TOWARD SPACE- AND ENERGY-EFFICIENT COMPUTATIONS

Anne Condon, University of British Columbia
and Chris Thachuk, California Institute of Technology

Introduction

How might a simulation of computation that is both space and energy efficient be possible? If a Turing machine, on a problem instance of size n, requires $t(n)$time and $s(n)$space to complete, then a simulation of the computation is (1) space efficient if it requires at most $poly(s(n))$space, and (2) energy efficient if it dissipates at most $\epsilon\, t(n)$ energy over the course of the computation, for sufficiently small $\epsilon \geq 0$. Lecerf (1963) and Bennett (1973) made significant progress on this question by introducing the notion of logically reversible computation—previously thought to be a prerequisite for energy-efficient computation[1]—and devising simulations of irreversible Turing machine computations by logically reversible Turing machines with a constant factor increase in time. Bennett (1989), Lange, McKenzie, and Tapp (2000), and others subsequently made progress on space-efficient simulations.

Recent work by Qian, Soloveichik, and Winfree (2011) and others made further significant progress by bridging

[1] Logical reversibility is not a prerequisite for energy-efficient computation (Sagawa 2014; Grochow and Wolpert 2018). For a more nuanced view of this topic beyond logical reversibility, see the chapters by Grochow and Wolpert and by Ouldridge et al. in this volume.

the gap between logically reversible and physically realizable computations using DNA strand displacement systems. Their strand displacement simulations of Turing machines use arbitrarily little energy per step while incurring a quadratic slowdown in time. However, to complete, their simulations may require exponentially more molecules (physical space) than the space used by the Turing machine.

Building on these two earlier threads, we showed how computations that are logically reversible, with balanced, symmetric transitions, have energy-efficient implementations as DNA strand displacement systems and only require a quadratic increase in the number of molecules over the theoretical space required of the computation (Thachuk and Condon 2012).

Here we review these three lines of work with the goals of (1) elucidating the current state of knowledge on the theory of space- and energy-efficient computations, and (2) pointing to fruitful directions for future improvements.

Background

We start with a brief introduction to logical reversibility in the context of time-bounded computation as well as early work on space-bounded, logically reversible computations. We then describe a progression of ideas on how the abstract concept of logical reversibility might be implemented chemically, and how the energy efficiency of chemical implementations can be measured.

LOGICAL REVERSIBILITY AND TIME-BOUNDED COMPUTATION

Landauer (1961) asked: can we build computing devices that are energy efficient, that is, devices that dissipate arbitrar-

ily little energy per step? His work suggested that energy must be dissipated when bits of information are irreversibly erased. This led researchers to consider how logically reversible computation—where information is never irreversibly erased during the computation—may lead to energy-efficient computation. Note that to be reusable for another problem instance, a logically reversible computer must be "reset," a potentially energy-consuming process that includes erasure of the previous input and output.

Lecerf (1963) and Bennett (1973) formalized the notion of logically reversible, deterministic Turing machines. Transition functions of such machines not only have unique images that guarantee determinism, but also unique preimages, thereby ensuring that, when symbols on a tape are overwritten, those symbols can be reconstructed simply by reversing the transition. Thus any computation graph of such a machine is a line from a valid input configuration to a final configuration (i.e., no branching or merging). Turing machines that are logically reversible can simulate arbitrary (irreversible) Turing machine computations with a constant factor increase in the time needed (Bennett 1973); that is, for inputs of length n, if DTIME $(t(n))$ and ReversibleTIME $(t(n))$ are the classes of languages recognizable by deterministic and logically reversible Turing machines in $O(t(n))$ time, respectively, then DTIME $(t(n))$ = ReversibleTIME $(t(n))$.

LOGICALLY REVERSIBLE SPACE-BOUNDED COMPUTATION

Bennett also asked whether Turing machine computations can be simulated in a logically reversibly manner that is efficient with respect to the space of the machine being simulated. Using a recursive simulation of deterministic space-bounded

Turing machines with recursion depth proportional to the space used, he showed that DSPACE ($s(n)$) is contained in ReversibleSPACE ($s^2(n)$) (Bennett 1989).

Whether the quadratic increase in space was necessary remained uncertain for another decade, when Lange, McKenzie, and Tapp (2000) showed that, indeed, logically reversible space equals deterministic space, that is, DSPACE ($s(n)$) = ReversibleSPACE ($s(n)$). Their construction performs an Eulerian tour of the configuration graph of a space-bounded computation.

