# Space and energy efficient computation with DNA strand displacement systems

Chris Thachuk and Anne Condon

Department of Computer Science, University of British Columbia, Vancouver, BC, Canada

**Abstract.** Chemical reaction networks (CRN's) are important models of molecular programming that can be realized by logically reversible, and thus energy efficient, DNA strand displacement systems (DSD's). Qian *et al.,* [12] showed that energy efficient DSD's are Turing-universal; however their simulation of a computation requires space (or volume) proportional to the number of steps of the computation. Here we show that polynomially space bounded computations can be simulated in both a space and energy efficient manner using logically reversible CRN's and DSD's. A consequence of our proofs is that determining whether a particular molecular species can be produced from an initial pool of molecules of a CRN or DSD is PSPACE-hard, and thus also verifying the correctness of CRN's and DSD's is PSPACE-hard.

## 1 Introduction

The area of molecular programming enjoys active research from both theoreticians and experimentalists due in part to its promise of embedded logical computation that can naturally interface with biological systems. For instance, *if* a condition is detected in a cell, *then* a certain therapeutic agent can be released. Molecular programs can be written in the language of chemical reaction networks (CRN) which detail how sets of reactant molecules can be transformed into new sets of product molecules. A most widely studied and experimentally practical model of computation in molecular programming entails so-called DNA strand displacement systems (DSD). DSDs leverage the fact that substrings of DNA strands will hybridize to their perfect complements and can also displace other bound strands sharing the same substring (see Fig. 1), and can in general realize any CRN [12, 10]. By a careful, non-trivial design of strands one can realize a complex, yet deterministic computation. DSDs have been experimentally implemented and verified to simulate logic gates [14], neural networks [13], and DNA walkers [15], among numerous other applications.

Aside from the potential biological and chemical applications, DSDs and CRNs are also of independent interest due to their promise for realizing energy efficient computation. Rolf Landauer proved that logically irreversible computation—computation as modeled by a standard Turing machine—dissipates energy proportional to the number of bits of information lost, such as previous state information, and therefore cannot be energy efficient [8]. Surprisingly, Charles Bennett showed that in principle energy efficient computation is possible, by proposing a logically reversible universal Turing machine and by identifying nucleic acids (RNA/DNA) as a potential medium for reversible computation [1]. A logically reversible computation is a chain from an initial state to

a final state where each intermediate state has exactly one predecessor and one successor. Bennett's seminal work was space inefficient as his reversible Turing machine simulation required $O(T(n))$ space to simulate a non-reversible machine that required $T(n)$ steps to complete, regardless of its space usage. He later proved that PSPACE equals reversible PSPACE [2]—the class of problems solvable in deterministic polynomial space can be solved by a reversible Turing machine in polynomial space. This result has since been generalized to prove DSPACE equals reversible DSPACE [9]. Until recently, it remained unclear if a physical system could realize logically reversible computation. Qian *et al.,* [12] gave a DSD implementation of a stack machine capable of energy efficient Turing universal computation. Similar to Bennett's seminal work, their implementation requires space proportional to the number of steps in the computation as it consumes *fuel* molecules to drive the overall process forward. Condon *et al.,* [4] demonstrated that, in principle, logically reversible and space efficient computations can be realized in CRNs and DSDs by giving an $n$-bit Gray code counter that progresses through $2^n$ states using only $O(poly(n))$ space.

In this work, we ask the question: can space *and* energy efficient computation be realized? We answer in the affirmative by showing how any problem in PSPACE can be solved by a logically reversible CRN using polynomial space. Our CRN can be realized by an energy efficient DSD implementation. Not only do our results further characterize the computational power of CRNs and DSDs, they shed light on the complexity of a number of important related problems such as CRN and DSD model checking and verification [6, 7]. We show that even determining if an arbitrary state is reachable from an initial state of a CRN or DSD—a question that must be solved when verifying the correctness of a CRN or DSD—is PSPACE-hard. We show that the problem is PSPACE-complete for restricted classes of CRNs and DSDs. Our results also gives strong evidence that predicting low energy barrier folding pathways for multiple interacting nucleic acid strands is PSPACE-complete.

In section 2, we give definitions and the necessary background information for our results. In section 3 we develop our main result by showing how a PSPACE-complete problem can be solved by a logically reversible CRN. In section 4 we show how our CRN can be realized as an energy efficient DSD. In section 5 we demonstrate a number of consequence of our result and resolve the complexity of a number of related problems. We summarize our results in section 6.

## 2 Preliminaries

A *chemical reaction equation* details a process whereby certain molecule types can be consumed—the *reactants*—and others produced—the *products*—within some reaction volume. For simplicity, we assume that the reaction volume is a closed system. A reaction may also require the presence of *catalyst* molecules of certain types. We refer to all three categories, generically, as *signals*. For example, the reaction $A \xrightarrow{C} B$ consumes a signal of type $A$ and produces a signal of type $B$ in the presence of the catalyst signal $C$. This reaction is *irreversible*; however, the reaction $A \overset{C}{\rightleftharpoons} B$ is *reversible* meaning that a signal of type $A$ can also be produced by consuming a signal of type $B$ in the presence of the catalyst signal $C$. A *chemical reaction network* (CRN) is a set of chemical reactions, in addition to a multiset of signals present within the reaction volume,

prior to any reaction occurring, called the *initial* set. A CRN is *proper* if every reaction consumes the same number of signals that it produces. The *state* of a CRN is the current composition of signals within the reaction volume. The CRN can move from some current state, $S$, to a new state, $S'$, by applying any reaction requiring reactants and catalysts present in $S$. In general, many reactions may be applicable for the current state. We define a computation of a CRN $\mathcal{C}$, as a trace of the states from $\mathcal{C}$'s initial state $S_{init}$ to some final target state $S_{end}$. We say that $\mathcal{C}$ is a *logically reversible CRN* if the computation forms a chain from $S_{init}$ to $S_{end}$ such that in any non-terminal state along the chain, exactly two reactions are possible: a reversal of the previous reaction, and one other reaction. In the CRNs considered in this work, we assume applicable reactions are selected with equal probability and with the same rate in the forward and reverse direction. Thus, a computation of length $n$ for a logically reversible CRN will perform an unbiased random walk along the state chain and is expected to reach the end state within $O(n^2)$ steps [5].

