input species $X$ and $Y$ be computed exactly, and efficiently, with the use of a constant number of stack types?
- What additional computational power, if any, arises when the model is broadened to included: (i) double ended stacks / queues, (ii) stacks/queues/polymers that can be efficiently concatenated with another, at their ends, (iii) polymers that can be efficiently "merged" with another, at multiple locations along the polymer?

## 2.5 CRNs and circuit complexity

**Overview.** Consider CRNs in which we give input using the binary convention: to specify the $i$th bit is 1, have count 1 of $X_i$ in the initial state; otherwise have count 0.

**Research questions and challenges.** What is the relationship between resource bounds (e.g., number of species/reactions) for CRNs and resource bounds (e.g., number of gates, depth) for Boolean circuits?

Precise question (one example): Let $n \in \mathbb{N}$ and let $\phi : \{0,1\}^n \rightarrow \{0,1\}$ be a Boolean function with $n$ inputs. What is the relationship between the Boolean circuit size $size_{BC}(\phi)$ of $\phi$ (the minimum number of wires in a Boolean circuit with AND, OR, and NOT gates that computes $\phi$) and the "CRN size complexity" $size_{CRN}(\phi)$ of $\phi$ (the minimum number of species/reactions in any CRN that stably computes $\phi$)?

Here, "the CRN stably computes $\phi$" is formally defined in several papers, e.g., [4]. Briefly, it means that there are species $Y$ ("yes") and $N$ ("no"), and starting from the initial state $\bar{x}$ for any state $\bar{c}$ reachable fro $\bar{x}$, there is a stat $\bar{o}$ reachable from $\bar{c}$ with the following property: If $\phi(\bar{x}) = 1$ then $Y$ is present in $\bar{o}$ and $N$ is absent, and if $\phi(\bar{x}) = 0$ then $N$ is present in $\bar{o}$ and $Y$ is absent, and this remains true in any state reachable from $\bar{o}$.

It is clear that $size_{CRN}(\phi) = O(size_{BC}(\phi))$. The reverse direction is not clear.

Some preliminary work suggests perhaps $size_{CRN}(\phi)$ could be characterized in terms of the depth of Boolean circuits that compute $\phi$.

## 2.6 Families of CRNs

**Overview.** Models of computation are often presented either as (i) a single device that works for all inputs (e.g. a Turing machine, a Python program), or (ii) a family of devices where we have one device for each input length (e.g. a family of Boolean circuits with exactly one circuit cncn for each input string length nn). The relationship between standard models and CRNs where in both models we have (i) one device for all inputs is beginning to be well-understood (but still has interesting questions), a number of the questions above are best formalized in the (ii) family of devices setting. The purpose of this section is to describe how the latter setup might look.

Families of devices can be uniform or nonuniform: we say that a family of devices is uniform if there is an algorithm that describes the entire family. For example, we say that a language $L$ is accepted by a uniform Boolean circuit family $\mathcal{C}$ if there exists a computable function $f : 1 \to \mathcal{C}$ such that

$$\mathcal{C} = \{c_n \forall n \in \mathbb{N}, f(1^n) = c_n \text{ is a circuit that accepts} \{0,1\}^n \cap L\}.$$

The distinction between single-machine versus families of devices has appeared in work on CRNs. For example, here are two example ways for a CRN to compute a function of its input.

The input could be the initial count of some single species $X$. Then any natural number nn can be represented, or equivalently one could represent a binary string xx by using natural number nn as input, if $x$ is the $n$th binary string in lexicographic order. The input could be a binary string $x$ of length $n$ represented by nn different input species $X_1, \ldots X_n$. The presence or absence of each of these species is then able to represent any bit string of length nn. Indeed, for each $n \in \mathbb{N}$ there is exactly one CRN, and we call the entire infinite set of CRNs a family.

**Research questions and challenges.** What is the relationship between uniform families of CRNs and uniform families of Boolean circuits, asynchronous Boolean circuits, or other "standard" models? One can also ask about CRN size, time, restricted forms of input and output, and many other questions in this setting). In order not to trivialize the theory, when we have an infinite set of devices computing some function or deciding a language, it is important to set up clear input and output conventions, and to define how powerful the individual devices can be. We propose the following definition of language acceptance by a uniform family of CRNs

Let $\mathcal{R}$ be a set, called a family, of CRNs. We say that a language $L$ is accepted by a uniform family of CRNs $\mathcal{R}$ if there exist two functions: (1) $f : 1 \to \mathcal{R}$ and (2) $e$ (called the input encoder) with domain $0, 1$ and range the set of all multisets over the (input) species of the CRNs, and $\mathcal{R} = \{r_n \forall n \in \mathbb{N}, f(1^n) = r_n$ is a CRN that accepts the input $e(x)$ for all $x \in \{0,1\}^n \cap L\}$.

Usually, we require that $f$ and $e$ are very simple (e.g. logspace computable) in a complexity theoretic sense, as we want the CRN to do the work and not (say) the input encoder $e$. As an example, $e(x)$ could be the indicator species for the string $x$; but we want to consider other input formats, including counts. One could have a similar definition of uniform families of CRNs that compute functions. If there are no (computability) restrictions on ee and ff we say the family is nonuniform; this incredibly powerful model finds most use in proving lower bounds.

## 2.7 Relating space-bounded, logically reversible Turing machines with CRNs

**Overview.** The overarching problem here is to understand how reversible CRNs relate to other reversible computing models, particularly space bounded reversible Turing machines (TMs).