CHEMICAL IMPLEMENTATIONS OF LOGICALLY REVERSIBLE COMPUTATION

Bennett (1973) envisioned Brownian computers that implement reversible computations chemically, that are "capable of dissipating an arbitrarily small amount of energy per step if operated sufficiently slowly." A single copy of a major reactant, such as a DNA molecule, encodes the current configuration. Reactions in the forward direction involve the major reactant plus additional minor reactants, which are present at "definite concentrations;" these reactions change the major reactant and generate additional (waste) products. The forward direction of each reaction corresponds to a computation step. We describe herein the relationship between the energy dissipated by a reaction and the ratio of the concentrations of minor reactants to products.

Bennett (1981) considered the possibility that reactions of a Brownian computer could arise purely as a result of "random thermal jiggling of its information-bearing parts," with concentrations of minor reactants and products at equilibrium. In this case, a logically reversible computation can proceed in the forward and reverse directions at the same rate while "high-potential-energy barriers" prevent the system from deviating from the computation path. However, he

dismissed calling this possibility computation because, if "the major reactant initially corresponds to the initial state of a v-step computation, the system will begin a random walk through the chain of reactions, and after about v^2 steps will briefly visit the final state. This does not deserve to be called a computation" (Bennett 1973, 531).

Rather, to ensure high probability of completing the computation within reasonable time while keeping energy dissipation per step low, Bennett proposed that the concentrations of minor reactants driving the computation forward be maintained at a small percentage above their equilibrium concentrations. Reactions can still happen in both the forward and reverse directions in this scenario, but the forward bias that results from the higher concentrations of forward-driving reactants, compared with the reaction products, is sufficient to ensure that the final state is reached in a number of steps proportional to the computation length. The forward bias at the final step can be significantly higher to ensure that, once the final configuration is reached, the system will subsequently be in this configuration with high (say, at least 95 percent) probability.

Recently, researchers in the field of DNA computing and molecular programming have developed and implemented mechanisms for computing, with DNA strand displacement systems (DSDs), that are similar in principle to Bennett's proposal involving major and minor reactants (Soloveichik, Seelig, and Winfree 2010). We provide an example of DNA strand displacement later (see fig. 92). Qian, Soloveichik, and Winfree (2011) described how a logically reversible Turing machine could in principle be physically realized with DNA strand displacement operations, in a way that requires arbitrarily little energy per computation step. The worst-case upper bound on the time needed for this simulation is quadratic

in the Turing machine time. An important assumption of this result is that the simulation occurs as an "open" system, where minor reactants are maintained at definite concentration by an external and energy-efficient metabolism process. Under this assumption, the physical space required to hold all major reactants of the simulation (i.e., the required volume) is also efficient with respect to Turing machine space. However, an external metabolism to maintain minor reactants for DSDs is as yet unknown. Operating this simulation in a "closed" system, where all minor reactants required for the simulation to complete are initially present in the same volume as the major reactants, can require exponentially more physical space than Turing machine space. The reasoning will be made clear in subsequent sections where we focus exclusively on simulations in closed systems to account fully for required (physical) space usage.

ACCOUNTING FOR ENERGY IN A CHEMICAL COMPUTATION

We follow the accounting of Bennett (1981) and Qian, Soloveichik, and Winfree (2011) for logically reversible computations where computation steps correspond to the forward direction of chemical reactions. We assume that the computation proceeds in a fixed volume that is maintained at constant temperature. Suppose that the concentrations of forward-driving reactants are X percent in excess of the equilibrium concentrations, relative to the concentrations of the reaction products. Then the energy dissipated (per step) is proportional to $\ln((100+X)/100)k_BT$, where k_B is Boltzmann's constant and T is temperature. The smaller X is, the less energy is dissipated, the higher is the ratio of reverse to forward reactions, and the smaller is the net forward bias of the reaction. If initial concentrations are sufficiently high

relative to the length of the computation, the change in concentrations of reactants and products is negligible throughout, and so the reactions have approximately the same forward bias throughout. If the energy dissipated per step is ϵ for sufficiently small $\epsilon > 0$, the forward/reverse ratio is $e^{\epsilon/(k_B T)}$, and the forward reaction bias is roughly proportional to ϵ.

In contrast, if the concentrations of reactants and products are in equilibrium, the system is driven only by Brownian motion, with forward and reverse reactions being equally likely. The energy dissipated per step is 0.