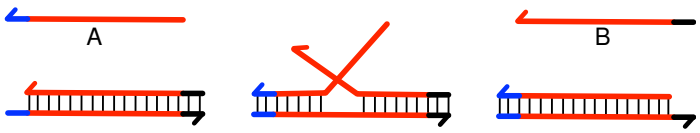


Fig. 1: A gate implementing the reaction $A \rightleftharpoons B$ via toehold-mediated branch migration.

A *DNA strand displacement* (DSD) system is an implementation of a CRN, consisting of single stranded *signals* and double stranded *gates* that facilitate reactions. Strands in the system are composed of two types of domains: short *toehold* domains, and *long domains*. Toehold domains bind reversibly, and long domains irreversibly, to complementary regions on gates. The fundamental operation in a DSD is *toehold mediated strand displacement*, whereby the toehold of a signal strand can bind to an unbound complementary toehold domain of a gate and, if the adjacent long domain is complementary, it can displace a currently bound signal strand of the same length (see Fig. 1).

Consider an example DSD where the initial state consists of two copies of the $A$ signal, but only one copy of the gate $\mathcal{T}$ is available in the reaction volume. It would be impossible to produce two copies of signal $B$. To properly account for gates at the CRN level of abstraction, we augment chemical reactions with unique *tags*. For example, the reaction of Fig. 1 can be described by the tagged reaction $A+\mathcal{T} \rightleftharpoons \mathcal{T}'+B$, denoting that a gate of type $\mathcal{T}$ is required to consume signal $A$, produce signal $B$, and results in the gate being reversed—that is, the same gate can only be used next to consume a $B$ signal, produce an $A$ signal, and reset itself to the forward orientation. We define the space complexity of a trace for a *tagged* CRN as the sum of two quantities: the maximum signal set size of any state in the trace, and the number of tags required to complete the computation. Intuitively, this corresponds to the required size of the reaction volume. In the remainder of the paper we only consider tagged CRNs; however, for simplicity we omit the actual tag signals in the reaction equations. We observe the following obvious, but useful property of proper CRNs.

**Lemma 1.** *A proper CRN with initial set $S$ will always have $|S|$ free signals during a computation.*

CRNs can be implemented by DSDs in a number of ways. We will leverage one such implementation in our results, which relies on the assumption that certain signals only occur as a single copy within the reaction volume. The use of a single copy mutex species is used to ensure that a strand displacement cascade which implements any particular reaction will occur as a transaction and therefore appear *atomic*. Specifically, either the entire cascade implementing a reaction will succeed, or it will return to the state prior to beginning the cascade. Importantly, the mutex molecule is sequestered during the cascade and therefore another reaction cannot begin.

**Theorem 1 (Qian *et al.*, [12], Condon *et al.*, [4]).** *Any logically reversible tagged CRN requiring $O(s)$ space can be simulated by a DSD in $O(poly(s))$ space that ensures reactions appear atomic and occur in the same logical reaction sequence.*

Finally, we formally define three problems we reason about in this work.

CRN REACHABILITY (CRNR)
*Instance*: A chemical reaction network with initial state $S_{init}$ and an arbitrary state $S'$.
*Question*: Is $S'$ reachable from $S_{init}$?

DSD REACHABILITY (DSDR)
*Instance*: A DNA strand displacement system with initial state $S_{init}$ and an arbitrary state $S'$.
*Question*: Is $S'$ reachable from $S_{init}$?

TOTALLY QUANTIFIED 3-SATISFIABILITY (Q3SAT)
*Instance*: A totally quantified boolean formula $\psi$ of $n$ variables in prenex normal form, $\forall x_n \exists x_{n-1} \forall x_{n-2} \ldots Q_1 x_1\ \phi$, where $\phi$ is an unquantified boolean formula of $m$ clauses in conjunction normal form, each containing a literal for 3 distinct variables.
*Question*: Is the formula $\psi$ satisfiable?

## 3 Space efficient CRN simulation of PSPACE

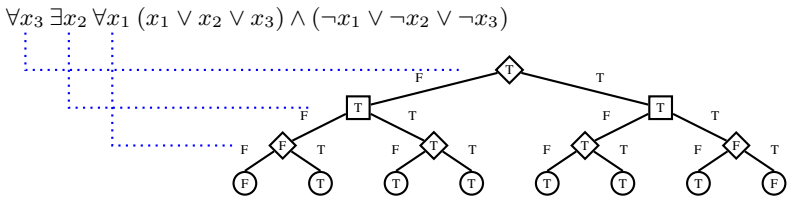


Fig. 2: Solving a Q3SAT instance. Edge labeled paths from root to leaf denote variable assignments. Nodes are satisfied based on quantifier and satisfiability of left and right subtrees.

Our goal is to demonstrate that any problem in PSPACE can be solved by a space efficient, logically reversible, tagged CRN. To that end, we will show how a CRN with

those properties can be constructed to solve any arbitrary instance of the Q3SAT problem. We present our solution in three logical parts. In section 3.1, we demonstrate how to construct a CRN for verifying if a 3SAT formula is satisfied. In section 3.2, we present an elegant solution for traversing a perfect binary tree in post-order that is both space efficient and logically reversible. In section 3.3, we demonstrate how the two CRNs can be integrated and then modified to capture the semantics of strictly alternating variable quantifiers in the Q3SAT instance.

