A Coupled Reconfiguration Mechanism That Enables Powerful, Pseudoknot-Robust DNA Strand Displacement Devices with 2-Stranded Inputs

Hope Amber Johnson
The University of British Columbia, Vancouver, Canada

Anne Condon
The University of British Columbia, Vancouver, Canada

Abstract

DNA strand displacement, a collective name for certain behaviors of short strands of DNA, has been used to build many interesting molecular devices over the past few decades. Among those devices are general implementation schemes for Chemical Reaction Networks, suggesting a place in an abstraction hierarchy for complex molecular programming. However, the possibilities of DNA strand displacement are far from fully explored. On a theoretical level, most DNA strand displacement systems are built out of a few simple motifs, with the space of possible motifs otherwise unexplored. On a practical level, the desire for general, large-scale DNA strand displacement systems is not fulfilled. Those systems that are scalable are not general, and those that are general don't scale up well.

We have recently been exploring the space of possibilities for DNA strand displacement systems where all input complexes are made out of at most two strands of DNA. As a test case, we've had an open question of whether such systems can implement general Chemical Reaction Networks, in a way that has a certain set of other desirable properties – reversible, systematic, $O(1)$ toeholds, bimolecular reactions, and correct according to CRN bisimulation – that the state-of-the-art implementations with more than 2-stranded inputs have. Until now we've had a few results that have all but one of those desirable properties, including one based on a novel mechanism we called coupled reconfiguration, but that depended on the physically questionable assumption that pseudoknots cannot occur. We wondered whether the same type of mechanism could be done in a pseudoknot-robust way.

In this work we show that in fact, coupled reconfiguration can be done in a pseudoknot-robust way, and this mechanism can implement general Chemical Reaction Networks with all inputs being single strands of DNA. Going further, the same motifs used in this mechanism can implement stacks and surface-based bimolecular reactions. Those have been previously studied as part of polymer extensions of the Chemical Reaction Network model, and on an abstract model level, the resulting extensions are Turing-complete in ways the base Chemical Reaction Network model is not. Our mechanisms are significantly different from previously tested DNA strand displacement systems, which raises questions about their ability to be implemented experimentally, but we have some reasons to believe the challenges are solvable. So we present the pseudoknot-robust coupled reconfiguration mechanism and its use for general Chemical Reaction Network implementations; we present the extensions of the mechanism to stack and surface reactions; and we discuss the possible obstacles and solutions to experimental implementation, as well as the theoretical implications of this mechanism.

2012 ACM Subject Classification Theory of computation; Applied computing → Chemistry

Keywords and phrases Molecular programming, DNA strand displacement, Chemical Reaction Networks

Digital Object Identifier 10.4230/LIPIcs.DNA.31.2

Funding *Hope Amber Johnson*: Supported by an NSERC Discovery Grant.
Anne Condon: Supported by an NSERC Discovery Grant.

1 Introduction

One of the more ambitious goals of molecular programming is to come up with a general framework to write programs for biocompatible molecules, possibly up to the complexity of modern silicon computers. Such a framework would likely involve an abstraction hierarchy, where human-understandable programs are written at higher levels and translated through the layers, eventually producing a set of molecules that physically execute the program. Over the past decades, one candidate pair of layers for such a hierarchy has emerged: the Chemical Reaction Network as an abstract model, and DNA strand displacement as a more physical description of molecules.

The Chemical Reaction Network (CRN) model describes abstract chemical species and the arbitrarily specified reactions they undergo. The stochastic CRN model, in particular, can compute semilinear functions in an "always eventually always" errorless sense [1, 2, 7, 13, 22, 27], and can simulate Turing machines with arbitrarily small probability of error [29]. Various extensions of the model to include polymers have been proposed, which can do Turing-universal computation much more robustly [6, 10, 19, 20, 24, 26, 32]. So a reliable and scalable general CRN implementation would be a good candidate for a general-purpose molecular computer, and an implementation of one of the polymer extensions even more so.