Although not discussed by Bennett (1981) or Qian, Soloveichik, and Winfree (2011), there is also an energetic cost to setting up the initial configuration of the computation—in our systems, this is the same cost associated with "resetting" the computation for a new input instance—and conducting a readout of the current configuration at any given time. For instance, logically reversible simulations in closed systems typically rely on an initial out-of-equilibrium input state relaxing to equilibrium as the computation proceeds. This "distance from equilibrium" is characterized by the nonequilibrium generalized free energy (Esposito and Van den Broeck 2011; Parrondo, Horowitz, and Sagawa 2015). The point is that the system free energy increases over time until equilibrium is reached. For a logically reversible computation over N states, this difference in free energy is $k_B T \ln(N)$, which is the configurational entropy cost of beginning in a particular state, of N states that are equally likely at equilibrium.

Challenges with Space-Efficient Chemical Implementations of Logically Reversible Computations

The simulation of Qian, Soloveichik, and Winfree (2011) in a closed system requires space (or volume) proportional to the

number of steps of the computation, leaving open the question of how to do computations in both a space- and energy-efficient manner. We next provide an example to illustrate why a DSD-based chemical implementation of space-bounded, logically reversible computations may incur an exponential space blowup.

EXAMPLE: A BINARY COUNTER

For simplicity, we will use chemical reaction networks (CRNs) rather than Turing machines as our programming model. We choose CRNs because of the ease with which we can describe our binary counter, as CRNs are powerful enough to encompass general computation (Soloveichik et al. 2008) and can easily be "compiled" into DSDs (Soloveichik, Seelig, and Winfree 2010).

$$
\begin{array}{cll}
 & (1) & 0_1 \rightleftharpoons 1_1 \\
[] & (2) & 0_2 + 1_1 \rightleftharpoons 1_2 + 0_1 \\
 & (3) & 0_3 + 1_2 + 1_1 \rightleftharpoons 1_3 + 0_2 + 0_1
\end{array}
$$

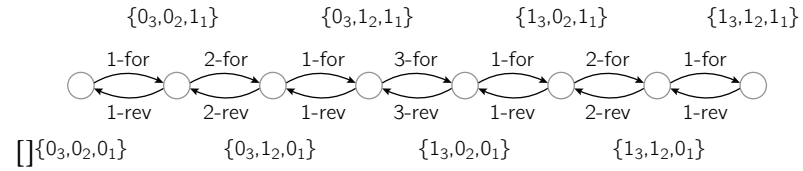

Figure 91. (a) Chemical reaction equations for a 3-bit standard binary counter. (b) The configuration graph of the computation performed by the 3-bit standard binary counter forms a chain and is logically reversible.

A standard n-bit binary counter begins at count 00..0, advances to 00..1, and so on, until reaching the count 11..1. A CRN that implements this counter for $n = 3$ is in figure 91a (Condon et al. 2012). The reactions involve six molecular species, with molecules 0_i and 1_i denoting that bit i has value 0 and 1, respectively, for $1 \leq i \leq 3$. We call 0_i and 1_i *signal*

molecules; these correspond to Bennett's major reactants. The forward direction of the three (reversible) chemical reactions of figure 91a enable the counter to advance in the standard sequence if, initially, a single copy of molecules 0_3, 0_2, and 0_1 are present. The reactions can be generalized to n bits in a natural way, with reactions having up to n reactants and products. Because high-order reactions are atypical in chemical systems, it is worth noting that they can always be emulated by a sequence of binary reactions (Condon et al. 2012).

Figure 91b depicts all possible configurations of the counter as nodes, with edges between configurations that are reachable within one reaction step. Because the configuration graph forms a chain, it represents a logically reversible computation.

DSD IMPLEMENTATION OF A CRN REACTION

We do not know how to design a set of molecules that can directly implement all reactions of an arbitrary CRN without implementing additional (spurious) reactions. By "directly implement," we mean that there is a one-to-one correspondence between the designed molecular species and the species of the CRN.

However, seminal work by Soloveichik, Seelig, and Winfree (2010) demonstrated that an arbitrary CRN can be emulated by a DSD system, in which certain DNA strands represent the molecular species of the CRN (the major reactants) while additional *transformer* molecules (the minor reactants and products) also facilitate each reaction. A *formal* CRN reaction is often emulated by a sequence of *detailed* DSD reactions. See figure 92. Transformer molecules are often described as fuel molecules, because their concentrations can be set so as to bias a computation forward. Many DSD architectures now exist

with this capability (cf. Cardelli 2013; Qian, Soloveichik, and Winfree 2011), and all make use of transformers.