To understand the intuition behind our construction, consider that a perfect binary tree of height $n$, with each level of the tree representing a variable, has $2^n$ leaves, each with a unique path from the root specifying a unique variable assignment. A tree defined in this manner can be used to express the semantics of strictly alternating quantifiers in the Q3SAT instance (see Fig. 2). Leaf nodes are considered satisfied if and only if the current variable assignment satisfies the unquantified 3SAT formula of the Q3SAT instance. Internal nodes can be used to propagate satisfiability of a partially solved instance up the tree. For example, if an internal node represents a universally quantified variable, then it is marked as true if and only if both of its subtrees are satisfied. Similarly, a node representing an existentially quantified variable is marked false if and only if both subtrees are not satisfied. In this straightforward manner, the overall quantified formula can be determined to be satisfied or not, once the root is marked. Since the satisfiability of a node can immediately be determined once that of its two subtrees is known, we perform a post-order traversal of the tree. Furthermore, we exploit the fact that once the satisfiability of a subtree is marked, the satisfiability of its descendants is irrelevant and can be *forgotten*. This allows us to smartly reuse space in our tree traversal procedure.

### 3.1 Verifying a 3SAT instance variable assignment

We first demonstrate how the formula $\phi$ can be verified as satisfied or unsatisfied for a particular variable assignment. A variable assignment ensures exactly one signal for each variable $x_i$ is present: $x_i^T$ for a true assignment, and $x_i^F$ otherwise. We first introduce the necessary reactions to verify an individual clause and demonstrate how the overall formula can be determined true or false.

**Verifying an arbitrary clause.** Recall that in a 3SAT instance, each clause consists of exactly three literals, each for a distinct variable. As such, there are exactly eight possible truth assignments and we create a reversible reaction for each. The reactions for verifying the $i^{th}$ clause, containing literals for variables $x_j$, $x_k$ and $x_l$ are given in Fig. 3 (left). When the clause signal molecule $C_i^?$ is present, exactly one of the eight reactions can be applied, specified by the current variable assignment. The variable signals act as catalysts and the $C_i^?$ signal is consumed producing either a $C_i^T$ signal if the clause is satisfied, or $C_i^F$ otherwise.

For example, suppose $C_i$ represents the following clause: $(x_j \vee \neg x_k \vee x_l)$. The reaction having catalysts $x_j^F$, $x_k^T$, and $x_l^F$ will produce $C_i^F$. The other seven reactions will produce $C_i^T$. Note that for a particular variable assignment, only one reaction will apply in both the forward and reverse direction, ensuring the process is logically reversible.

$$
\begin{array}{lll|lll}
(1) & C_i^? \xrightleftharpoons{x_j^F + x_k^F + x_l^F} C_i^{[F/T]} & , 1 \le i \le m & (2) & \phi^? \rightleftharpoons C_1^? & \\
 & C_i^? \xrightleftharpoons{x_j^F + x_k^F + x_l^T} C_i^{[F/T]} & , 1 \le i \le m & (3) & C_i^T \rightleftharpoons C_{i+1}^? & , 1 \le i < m \\
 & \vdots & & (4) & H_i^? + C_i^F \rightleftharpoons H_i^{\checkmark} + \phi^F & , 1 \le i \le m \\
 & C_i^? \xrightleftharpoons{x_j^T + x_k^T + x_l^T} C_i^{[F/T]} & , 1 \le i \le m & (5) & C_m^T \rightleftharpoons \phi^T &
\end{array}
$$

Fig. 3: (left) Eight reaction equations to verify an arbitrary 3SAT clause $C_i$ for each combination of variable assignments. The product of the reaction is $C_i^T$ for assignments that satisfy the $i^{\text{th}}$ clause, and $C_i^F$ otherwise. (right) Reaction equations to verify the overall 3SAT formula $\phi$, consisting of $m$ clauses.

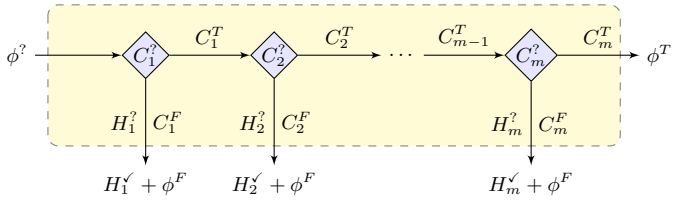


Fig. 4: Flow control when verifying a formula $\phi$ having $m$ clauses.

**Verifying the overall formula.** The overall process of verifying the formula $\phi$ can be thought of as a process that is initiated by consuming the signal $\phi^?$ and completes by producing either the signal $\phi^T$, if $\phi$ is satisfied, or $\phi^F$ otherwise. The variable assignment signals are catalysts, and their values are maintained after the process completes.

For the formula to be true, all clauses must be satisfied. However, any combination of unsatisfied clauses will result in $\phi$ being false. For this reason, care must be taken that clauses are checked systematically to ensure reversibility. The overall process is depicted in Fig. 4 and the reactions are given in Fig. 3 (right). The process checks each clause, in sequence, and if the current clause is unsatisfied then the $\phi^F$ signal is immediately produced in addition to a history signal denoting the first clause to be unsatisfied. The sole purpose of the history signal is to ensure the reversibility of the computation, should the $\phi^F$ signal be produced. Otherwise, all clauses are satisfied, and thus the signal $\phi^T$ can be produced and is sufficient to ensure the computation is reversible.