DNA strand displacement (DSD) refers to certain behaviors of DNA, involving parts of strands displacing bonds between other parts, that have been used to build many useful molecular devices [28]. Most such devices have been built out of simple toehold exchange motifs, which in Section 2 we call the tx_3 and tx_4 motifs. To help analyze these systems, reaction enumerators [3, 21] and formal models [16, 23] have been developed, turning DSD from a description of experimental reality into a formal system.

Many general implementation schemes for arbitrary CRNs have been designed in DSD, with various relative advantages and disadvantages [5, 8, 18, 24, 30, 31]. Figure 1 gives an example. Such a system will generally involve *signal* strands, representing the formal species of the CRN, and *fuel* complexes, assumed "always present" and consumed to drive the CRN's reactions. While some have been implemented experimentally [8, 31], even 3 species and 3 reactions in a formal CRN is enough to have problems scaling up robustly [31]. There have been attempts to address this, including a design that would be "leakless" on a theoretical level [33, 35]. However, these have not yet led to effective scalable implementations.

We have recently been trying to explore the space of possibilities in DSD, beyond the tx_3 and tx_4 motifs that make up most existing DSD systems. This is partly out of theoretical interest, and partly from a more practical motivation: CRN implementation schemes use fuels with at least 3 strands, and have issues scaling up [31]. Meanwhile seesaw gates and some other systems, though they can't implement arbitrary CRNs, have been made with 2-stranded inputs and scaled up well to larger systems [9, 25, 34]. Therefore we investigated whether arbitrary CRN implementations could be made with 2-stranded inputs (i.e., fuels and signals), with an additional set of desirable conditions: reversible, systematic implementation, uses $O(1)$ orthogonal toehold domains, correct according to modular CRN bisimulation, and built out of bimolecular reactions [16]. We found two systems that can implement arbitrary CRNs, but each with one undesirable condition: one uses $O(n)$ toehold domains, while one requires trimolecular reactions [18]. Both of these conditions would cause issues with experimental implementations. This was not satisfying, so we kept searching.

Existing DSD implementations often implement $A + B \to \dots$ logic in a way that requires a fuel with at least 3 strands: one strand to bind the reactants A and B, one strand to be released in the reaction that binds A, and one to be released in the reaction that binds B [30, 31]. The mechanism shown in Fig. 1, for example, represents the binding of A by

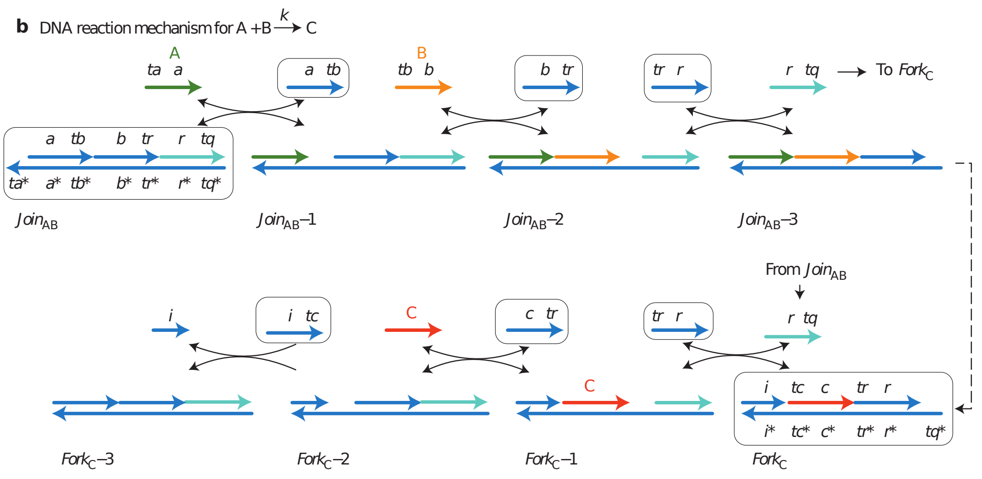


Figure 1 An example CRN implementation with DSD [8]. The reaction $A+B \to C$ is implemented using two 4-stranded fuels.