[]

$$\begin{array}{lll} \text{(formal)} & 0_1 & \rightleftharpoons \quad 1_1 \\ \text{(detailed-1)} & 0_1 + T_1^f & \rightleftharpoons \quad I_1 + I_1^r \\ \text{(detailed-2)} & I_1 + I_1^f & \rightleftharpoons \quad 1_1 + T_1^r \end{array}$$

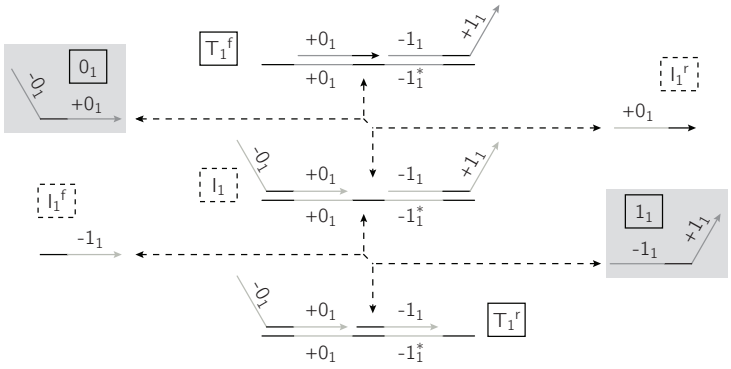

Figure 92. A strand displacement implementation of the reaction $0_1 \rightleftharpoons 1_1$ as proposed by Qian, Soloveichik, and Winfree (2011). (a) The single formal reaction is emulated as a sequence of two detailed reactions in the strand displacement implementation. (b) From top to bottom, the reactant signal strand 0_1 (shown in a shaded box on the left) and transformer molecule (labeled T_1^f, top middle) react, producing an unbound product transformer (labeled I_1^r, top right) and an intermediate complex (labeled I_1, middle). The intermediate complex, together with a second transformer reactant (labeled I_1^f, bottom left), reacts to produce the signal strand 1_1 (shaded box on the bottom right) and a final product transformer (labeled T_1^r, middle bottom). The product transformers can be applied in the opposite direction (from bottom to top) to consume signal 1_1 and produce signal 0_1 as well as the two original reactant transformers. Throughout, connected line segments represent DNA strands, with one-sided arrows denoting the 5' end. Black line segments represent short "toehold" sequences that enable strands to bind weakly to each other via Watson–Crick base pairing, thus enabling the reaction. Gray line segments represent "long domain" sequences that enable molecules to bind together more stably, while also displacing long domains of other strands. Long domains are labeled, and the Watson–Crick complement of a DNA sequence labeled x is labeled x^*.

We illustrate how the first reaction of our 3-bit binary counter CRN can be implemented using DNA strand displacement (fig. 92). From top to bottom, the 0_1 signal strand interacts with a transformer first to become consumed—sequestered on a

double-stranded complex—and ultimately the 1_1 signal strand is produced, released from a double-stranded complex. The strands contained within a shaded box are the signal strands. Two transformers (T_1^f and I_1^f) are reactants, and two transformers (I_1^r and T_1^r) are produced. The product transformers can perform reaction (1) in reverse (from bottom to top in figure 92). The important point is that the reactant transformers are not the same as the product transformers.

Note that this DSD scheme generally applies to any formal reaction: a sequence of detailed reactions each consumes a single reactant (of the formal reaction), and then a subsequent sequence of detailed reactions each produces a single product (of the formal reaction). The example in figure 92 has one reactant and one product and thus requires a sequence of two detailed reactions. Similarly, the reaction $0_2 + 1_1 \rightleftharpoons 1_2 + 0_1$ requires a sequence of four detailed reactions (Qian, Soloveichik, and Winfree 2011).

TAGGED CRNS

To capture this notion of transformer orientation at the level of a CRN, we can *tag* each side of a formal reaction to represent the set of transformers that is necessary to perform a reaction in the respective direction; intermediate reactions and intermediate species arising in a strand displacement emulation can be safely ignored as implementation details. In the case of reversible reactions, when considered as two separate reactions, the forward tag of one constitutes the reverse tag of the other. We call these *tagged chemical reaction equations*. This simple concept of tags allows us to account for the number of transformers and the minimum size of the reaction volume required to complete a computation if it were to be physically realized as a DSD system.