**Lemma 2.** *A 3SAT boolean formula of $m$ clauses over $n$ variables can be verified by a logically reversible tagged CRN in $O(m)$ reaction steps using $\Theta(m+n)$ space.*

*Proof.* Importantly, we must now establish that the process is logically reversible. We argue formally by induction on $m$, the number of clauses of the 3SAT formula $\phi$. In addition to the clause history domains, we assume initially that the signal $\phi^?$ is present and exactly one signal for each variable $x_i$ denoting its truth assignment—$x_i^T$ or $x_i^F$. Consider the base case when $m = 1$. Given the initial set of signals, only reaction (2) can be applied which produces $C_1^?$. At this point, exactly two reactions can occur: a reversal of the previous reaction, or the clause checking reaction that consumes $C_1^?$

and corresponds to the current variable assignment of the three variables in clause 1. If clause 1 is satisfied (not satisfied), $C_1^T$ ($C_1^F$) is produced. In both cases, other than reversing the previous reaction, only one reaction is possible. If the clause is satisfied, $\phi^T$ is next produced ending the process. If the clause is unsatisfied, $\phi^F$ and $H_1^{\checkmark}$ are next produced, ending the process. Note that in both cases, only the reverse of the previous reaction could be applied next. Thus, the process is logically reversible. Suppose the same holds for $m-1$ clauses and consider the case when $\phi$ has $m$ clauses. We consider two cases:

*Case 1 (The first $m-1$ clauses of $\phi$ are satisfied).* By the inductive hypothesis, signal $C_m^?$ will eventually be produced, in a logically reversible manner. As before, other than the reverse of the previous reaction, only one clause reaction will be applicable and will produce either $C_m^T$ or $C_m^F$. If $C_m^T$ is produced, the reverse of the previous reaction can be applied, or $\phi^T$ is next produced, ending the process. Similarly, if $C_m^F$ was produced, either the reverse of the previous reaction can be applied, or $\phi^F + H_m^{\checkmark}$ is next produced, ending the process. Note that in either case, only one reaction can be applied next: the reverse of the last reaction. Thus, the process is logically reversible.

*Case 2 (At least one of the first $m-1$ clauses of $\phi$ are unsatisfied).* By the inductive hypothesis, this case will correctly produce $\phi^F$ and a history signal denoting the first unsatisfied clause. The new reactions pertaining to clause $m$ are not applicable and thus inconsequential.

It is easy to see that in the worst case, $O(m)$ reactions steps are required. Finally, we establish the space claim. The initial set of signals has size $\Theta(m+n)$ as it consists of the $n$ variable signals, $m$ clause history domains and the signal $\phi^?$. The CRN is proper, therefore by Lemma 1 the number of signals will remain the same throughout the computation. The CRN has $\Theta(m)$ reactions since there are a constant number for each of the $m$ clauses and the overall formula verification. Since each reaction is applied at most once when verifying a formula, one tag per reaction is sufficient, therefore establishing the $\Theta(m+n)$ space bound. □

### 3.2 A space efficient post-order tree traversal

Next we demonstrate how to perform a post-order traversal of a perfect binary tree in a space-efficient manner. Importantly, the procedure must be logically reversible. The intuition is captured in Fig. 5. For any node with a left and right child, once the descendants of the left child have been recursively traversed (Fig. 5 (a)), the left child can be marked (Fig. 5 (b)). Any information stored in those descendant nodes is no longer required and the whole traversal of that subtree can be reversed (Fig. 5 (c)), the traversal can move to the right child (Fig. 5 (d)), the right subtree can be recursively traversed (Fig. 5 (e)), and finally the right child marked (Fig. 5 (f)).

**Lemma 3.** *Given a perfect binary tree of height $h$, all descendants of the root can be traversed in post-order, by a logically reversible tagged CRN, in $\Theta(3^h)$ reaction steps, using $\Theta(h)$ space.*

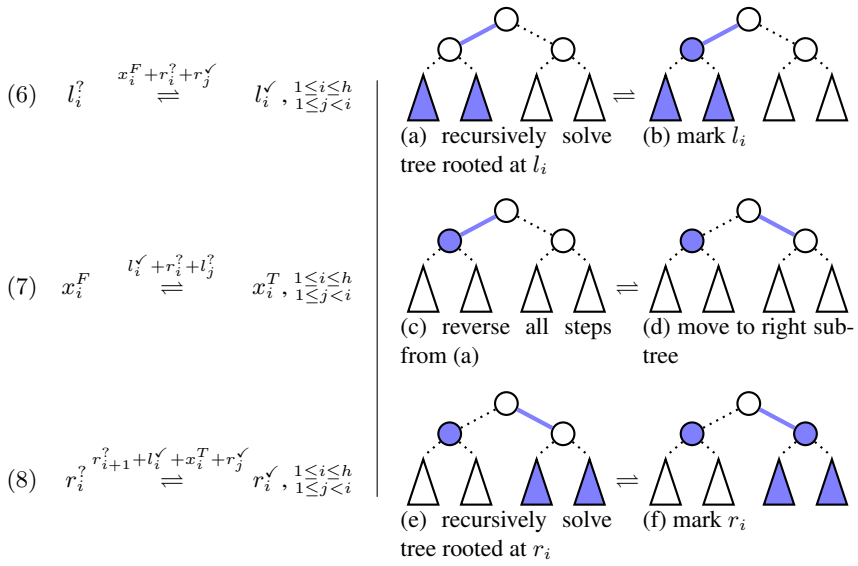


Fig. 5: A logically reversible post-order traversal of all descendants of the root of a height $h$ perfect binary tree.

*Proof.* As each reaction of the CRN is reversible, after every reaction step, the reverse of the previous reaction can always be applied. To demonstrate the CRN is logically reversible, we need to demonstrate that at any point there is at most one other reaction that can be applied. We will further establish the invariant that each reaction strictly alternates in being applied in the forward and reverse direction, ensuring at most one tag is required for each type of reaction. We will argue by structural induction. Let $s_h$ denote the number of reaction steps required for a tree of height $h$.