opening a shared toehold that allows the binding of B, but this required kicking off an A-specific strand. Recently we found a mechanism we called *coupled reconfiguration*, where instead of binding one strand and releasing another, two strands can reconfigure each other and split apart. Thus the A signal strand can reconfigure into an "off" state, and the fuel strand can reconfigure into a "bound A" state, allowing us to implement arbitrary CRNs with 1-stranded inputs [15]. However, our mechanism depended on a common assumption in DSD models that certain structures called *pseudoknots* don't form, and would have undesired pathways if pseudoknots could occur. Specifically, it represented the binding of A by changing which of an A_1 or A_2 domain was bound to its complement, such that the interaction with B would be non-pseudoknotted only in the "bound A" state. While our mechanism was great on paper in the pseudoknot-free model, that model is based not on the belief that pseudoknots actually don't occur in reality, but the belief that they're too complex and we don't want to deal with them. If implemented in reality, it's likely that the mechanism would simply fail due to pseudoknotted undesired pathways. We again found this unsatisfying, and wondered if coupled reconfiguration could be done without depending on the no-pseudoknots assumption.

Here in Section 3 we present a mechanism we call CR_4, based on 4-way branch migration, that does coupled reconfiguration without relying on the no-pseudoknots assumption. In particular, the CR_4 mechanism changes which toeholds are open and closed, meaning we can compose A and B with the same shared toehold logic used in Fig. 1 and most similar systems; but as a coupled reconfiguration mechanism, it does so without requiring an additional strand. In Section 4 we show how this mechanism can implement arbitrary CRNs with 1-stranded inputs, with all the other desirable conditions as well. The CR_4 mechanism is based on a $2tx_4$ motif, which turns out to be useful for implementing polymer CRNs, and in Section 5 we give mechanisms for stack push and pop reactions and for bimolecular surface CRN reactions, all with 1- and 2-stranded inputs. These mechanisms could be slotted in to the existing theory for stacks [24] and surface CRNs [26] without changing the higher abstraction layers, if desired. Our mechanisms are sufficiently far from existing experimentally tested mechanisms that there will likely be challenges implementing them, but we suspect it would be possible. In Section 6 we give some speculation on what challenges may arise and how to overcome them, as well as further theoretical implications.

2 Formal DSD, motifs, and correctness

We work with a formal model of DNA strand displacement systems, and are trying to explore the limits of that model. The more abstract Chemical Reaction Network model is both an important use case for DSD systems and a good way to test the computational power of a class of DSD systems. In this section we give some definitions of Chemical Reaction Networks, DSD systems, and the definition of a correct implementation that we use. Some more formal definitions were omitted for space, and can be found in Appendix A.

▶ **Definition 1.** *A* Chemical Reaction Network *is given by a finite set of (abstract) species, with a finite set of specified reactions between those species. Reactions are pairs (R, P), where R and P are each a multiset of species. The* states *of a CRN are then all multisets over the set of species, and transition between each other according to the reactions.*

Often reactions will be given as a triple (R, P, k), written $R \xrightarrow{k} P$, where k is a rate constant. For our purpose we care only what can happen, not how fast it can happen, so we use only $R \to P$. The reversible reaction $R \rightleftharpoons P$ means $R \to P$ and $P \to R$.

▶ **Definition 2.** *A* strand *is a sequence of* domains*, listed from "5' end" to "3' end". Here a domain is simply one of some finite set of domains, such that for each domain x there is a complement x^*, where $(x^*)^* = x$. In general and in our work $x^* \neq x$, but this is not a requirement. Each x is defined to be either* long *or* short*, with x^* short if and only if x is. We often refer to short domains as* toeholds*.*

A complex *is a finite multiset of strands with a list of bonds between pairs of their domains, such that each domain x is either unbound or bound to exactly one x^*, and the resulting graph is connected.*

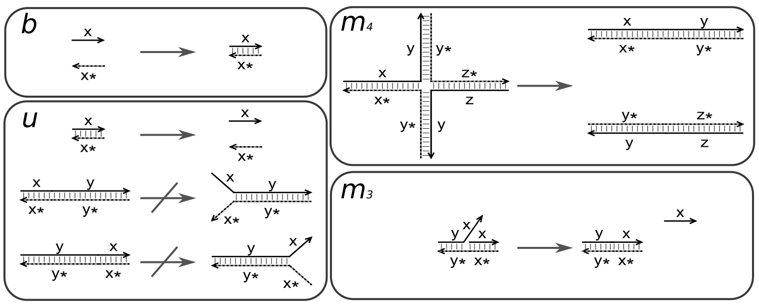


Figure 2 The four basic steps that define our DSD model. Modified from [16].