$$
\begin{array}{lll}
(1) & \mathrm{T}^{\mathrm{f}}_{1} + 0_1 & \rightleftharpoons \quad \mathrm{T}^{\mathrm{r}}_{1} + 1_1 \\
(2) & \mathrm{T}^{\mathrm{f}}_{2} + 0_2 + 1_1 & \rightleftharpoons \quad \mathrm{T}^{\mathrm{r}}_{2} + 1_2 + 0_1 \\
(3) & \mathrm{T}^{\mathrm{f}}_{3} + 0_3 + 1_2 + 1_1 & \rightleftharpoons \quad \mathrm{T}^{\mathrm{r}}_{3} + 1_3 + 0_2 + 0_1
\end{array}
$$

Figure 93. Tagged chemical reaction equations for a 3-bit standard binary counter.

ACCOUNTING FOR SPACE IN A TAGGED CRN COMPUTATION

~220~

In a closed DSD system that is simulating a tagged CRN, we assume that the computation proceeds in a fixed volume that is at least proportional to the maximum number of molecules present at any step of the computation. Volume must be as large as the physical space requirement (Cook et al. 2009), and so when we account for space at the CRN level, we must include sufficient space to store not only the signal molecules but also the transformers that facilitate the reactions. Because tags denote sets of transformers, this accounting is straightforward. A stricter accounting would reflect the number of bases of molecules at the DSD level, but, for simplicity, we'll ignore this level of detail and simply count molecules at the tagged CRN level of abstraction, with one tag molecule representing a transformer set.

Returning to the tagged 3-bit standard counter CRN, every configuration has exactly three signal molecules present, denoting the current bit sequence. However, the space requirement for tags is not nearly as succinct. Reaction (1) is used in the forward direction four times during the computation, corresponding to changes in the counter's lowest-order bit; reaction (2) is used twice; and reaction (3) is used once. In total, seven tags are required to advance through the eight states of the counter, and thus seven tags must be present in the initial configuration (and all subsequent

configurations). In general, and despite the fact that all reactions are reversible, the progress of an n-bit counter relies on a sequence of $2^n - 1$ forward reactions and, consequently, $2^n - 1$ tags. In summary, molecular simulations of tagged CRNs in closed systems necessarily use space that grows at least proportionally with time, provided they (1) do not have an external "metabolism" to recycle minor reactants, and (2) always use reactions in the forward direction.

Space- and Energy-Efficient Implementation of Logically Reversible Systems

We continue with the example of a counter to illustrate how to avoid the exponential space blowup of the previous section. The idea is to use reactions in both the forward and reverse directions to drive the computation forward, to reuse transformer molecules. As we discuss later, our use of reactions in both directions resembles symmetric models of computation (Lewis and Papadimitriou 1982).

Reducing the space usage in this way relies on the computation to be unbiased; that is, concentrations of reactant and product transformers should be at equilibrium. However, Bennett asserts that an unbiased, logically reversible computation chain of this form, with output present only in the final state, does not constitute computation. We agree only in that the output of the computation must be observable with high probability. To address this concern, we show how any such chain implemented as a tagged CRN can be augmented to ensure that the output can be read with high probability without an asymptotic increase in time or space usage. We end this section with a summary of known results for space- and energy-efficient CRNs.

A GRAY CODE COUNTER

We review our space-efficient tagged CRN counter (Condon et al. 2012), which is based on the binary reflecting Gray code sequence (Savage 1997). The sequence is a *Gray code* as each successive value differs from the previous in exactly one bit position. It is called a binary *reflecting* Gray code (BRGC) because of its elegant recursive definition: the n-bit BRGC sequence is formed by reflecting the $(n-1)$-bit sequence across a line, then prefixing values above the line with 0 and those below the line with 1. A tagged CRN that implements a 3-bit BRGC counter is given in figure 94a. The recursive nature of the counter can be seen in the computation chain of figure 94b (and is reminiscent of the simulation of Bennett (1981)). For example, when the high-order bit is turned from a 0 to a 1 for the first time (reaction 3 in the forward direction), the sequence of reactions up until that configuration are next executed in reverse as the computation progresses. As a result, each specific reaction strictly alternates between its forward and reverse directions as the computation proceeds forward. Only a single transformer (the forward transformer) per reaction is initially needed to complete the whole computation. While a tagged CRN simulation of a standard n-bit counter requires $\Theta(2^n)$ transformer molecules, the BRGC variant requires only $\Theta(n)$. Note that both counters require only $\Theta(n)$ molecule types.