Consider the base case when $h = 1$ with initial set $\left\{r_2^?, x_1^F, l_1^?, r_1^?\right\}$. Reaction (8) cannot be applied until the signal $x_1^T$ is present which is produced by reaction (7). Similarly, reaction (7) cannot be applied until signal $l_1^\checkmark$ is present. Thus, it is easy to see that reaction (6) must first be applied—marking the left subtree—followed by reaction (7)—moving to the right subtree—and finally reaction (8)—marking the right subtree and completing the traversal in $s_1 = 3$ reaction steps. Each reaction was only applied once, in the forward direction, so the strictly alternating invariant is trivially maintained.

Suppose the traversal completes in $s_{h-1}$ reactions steps, is logically reversible, and the strictly alternating invariant is maintained for a tree of height $h - 1$. Consider a tree of height $h$, having initial set $S = \{r_{h+1}^?\} \cup \bigcup_{1 \le i \le h} \left\{x_i^F, l_i^?, r_i^?\right\}$. Before reaction (6) (and thus reaction (7) and (8)) can be applied, the signals $\bigcup_{1 \le j < h} \left\{r_j^\checkmark\right\}$ must be present. As the left subtree is selected, the signal $r_h^?$ is present, and by the induction hypothesis, the only available action is to produce these signals in $s_{h-1}$ logically reversible reaction steps, that maintain the strictly alternating invariant, by traversing the subtree rooted at $l_h$ (see Fig. 5 (a)). Importantly, the signals $\bigcup_{1 \le j < h-1} \left\{r_j^?\right\}$ are now absent and therefore no reaction affecting levels $1, \ldots, h-2$ can occur. Other than reversing the previous reaction, which produced signal $r_{h-1}^\checkmark$, only reaction (6) can be

applied for level $h$, thus producing $l_h^{\checkmark}$ (see Fig. 5 (b)). Next, observe that reaction (7) cannot be applied until the signals $\bigcup_{1 \leq j < h} \left\{ l_j^? \right\}$ are present. Other than reversing the previous reaction, only a reversal of all $s_{h-1}$ reaction steps that traversed the left subtree can be applied next, yielding the required signals to next apply reaction (7), producing signal $x_h^T$, denoting a move to the right subtree (see Fig. 5 (c) and (d)).

Note that the reversal of the left subtree will maintain the strictly alternating invariant as it ensures all lower level reactions have been reset to their initial state, in order to be used again in the right subtree. Similar to reaction (6), reaction (8) cannot be applied at level $h$ until the right subtree is traversed in $s_{h-1}$ logically reversible reaction steps (see Fig. 5 (e)). Other than reversing the previous reaction, only reaction (8) can next be applied at level $h$ producing the signal $r_h^{\checkmark}$ and ensuring no further reactions on lower levels can occur. The traversal is complete and no reaction, other than the reverse of the previous, can occur. Thus, the overall traversal is logically reversible, and is clearly in post-order. As the strictly alternating invariant was maintained for all lower level reactions, and all reactions at level $h$ have been applied for the first time, and only once, the invariant is maintained for a tree of height $h$.

Exactly 3 reactions occurred at level $h$, and $3s_{h-1}$ reactions were required for the two traversals and one reversal of the height $h-1$ subtrees, giving us the recurrence $s_h = 3s_{h-1} + 3$. Solving $s_h$ with $s_1 = 3$ gives us the closed form expression $\frac{3}{2}(3^h - 1)$, establishing the claimed $\Theta(3^h)$ reaction steps.

Finally, consider the space claim. As we have shown that reactions strictly alternate being applied in the forwards and reverse direction, at most one tag for each of the $\Theta(h)$ reactions is sufficient. Consider that the initial set for a tree of height $h$ is $S = \{r_{h+1}^?\} \cup \bigcup_{1 \leq i \leq h} \left\{ x_i^F, l_i^?, r_i^? \right\}$ and therefore $|S| = 3h + 1$. Since the CRN is proper, we immediately establish the space claim by Lemma 1. □

### 3.3 Solving a Q3SAT instance

We now have the means to verify if a variable assignment satisfies a 3SAT formula $\phi$. We can also traverse a perfect binary tree in post-order, and in the process enumerate all possible variable assignments for $\phi$. What remains is to combine these processes together in order to determine if a Q3SAT instance can be satisfied. We approach the integration in two parts. First, we will demonstrate how the formula verification process can be triggered immediately prior to the tree-traversal marking a leaf node and how the verification reactions can be entirely reversed, prior to the next time the verification procedure must run. This effectively demonstrates how any problem in NP can be solved by a logically reversible CRN in polynomial space, if we specify the end of computation as the presence of the signal $\phi^T$, or the signal $\phi^F$ in conjunction with the signals for the final variable assignment to be enumerated. Finally, we demonstrate how the tree traversal reactions of Fig. 5 can be augmented in order to capture the semantics of alternating universal and existential quantifiers, thus demonstrating how any problem in PSPACE can be solved in polynomial space by a logically reversible CRN.

**Integrating formula verification and tree traversal.** Recall the sequence of logical steps in traversing level 1 of the tree, *i.e.,* the leaves: (1) *mark left*, (2) *move right*, (3) *mark right*. We augment the reactions for level 1 to force the following sequence: (1a)

*verify* $\phi$, (1b) *mark left*, (2a) *reverse (1a)*, (2b) *move right*, (3a) *verify* $\phi$, (3b) *mark right*. This new sequence ensures two invariants: (i) the current variable assignment is verified prior to marking the current leaf, and (ii) the verification procedure is fully reversed prior to the next verification.