▶ **Definition 3.** *The* anchored, pseudoknot-allowed DSD model *is the following four basic steps, illustrated in Figure 2:*

- Binding *(b): Two unbound domains x and x^* can bind.*
- Toehold unbinding *(u): Two domains (x, x^*) bound to each other can unbind if x is short and the bond (x, x^*) is not* anchored *by an adjacent bond as shown in Figure 2.*
- Three-way branch migration *(m_3): Given three domains, one unbound instance of x and a bound pair (x, x^*), the two instances of x can exchange which is bound to the x^*, if the resulting bond is* anchored *by an adjacent bond as shown in Figure 2.*
- Four-way branch migration *(m_4): Given four total domains, two bound pairs of (x, x^*) for the same x, the two pairs can exchange which x is bound to which x^*. This can happen only if both of the resulting bonds are* anchored *by adjacent bonds as shown in Figure 2.*

Basic steps are meant to be applied to complexes, producing reactions (also called steps) whose reactants and products are complexes. A step between complexes is *reversible* if its reverse is also one of the steps. b is reversible if the resulting bond meets the conditions for u (short domain and not anchored); u is always reversible by b; m_3 is reversible by m_3 if the initial (x, x^*) bond was anchored; and m_4 similarly to m_3.

Often when working with DSD systems we want to require that all complexes involved are *non-pseudoknotted.* This greatly simplifies analysis without too much loss in potential, as pseudoknots are not well understood and thus not reliable for use in engineered systems.

▶ **Definition 4.** *A complex is* non-pseudoknotted *if there is some ordering of the strands such that, for any bonds (x, x^*) and (y, y^*), the x and x^* are either both between y and y^* or both outside them. If such an ordering exists, it is unique up to circular permutation [12].*

The anchored, pseudoknot-free DSD model *is the anchored, pseudoknot-allowed model with the following modification: the b step can only happen if the resulting complex is non-pseudoknotted. (The u, m_3, and m_4 steps, by their construction, cannot make a non-pseudoknotted complex pseudoknotted.) This model is equivalent to Peppercorn's model with the* `--reject-remote` *option [3].*

▶ **Definition 5.** *A* DSD system *is a DSD model and a set of initial complexes. The* enumerated CRN *of a DSD system is the (smallest) CRN where every initial complex is a species, every basic step applicable to species is a reaction, and the products of every such step are species.*

The process of generating the enumerated CRN is often called *reaction enumeration*, and can be done automatically by tools such as Visual DSD [21] and Peppercorn [3].

While many sequences of steps are in theory possible, the designs in this paper are built out of the same sequence of steps used repeatedly. We often call these *motifs* [18]. Figure 3 shows the tx_4 (toehold exchange with 4-way branch migration) and $2tx_4$ motifs we use.

Following models such as Peppercorn, we often assume that once two complexes combine and start interacting, no third complex can bind to the result until it's finished all meaningful unimolecular steps. This is based on a low-concentration assumption where bimolecular steps are slow relative to unimolecular steps, making that assumption realistic. Peppercorn calls this a "condensed reaction", and gives a proper definition [3]. We use this to implicitly exclude certain complicated and unlikely undesired pathways, but don't mention it much.

When a DSD system is made to implement CRNs, its initial complexes will generally be made of *signals* and *fuels.* Signals represent the formal species of the CRN, while fuels are assumed to be "always present" (left as an exercise to the experimentalist) so they can drive the reactions they're meant to enable. Once an implementation CRN is enumerated from the DSD system, the fuels are removed (so e.g. $A + f \to B$ becomes $A \to B$ for f a fuel), and we consider an implementation correct if the result meets the conditions of CRN bisimulation [17]. These are, approximately, "anything that can happen, should" and "anything that should happen, can", where "can" refers to the DSD implementation and "should" to the formal CRN. When we discuss the number of strands in the *inputs* of a CRN implementation, we mean the signals and fuels. Experimentally, these are the complexes that have to be created externally, one way or another, to make the system run.