OBSERVING OUTPUT WITH HIGH PROBABILITY

In this running example of a counter, we define the *output* of the computation to be the counter's final value. As the computation performs an unbiased random walk along the symmetric and logically reversible computation configuration space, the steady state probability of observing the output is $p = 2^{-n}$ for an n-bit counter. This probability can be increased

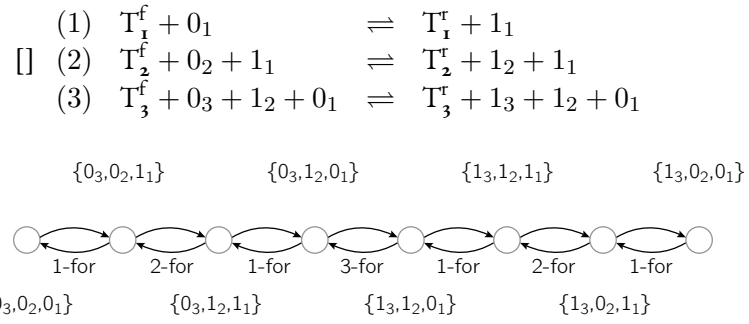

Figure 94. (a) Tagged chemical reaction equations for a 3-bit binary reflecting Gray code (BRGC) counter. (b) BRGC configuration graph. To reach the end configuration, the BRGC counter must perform a sequence of reactions that individually alternate in the forward and reverse directions, thus initially requiring only one transformer per reaction type.

in a number of ways. For instance, by adding one additional reaction type that produces a new signal and requires the final signal multiset of the original computation chain as catalysts, we can double the length of the new chain. The strategy is outlined in figure 95. In this case, the probability of observing the output increases to $p' > 0.5$, because the last state of the first half of the chain and all states of the second half of the chain contain the output. In this manner, for every new reaction added to the CRN, the probability of not observing an output signal is cut in half. Formally, the probability of observing the output becomes $p'' > 1-2^{-c}$ when $c \geq 0$ number of new reactions are added to extend the computation chain. Choosing $c = 5$ meets Bennett's benchmark to observe the output with at least 95 percent probability at steady state, all while maintaining the same asymptotic time and space usage.

ENERGY-EFFICIENT SIMULATIONS OF SPACE-BOUNDED COMPUTATION

Building on the principles of transformer reuse, we previously proposed a tagged CRN—implementable as a DSD—to solve

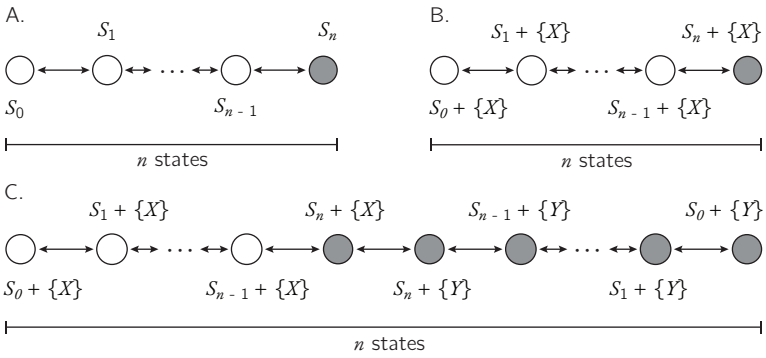

Figure 95. (a) A logically reversible computation chain of a tagged CRN that has a single configuration containing the output (shown shaded). S_i denotes the multiset of signals present in the ith configuration along the computation chain. (b) Adding a single copy of a new species, say, X, to the input configuration S_0, where X is not a reactant in the CRN, does not alter the progress of the computation. (c) The addition of a new reaction—$X + S_n \rightleftharpoons Y + S_n$, where S_n, the multiset of signals present in the original final configuration, catalyzes the new reaction—doubles the length of the computation and has the side effect that the output is now present in more than half of the configurations (shown shaded).

arbitrary instances of the PSPACE-complete quantified 3-satisfiability problem (Thachuk and Condon 2012). For an instance consisting of m clauses over n variables, the tagged CRN requires $\Theta(m + n)$ space. With a polynomial increase in size, the same tagged CRN can be modified to solve any instance over n variables. In this way, our solution to simulating space-bounded computation is not unlike that of a uniform circuit family. Building on known computational complexity results, we show that it is possible to simulate any space-bounded computation with a space- and energy-efficient tagged CRN. While our computation is unbiased, and therefore does not dissipate energy per reaction step, we rely on the computation beginning out of equilibrium in the input state and then relaxing to equilibrium as the computation proceeds. Other simulations, such as those of Bennett (1973) and Qian, Soloveichik, and Winfree (2011), share the assumption of

beginning in a particular input state. In our running example of an n-bit counter that steps through 2^n configurations, beginning in a particular input state requires setting n bits of information initially (i.e., fix all bits to the 0 value). In general, the energy expenditure of our simulation can be bounded as $\Theta(s(n))$ for an $s(n)$-space-bounded tagged CRN computation.