The augmented reactions are given in Fig. 6. Both reactions marking a leaf have been split into two variants, each ensuring the verification procedure has completed by requiring as a catalyst one of the two possible outcomes of the verification process. In addition, we add new signals to record whether or not the variable assignment for a particular leaf is a satisfying assignment for $\phi$. These signals will be used later to propagate satisfiability up the tree, once quantifiers have been integrated. Note that the reaction to move to the right leaf now requires the signal $\phi^?$ as a catalyst. This forces all steps performed in the previous verification to reverse. After moving to the right leaf, and thus swapping the value of variable $x_1$, the verification process can again run immediately prior to marking the right leaf. Importantly, we want to ensure that the verification procedure is completely integrated into the leaf level reactions and cannot perform any reactions while the traversal is marking higher level nodes. This is easily accomplished by augmenting reactions (2)-(5) to require $r_2^?$ as a catalyst. Note that the augmented variants of the tree traversal reactions are also fully distinguishable by their catalysts (and products), thus ensuring the process is logically reversible.

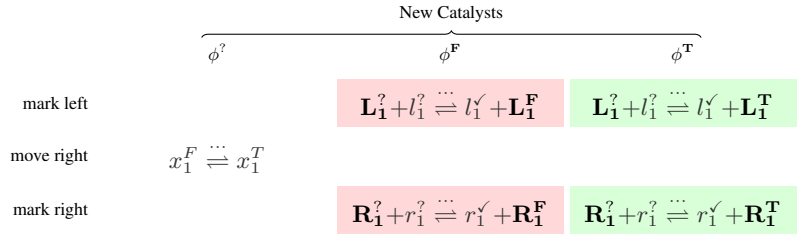


Fig. 6: Integrating the 3SAT verification procedure into the leaf level reactions of the tree traversal procedure. Two reaction variants are created for marking leaf nodes as either satisfied or unsatisfied based on the result of the verification procedure. The *move right* reaction requires $\phi^?$ as a catalyst, thus ensuring the verification procedure is reversed prior to the next verification step. Existing catalysts omitted for space.

**Integrating quantifiers into the tree traversal.** Integrating quantifiers in non-leaf levels of the tree is relatively straightforward. Recall that the levels of the tree strictly alternate between universal and existential quantification. For each level, we create four variants of the correct quantifier for both the left and right node marking reactions to additionally produce a signal indicating if the current subtree is satisfied. The reaction variants for marking a left node are given in Fig. 7. These reactions require as catalysts the signals indicating if the left and right subtree of the current node is satisfied and therefore four variants are sufficient to consider all cases, for each type of quantifier. As with the leaf level reactions, the augmented reaction variants can be fully distinguished by their catalysts ensuring the computation remains logically reversible, and the correct reactions are reversed.

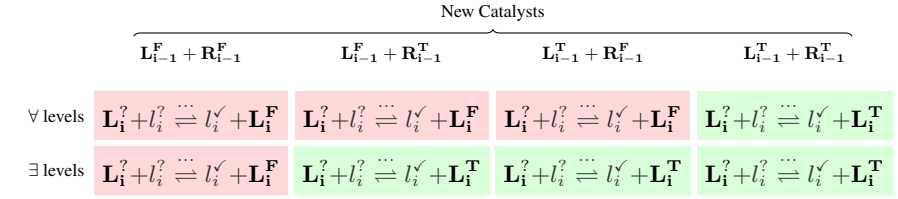


Fig. 7: Integrating quantifiers to non-leaf levels of the tree traversal. For both universal and existential levels, four variants of the left node reactions are created to process the four combinations of left and right subtree satisfiability. The integration is identical for right node reactions. Existing catalysts omitted for space.

**Ending the computation.** Once both subtrees of the root have been solved the output signal can be produced based on the satisfiability of the subtrees and on the quantifier imposed on the root level variable $x_n$. The reaction equations for the universal quantifier are shown in Fig. 8. Modifying the reactions for an existential quantifier is straightforward.

Recall that reactions at level $n-1$ cannot proceed unless the signal $r_n^?$ is present. We could have the reaction producing the solution signal also consume $r_n^?$. This would end the computation chain as only reversing the previous reaction would be possible next. However, for reasons we will make clear in the following section, the signal $r_n^?$ is never altered and therefore after the solution signal is produced, the entirety of the tree traversal steps will be reversed before reaching the end of the computation chain. The entire configuration of the CRN system will appear identical to the initial configuration, with the exception that the output has been written (*i.e.,* the $\psi^?$ signal has been consumed and been replaced by $\psi^F$ or $\psi^T$). See Fig. 9 for a schematic of the logically reversible computation chain.

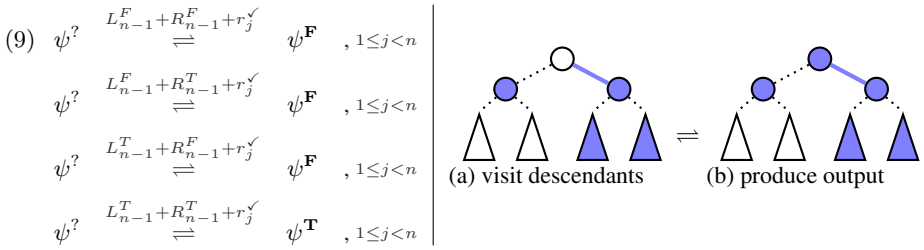


Fig. 8: After both subtrees of the root have been solved a solution can be determined based on the quantifier of the root level. Equations are shown assuming the root variable $x_n$ is universally quantified.