One concept we wish to explore here is *pseudoknot-robustness*, defined in terms of the correctness of CRN implementations. Most DSD systems don't involve pseudoknots, and many analysis algorithms [3, 4, 12] assume they can't happen. However, that assumption is based less on a belief that they're actually physically impossible, and more on a desire not to deal with them. We are therefore interested in systems that are correct whether pseudoknots are possible or not. Formally, this is defined by checking the "anything that can happen,

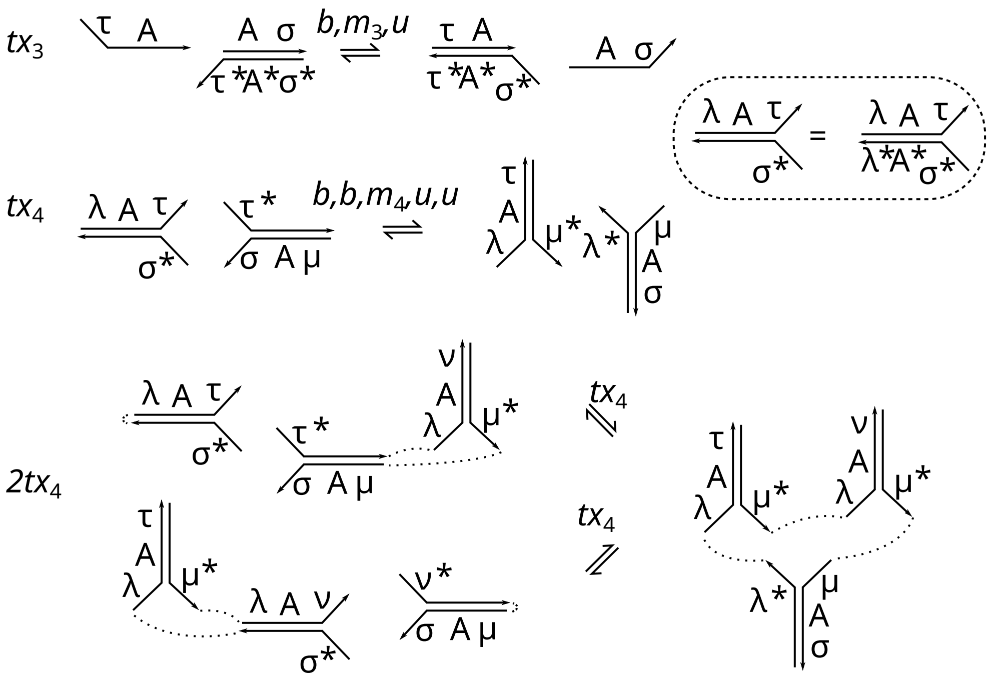


Figure 3 The tx_3 and tx_4 motifs, and further abstractions. The mechanisms in this paper are built on the sequence of two tx_4 motifs shown, so we give an abstracted notation that is useful to present them. The tx_3 motif used in most existing DSD systems is presented for comparison, but we won't use it further. Our typical use of the $2tx_4$ motif involves single strands and 2-stranded complexes, where strand connections (dotted lines) make the intermediate state a closed loop as shown. This gives the forks in question nothing to do other than reversing or completing the motif.

including pseudoknots, should" condition in the anchored, pseudoknot-allowed model, and checking the "anything that should happen, can even without using pseudoknots" condition in the anchored, pseudoknot-free model.

3 The four-way coupled reconfiguration mechanism

Our (first) goal with this mechanism is to implement generic $A + B \rightleftharpoons C + D$ logic, which is sufficient to implement arbitrary CRNs. However, the method we use is built out of smaller parts, namely the four-way coupled reconfiguration (CR_4) mechanism; and is best explained by first defining the CR_4 mechanism and how that mechanism can be abstracted into inputs and outputs, then showing how the general $A + B \rightleftharpoons C + D$ implementation is built out of the abstracted CR_4 mechanism. So Fig. 4 shows the configurations that a CR_4 mechanism reconfigures between, and how the toeholds serve as abstractable inputs and outputs; Fig. 5 shows the actual CR_4 mechanism; and Fig. 6 shows how two CR_4 mechanisms can be connected by a shared toehold. Ultimately in Fig. 7 we end up implementing $A+B \rightleftharpoons C+D$ as $A + B \rightleftharpoons int_r$, $C + D \rightleftharpoons int_r$ for int_r an intermediate specific to this reaction; this is the same strategy implemented in e.g. the tx_3-based CRN implementation [5, 8] in Fig. 1.