Theorem 2 (Thachuk 2013). *Any problem solvable in $s(n) \geq n$ space can be solved by a logically reversible tagged CRN using $O(s(n)^2 \log s(n))$ space and energy.*

Discussion

As noted earlier, Bennett raised two objections to chemical implementations of computations in which reactions (corresponding to forward computation steps) are unbiased. First, they are slow, because an unbiased random walk through v configurations requires v^2 steps, and second, the final configuration is visited rarely in such a walk. Here we have argued that, despite the first objection, unbiased chemical simulations are valuable to avoid blowup of space when simulating space-bounded computations. A key difference between our chemical simulations and those suggested by Bennett is that both the forward and reverse directions of reactions correspond to forward computation steps. We have also shown how the second objection can be overcome by ensuring that a constant fraction of configurations on the computation path contains the desired output.

Our Gray code counter simulation (and our more general simulations of space-bounded computations) are not only logically reversible but also symmetric, in that reactions can proceed in both the forward and reverse directions—in this sense, "symmetric" implies mechanistically reversible. As such, they are also related to symmetric models of computation, introduced by Lewis and Papadimitriou (1982), where, for a

given symmetric Turing machine, every transition rule t in its set of transition rules $\mathcal{T}$ implies that the inverse transition rule $t^{-1} \in \mathcal{T}$. Chemical implementations of computations in which reactions are unbiased, while slow, seem to be essential for simulation of symmetric computations.

Our preceding analyses focused on closed systems with fixed volume. Might space-efficient DSD implementations of the standard counter, and space-bounded computations more generally, be possible if the reactant transformers flow into the volume as needed to ensure an excess over the product transformers, and waste, that is, product transformers, is removed from the system? In such a scenario, although the volume used by a computation could be linear in the space of the computation being simulated, the number of reactant (forward) transformers required overall and the amount of waste (number of product, or reverse transformers) produced are exponential in the space. For this reason, we argue that such a simulation still requires exponential space, unless a way is found (i.e., an energy-efficient external metabolism) to convert the waste back into forward transformers.

As the DSD systems we presented fall within the larger field of DNA computing, it is interesting to contrast our approach with the field's seminal work by Adleman (1994) on solving large combinatorial search problems using DNA, in terms of space usage. Whereas his work required infeasible counts of molecules to solve very large problem instances, our space requirements are efficient. However, unlike Adleman's approach to DNA computing, our energy-efficient simulation and that of Qian, Soloveichik, and Winfree (2011) have an implicit "single-copy" assumption (Condon et al. 2012) where it is expected that only a single molecule is initially present for certain species types. These simulations fail to be logically

reversible when this assumption is violated (Thachuk 2013).

We conclude with two open questions.

BALANCE VERSUS SPACE EFFICIENCY?

We say that a logically reversible computation is *k-balanced* if, for all transition types, within every computation prefix, the number of times the transition is executed in the forward direction differs from the number of times the transition is executed in the reverse direction by at most k.

The logically reversible simulation of irreversible space-bounded Turing machines of Lange, McKenzie, and Tapp (2000) is not balanced: just like the standard binary counter CRN, transitions are always executed in the forward direction, because they simulate an Eulerian tour of a configuration graph. Thus, if this simulation were implemented chemically using DSDs, it would incur an exponential space blowup.

In contrast, while Bennett did not explicitly design his recursive simulation of irreversible space-bounded Turing machines by logically reversible Turing machines to be balanced, it seems possible to implement it in a balanced way by exploiting the backtracking inherent to recursive procedures to follow transitions in the reverse direction. Perhaps Bennett's simulation can be adapted to show that DSPACE ($s(n)$) is contained in BalancedSPACE ($s^2(n)$), thereby improving Theorem 2 by a logarithmic factor.

Showing that there is a balanced simulation of irreversible space-bounded Turing machines that avoids a quadratic increase in space may be more challenging. That is, if BalancedSPACE ($s(n)$) is the class of languages recognizable by $O(1)$-balanced, logically reversible Turing machines, can we show that DSPACE ($s(n)$) = BalancedSPACE ($s(n)$)?