**Theorem 2.** *Any arbitrary instance of* Q3SAT *with $n$ variables and $m$ clauses can be solved by a logically reversible tagged CRN in $O(m\,3^n)$ reaction steps using $\Theta(m+n)$ space.*

*Proof.* Let $\psi$ be the totally quantified boolean formula of the instance and $\phi$ be the unquantified 3SAT formula. By Lemma 2 a set of $\Theta(m)$ reactions can be created to verify if $\phi$ is satisfied, or not, for a particular variable assignment. By Lemma 3, a set of $\Theta(n)$ reactions can be created to traverse the height $n$ tree representing all possible assignments of the $n$ variables. Furthermore, the above modifications demonstrate how these two processes can be integrated into one logically reversible computation chain, and how quantifiers can be added to the non-leaf levels to determine if there is a satisfying solution for $\psi$ by propagating satisfiability of subtrees up to higher levels. Importantly, the modifications only increase the number of reactions by a constant factor and are designed to maintain the property that the computation is logically reversible. Consider that the number of reaction steps acting on a tree node, prior to reaching the root, has not increased. However, prior to marking every leaf in the traversal, the verification procedure is run for the current variable assignment (and reversed in between). Therefore, by Lemmas 2 and 3, the root of the height $n$ tree can be reached, and a solution signal produced, within $O(m\,3^n)$ reaction steps. As forcing the entire tree traversal to reverse prior to the end of computation only doubles the number of reaction steps, the claim on computation length is established.

Next, consider the space required of the combined CRN. The modified verification procedure requires the following initial set, where $\mathcal{T}_{3sat}$ is the set of required tags: $\mathcal{S}_{3sat} = \bigcup_{1\leq i\leq m}\left\{C_i^?, H_i^?\right\} \cup \left\{\phi^?, r_2^?\right\} \cup \mathcal{T}_{3sat}$. The augmented tree traversal procedure requires the following initial set, where $\mathcal{T}_{tree}$ is the set of required tags: $\mathcal{S}_{tree} = \bigcup_{1\leq i<n}\left\{l_i^?, x_i^?, r_i^?, L_i^?, R_i^?\right\} \cup \left\{r_n^?, \phi^?, \psi^?\right\} \cup \mathcal{T}_{tree}$. The space required for the combined CRN is therefore $|\mathcal{S}_{q3sat}| = |\mathcal{S}_{tree} \cup \mathcal{S}_{3sat}|$. As the combined CRN maintains the property that reactions strictly alternate being applied in the forward and reverse direction, then one tag for each of the $\Theta(m+n)$ reactions is sufficient and $|\mathcal{S}_{tree}| \in \Theta(m+n)$. □

As Q3SAT is a complete problem for PSPACE [11], we immediately have the following.

**Corollary 1.** *Any problem in* PSPACE *can be solved by a logically reversible tagged CRN using polynomial space.*

## 4 Space and energy efficient DSD simulation of PSPACE

The remarkable consequence that Bennett's work demonstrates is that energy consumption is not necessarily an intrinsic cost of computation. In particular, if the computation is logically reversible, there is no theoretical energy expenditure. However, there must be a reasonable probability the actual solution can be observed. This can be problematic in a logically reversible computation which is free to immediately reverse once reaching a solution state. Qian *et al.,* [12] solved this problem by using fuel to provide a slight bias for remaining in a solution state once the computation completes. However, in our result, since reactions must be reused efficiently in both directions to maintain a polynomial space bound, they cannot be biased in general.

To overcome this, we have designed our reactions that produce an output signal to ensure the next logical step in the computation is to reverse the tree traversal. This effectively doubles the length of the logically reversible computation chain and established the important property that the output signal can be observed in strictly more than

half of the states (see Fig. 9). Notice that this was also the case for Bennett's original reversible Turing machine implementation[1]. As the computation performs an unbiased random walk along the logically reversible computation state space, the steady state probability of observing the output signal is $p > 0.5$. This probability can be further increased in a number of ways. For instance, at the DSD level, we could design the gates which implement the reactions producing the output signal to have a slight bias in the forward direction, by manipulating relative toehold lengths, effectively biasing our overall computation towards the second half of the chain shown in Fig. 9 [17]. As this reaction is only performed once, the gate implementing the reaction is not reused and therefore, the bias is not problematic for the overall computation to complete.

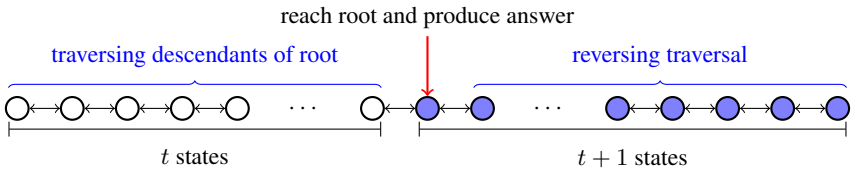


Fig. 9: The logically reversible computation chain of the Q3SAT CRN. In more than half of the states, the output signal is present (shown shaded).

Combining Theorem 1 and Corollary 1 we have the following.

**Theorem 3.** *Any problem in* PSPACE *can be solved by a space and energy efficient DSD.*

We note that the CRN and DSD description given here is a non-uniform model of computation. Specifically, the CRN description is dependent on, and encodes, a particular problem instance. Therefore, different problem instances will result in different CRN descriptions and thus a different DSD implementation. Particularly at the DSD level, where synthesizing strands and gates is challenging, it would be desirable if only the input strands differed between unique instances. This may be achievable by constructing a more general quantified boolean formula that is within a polynomial size of the original encoding described here. In such a construction, part of the input would describe which clauses are active for the particular problem instance. The generalized formula would be for a fixed number $n$ of variables and could be used to solve any instance having at most $n$ variables. We will explore the details of such a construction in the full version of this manuscript.

## 5 Complexity of verifying CRNs and DSDs

Next we show there exists a polynomial time and space reduction from an arbitrary Q3SAT instance $I$ into an instance $I'$ of the CRN reachability problem (Q3SAT $\leq_p$ CRNR), such that $I$ can be solved if and only if $I'$ can be solved.

[1] The forward traversal of the tree, production of the output signal, and reversal of the traversal are analogous to the *compute*, *copy output*, and *retrace* phases of Bennett's original reversible Turing machine simulation [1].