There is a rich history of research on reversible space bounded Turing machines. Following his seminal results on time-bounded, logically reversible TMs, Bennett (1973) asked whether TM-SPACE($s$) = reversible-TM-SPACE($s$). Fifteen years later he made some progress towards this,

showing that TM-SPACE($s$) $\subseteq$ reversible-TM-SPACE($s^2$). Many others worked to improve Bennett's result; finally Lange, McKenzie and Tapp (2000) answered Bennett's 1973 question in the affirmative. More recently, Thachuk et al. described what it means for a DNA strand displacement system (DSD) to use space (or equivalently volume) efficiently. Roughly, just as memory-efficient TMs reuse memory, space-efficient DSDs "recycle" strands by running reactions in both the forward and reverse directions. They also introduced restricted CRN models that can be compiled into DSDs in a space-preserving way.

While these CRN models have plausible physical realizations as DSD's, it is not clear that logically reversible, space-bounded reversible TMs have such physical realizations that preserve space. Roughly, the problem is that such TMs may "run" all of their transitions in the "forward" direction (rather than in both the forward and backwards directions, as do our recycling CRNs and DSDs), and so when one tries a straightforward approach to "compile" these space-bounded TMs into DSDs, the space blows up exponentially. Can this exponential space blow-up be avoided?

**Reearch questions and challenges.** One concrete first step to tackling this would be to study Bennett's 1989 construction. The description is at a fairly high level; it has an elegant and simple recursive structure that should be easily amenable to analysis. Is there a lower-level description, or "implementation" at the TM transition level, in which it is clear that transitions run alternately in the forwards and backwards direction? Progress on this should provide a nice strengthening of traditional complexity-theoretic results on space-bounded logically reversible computation and link this traditional work with current work on CRNs.

# References


[1] D. Angluin, J. Aspnes, and D. Eisenstat. Stably computable predicates are semilinear. In *PODC*, pages 292–299, 2006.

[2] D. Angluin, J. Aspnes, and D. Eisenstat. Fast Computation by Population Protocols With a Leader. In *20th International Symposium on Distributed Computing*, pp61-75, 2006.

[3] C.H. Bennett. Logical reversibility of computation. *IBM journal of Research and Development*, 17(6):525–532, 1973.

[4] L. Cardelli and A. Csikász-Nagy. The cell cycle switch computes approximate majority. *Scientific Reports*, 2, 2012.

[5] E. Cardoza, R. Lipton, A.R. Meyer. Exponential space complete problems for Petri nets and commutative semigroups (Preliminary Report) *Proceedings of the eighth annual ACM symposium on Theory of computing*, pages 50–54, 1976.

[6] H.L. Chen, D. Doty, and D. Soloveichik. Deterministic function computation with chemical reaction networks. *DNA Computing and Molecular Programming*, pages 25–42, 2012.

[7] A. Condon, A.J. Hu, J. Maňuch, and C. Thachuk. Less haste, less waste: on recycling and its limits in strand displacement systems. *Journal of the Royal Society: Interface Focus*, 2(4):512–521, 2012.

[8] Q. Dong. A bisimulation approach to verification of molecular implementations of formal chemical reaction network. Master's thesis. *SUNY Stony Brook*, 2012.

[9] H-L. Chen, R. Cummings, D. Doty and D. Soloveichik, Speed faults in computation by chemical reaction networks, *Distributed Computing*, 1–18, 2015.

[10] D. Doty and M. Hajiaghayi. Leaderless deterministic chemical reaction networks. In *Proceedings of the 19th International Meeting on DNA Computing and Molecular Programming*, 2013.

[11] M.R. Lakin, D. Parker, L. Cardelli, M. Kwiatkowska, and A. Phillips. Design and analysis of DNA strand displacement devices using probabilistic model checking. *Journal of The Royal Society Interface*, 2012.

[12] M.R. Lakin, D. Stefanovic and A. Phillips. Modular Verification of Two-domain DNA Strand Displacement Networks via Serializability Analysis. In *Proceedings of the 19th Annual conference on DNA computing*, 2013.

[13] R. Landauer. Irreversibility and heat generation in the computing process. *IBM Journal of research and development*, 5(3):183–191, 1961.

[14] L. Qian, D. Soloveichik, and E. Winfree. Efficient Turing-universal computation with DNA polymers (extended abstract). In *Proceedings of the 16th Annual conference on DNA computing*, pages 123–140, 2010.

[15] L. Qian and E. Winfree. Scaling up digital circuit computation with DNA strand displacement cascades. *Science*, 332(6034):1196–1201, 2011.

[16] L. Qian, E. Winfree, and J. Bruck. Neural network computation with DNA strand displacement cascades. *Nature*, 475(7356):368–372, 2011.

[17] G. Seelig, D. Soloveichik, D.Y. Zhang, and E. Winfree. Enzyme-free nucleic acid logic circuits. *Science*, 314(5805):1585–1588, 2006.

[18] S. W. Shin. Compiling and verifying DNA-based chemical reaction network implementations. Master's thesis. *California Insitute of Technology*, 2011.

[19] D. Soloveichik, M. Cook, E. Winfree, and J. Bruck. Computation with finite stochastic chemical reaction networks. *Natural Computing*, 7(4):615–633, 2008.

[20] C. Thachuk. *Space and energy efficient molecular programming*. PhD thesis, University of British Columbia, 2012.

[21] C. Thachuk and A. Condon. Space and energy efficient computation with DNA strand displacement systems. In *Proceedings of the 18th Annual International Conference on DNA computing and Molecular Programming*, 2012.

[22] G. Zavattaro and L. Cardelli. Termination Problems in Chemical Kinetics. In Proceedings of the 2008 Conference on Concurrency Theory, pages 477-491, 2008.