DSDS WITH UNIVERSAL TRANSFORMERS?

Of course, balanced simulations may not be necessary to avoid space blowups in chemical implementations of logically reversible computations. Is there a DSD system (or other molecular system) in which transformers are universal and thus identical in the forward and reverse directions of a reaction? Such a system could simulate computations in which all reactions occur in the forward direction, such as the standard binary counter, without incurring a significant space increase.

Acknowledgments

We thank Erik Winfree, David Soloveichik, Tom Ouldridge, and David Sivak for helpful discussions and the anonymous reviewers for their feedback.

REFERENCES

Adleman, L. M. 1994. "Molecular Computation of Solutions to Combinatorial Problems." *Science* 266 (5187): 1021–1024.

Bennett, C. H. 1973. "Logical Reversibility of Computation." *IBM Journal of Research and Development* 17 (6): 525–532.

———. 1981. "The Thermodynamics of Computation—A Review." *International Journal of Theoretical Physics* 21 (12): 905–940.

———. 1989. "Time/Space Trade-offs for Reversible Computation." *SIAM Journal on Computing* 18 (4): 766–776.

Cardelli, L. 2013. "Two-Domain DNA Strand Displacement." *Mathematical Structures in Computer Science* 23 (2): 247–271.

Condon, A., A. J. Hu, J. Maňuch, and C. Thachuk. 2012. "Less Haste, Less Waste: On Recycling and Its Limits in Strand Displacement Systems." *Journal of the Royal Society: Interface Focus* 2 (4): 512–521.

Cook, M., D. Soloveichik, E. Winfree, and J. Bruck. 2009. "Programmability of Chemical Reaction Networks." In *Algorithmic Bioprocesses,* 543–584. New York: Springer.

Esposito, Massimiliano, and Christian Van den Broeck. 2011. "Second Law and Landauer Principle Far from Equilibrium." *Europhysics Letters* 95 (4): 40004.

Grochow, J. A., and D. H. Wolpert. 2018. "Beyond Number of Bit Erasures: New Complexity Questions Raised by Recently Discovered Thermodynamic Costs of Computation." *SIGACT News* 49, no. 2 (June): 33–56.

Landauer, R. 1961. "Irreversibility and Heat Generation in the Computing Process." *IBM Journal of Research and Development* 5 (3): 183–191.

Lange, K. J., P. McKenzie, and A. Tapp. 2000. "Reversible Space Equals Deterministic Space." *Journal of Computer Systems Science* 60 (2): 354–367.

Lecerf, Y. 1963. "Logique Mathématique: Machines de Turing réversibles." *Comptes rendus des séances de l'académie des sciences* 257:2597–2600.

Lewis, H. R., and C. H. Papadimitriou. 1982. "Symmetric Space-Bounded Computation." *Theoretical Computer Science* XX:161–187.

Parrondo, J. M. R., J. M. Horowitz, and T. Sagawa. 2015. "Thermodynamics of Information." *Nature Physics* 11 (2): 131–XXX.

Qian, L., D. Soloveichik, and E. Winfree. 2011. "Efficient Turing-Universal Computation with DNA Polymers [extended abstract]." In *Proceedings of the 16th Annual Conference on DNA Computing,* 123–140. Berlin, Heidelberg: Springer-Verlag.

Sagawa, T. 2014. "Thermodynamic and Logical Reversibilities Revisited." *Journal of Statistical Mechanics: Theory and Experiment* 2014 (3): P03025.

Savage, C. 1997. "A Survey of Combinatorial Gray Codes." *SIAM Review* 39 (4): 605–629.

Soloveichik, D., M. Cook, E. Winfree, and J. Bruck. 2008. "Computation with Finite Stochastic Chemical Reaction Networks." *Natural Computing* 7 (4): 615–633.

Soloveichik, D., G. Seelig, and E. Winfree. 2010. "DNA as a Universal Substrate for Chemical Kinetics." *Proceedings of the National Academy of Sciences of the United States of America* 107 (12): 5393–5398.

Thachuk, C. 2013. "Space and Energy Efficient Molecular Programming and Space Efficient Text Indexing Methods for Sequence Alignment." PhD diss., University of British Columbia.

Thachuk, C., and A. Condon. 2012. "Space and Energy Efficient Computation with DNA Strand Displacement Systems." *Lecture Notes in Computer Science* 7433:135–149.