**Theorem 4.** *The reachability problem for CRNs (*CRNR*) is* PSPACE*-hard.*

*Proof.* Given an arbitrary Q3SAT instance, construct the CRN of Theorem 2 which is of polynomial size and can therefore be constructed in polynomial time and space. Ask the question of whether the state $S_{init}/\{\psi^?\} \cup \{\psi^T\}$ can be reached from $S_{init}$, where $S_{init}$ is the initial state of the CRN. □

By Lemma 1 it is easy to see the reachability problem for proper CRNs is in PSPACE. Whether other forms of CRNs are in PSPACE is dependent on their definition and how the required space to complete a computation is accounted for. Any tagged CRN accounts for the necessary fuel as part of the size of the reaction volume and therefore, by this interpretation, is in PSPACE.

**Corollary 2.** *The reachability problem for proper/tagged CRNs is* PSPACE*-complete.*

We note that other results are known for unrestricted CRNs which are not studied here. (Un-tagged) reversible CRNs correspond to reversible Petri nets where the reachability problem is EXPSPACE-complete [3]. CRN reachability has also been studied for the probabilistic case [16, 18] and nondeterministic case [18] and the connection with Petri nets was also explored [18].

By Theorem 1 and Theorem 4 we immediately have the following analogous results for DSDs.

**Corollary 3.** *The reachability problem for DSDs (*DSDR*) is* PSPACE*-hard.*

Clearly the reachability problem is PSPACE-complete for the set of DSDs implementing a proper CRN. When fuel molecules are considered part of the space usage, as would be the case for closed volumes that are studied here, then the reachability problem is PSPACE-complete. We also conjecture that a DSD instance created by the above chain of reductions (*i.e.,* Q3SAT $\leq_p$ CRNR $\leq_p$ DSDR), can be adapted to show the minimum energy barrier folding pathway prediction problem is PSPACE-complete. The challenge in achieving this result is to properly consider the possibility of blunt-end displacements, which are generally assumed to not occur when reasoning about DSD systems. We will explore the details of the proof for the full version of the manuscript.

*Conjecture 1.* The minimum energy barrier pseudoknot-free folding pathway problem for multiple interacting nucleic acid strands is PSPACE-complete.

## 6 Conclusions

In this work, we asked the question: can space *and* energy efficient computation be realized by chemical reaction networks (CRN) and DNA strand displacement systems (DSD)? We have shown this can be achieved in general by giving a logically reversible space efficient CRN implementation capable of solving any problem in PSPACE—the class of all problems solvable in polynomial space. Furthermore, our CRN can be realized by a space and energy efficient DSD. In addition to further characterizing the computational power of standard molecular programming systems, our result has a number of important consequences. For instance, we show that even determining if

a certain state is reachable in a CRN, such as a desirable or undesirable configuration, is PSPACE-hard, effectively demonstrating the intrinsic complexity of model checking and formal verification of chemical reaction networks. We further show the problem is PSPACE-complete for a restricted class of CRNs. The results also hold at the DSD level and give strong evidence of the hardness at the sequence level for the related problem of predicting minimum energy barrier folding pathways between two configurations of multiple interacting nucleic acid strands.

## References


1. C.H. Bennett. Logical reversibility of computation. *IBM journal of Research and Development*, 17(6):525–532, 1973.
2. C.H. Bennett. Time/space trade-offs for reversible computation. *SIAM Journal on Computing*, 18(4):766–776, 1989.
3. E. Cardoza, R. Lipton, and A.R. Meyer. Exponential space complete problems for Petri nets and commutative semigroups. In *Proceedings of the eighth annual ACM symposium on Theory of computing*, pages 50–54. ACM, 1976.
4. A. Condon, A. Hu, J. Maňuch, and C. Thachuk. Less haste, less waste: on recycling and its limits in strand displacement systems. *DNA Computing and Molecular Programming*, pages 84–99, 2011.
5. W. Feller. *An Introduction to Probability Theory and Its Applications, Vol. 1*. Wiley, 1971.
6. M. Lakin and A. Phillips. Modelling, simulating and verifying Turing-powerful strand displacement systems. *DNA Computing and Molecular Programming*, pages 130–144, 2011.
7. 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.
8. R. Landauer. Irreversibility and heat generation in the computing process. *IBM Journal of research and development*, 5(3):183–191, 1961.
9. K.J. Lange, P. McKenzie, and A. Tapp. Reversible space equals deterministic space. *Journal of Computer Systems Science*, 60(2):354–367, 2000.
10. L.Cardelli. Two-domain DNA strand displacement. In *Proc. of Developments in Computational Models (DCM 2010)*, volume 26 of *Electronic Proceedings in Theoretical Computer Science*, pages 47–61, 2010.
11. Christos M. Papadimitriou. *Computational complexity*. Addison-Wesley, Reading, Massachusetts, 1994.
12. L. Qian, D. Soloveichik, and E. Winfree. Efficient Turing-universal computation with DNA polymers. In *Proceedings of the Sixteenth Annual Conference on DNA Computing and Molecular Programming*, pages 123–140, 2011.
13. L. Qian, E. Winfree, and J. Bruck. Neural network computation with DNA strand displacement cascades. *Nature*, 475(7356):368–372, 2011.
14. G. Seelig, D. Soloveichik, D.Y. Zhang, and E. Winfree. Enzyme-free nucleic acid logic circuits. *Science*, 314(5805):1585–1588, 2006.
15. J.-S. Shin and N.A. Pierce. A synthetic DNA walker for molecular transport. *J Am Chem Soc*, 126:10834–10835, 2004.
16. D. Soloveichik, M. Cook, E. Winfree, and J. Bruck. Computation with finite stochastic chemical reaction networks. *Natural Computing*, 7(4):615–633, 2008.
17. Erik Winfree. Personal communication, 2012.
18. G. Zavattaro and L. Cardelli. Termination problems in chemical kinetics. In *Proceedings of the 19th International conference on Concurrency Theory*, pages 477–491, 2008.