In our previous work, we designed a tx_3-motif-based mechanism where two strands come together, reconfigure each other, and split apart; such that neither strand can reconfigure on its own, and when the two strands come together, they can split apart only if both reconfigure or neither does [15]. We named this property *coupled reconfiguration*, and it allowed us to design arbitrary CRN implementations using only 1-stranded inputs, as opposed to 3- or more-stranded complexes. To our knowledge, this was the first DSD mechanism designed with this property; others had relied on different combinations of strands to get different configurations. However, it was very reliant on the assumption that pseudoknots couldn't happen, with the strands able to reconfigure on their own via pathways involving pseudoknots. This makes the mechanism theoretically interesting, but likely not to work in reality, and thus unsatisfactory. So the core of the result of this paper is a CR_4 mechanism, based on the $2tx_4$ motif, that does coupled reconfiguration in a pseudoknot-robust way.

Within the pseudoknot-robust anchored model, there are two ways we're aware of for one tx_3 or tx_4 motif to depend on a previous one. (That is, for two motifs in a sequence to be meaningfully in that order, as opposed to two independent motifs that just happened to happen in that order.) First, specific to tx_4 motifs, one (tx_3 or tx_4) motif can put together the combination of two toeholds necessary for a (second) tx_4 motif to happen. For example, in the $2tx_4$ motif in Figure 3, the first tx_4 motif puts together the (μ, λ^*) combination necessary for the second to happen. This generally involves both motifs acting on the same long domains, so it's good for creating dependencies within the same long domain identity, but not between independent long domain identities. In the CR_4 mechanism in Figure 5, all the dependencies between the three $2tx_4$ motifs are of this type.

Second, the two motifs can have a shared toehold. By "shared toehold" here we mean a toehold flanked by two long domains, where a motif acting on one long domain opens the toehold, allowing it to be used in a motif acting on the other long domain. The opening of toeholds in most previous DSD implementations is of this type, as in Figure 1.

The CR_4-based CRN implementation will use both. Within a single long domain identity associated with a given formal species, a specific arrangement of toehold identities allows a sequence of $2tx_4$ motifs between a "gate" strand and a "signal" strand in an "on" configuration, after which the signal strand is in the "off" configuration and the gate strand has also reconfigured. The arrangement of toeholds ensures that at any point in the process there are at most two reactions available, the intended forward and reverse reactions, even if pseudoknots are allowed. The gate strand will have different toeholds open and closed before and after the reconfiguration, which allows composition if any of those toeholds is shared with another copy of the CR_4 mechanism. Opening and closing shared toeholds with coupled reconfiguration means we don't need the extra strands that most DSD CRN implementations use. While we use 7 (or 6, or possibly less with optimization) orthogonal toehold identities, we can reuse those same 6-7 sequences throughout, which is what we mean by $O(1)$ toeholds.

For one motif to enable another motif via a shared toehold, the first motif should transform the toehold from "unavailable" to "available", from a state where it can't do the second motif to a state where it can. In the pseudoknot-robust paradigm, "can" is measured in the pseudoknot-free model, so a toehold is available if it is both unbound and not nested inside another bond. But "can't" is measured in the model with pseudoknots allowed, so a toehold is unavailable if it is bound. A toehold that is unbound but nested inside another bond can't be pseudoknot-robustly said to be either available or unavailable.

Our mechanisms involve various $2tx_4$ motifs with the same long domain identity eventually opening up a shared toehold, which will then enable more motifs with a different long domain identity. Figure 4 shows, for 3 pairs of alternating long domains A and A^*, the possible configurations, and the availabilities of their toeholds. The "one nested" configurations are particularly useful. A strand that switches between them will have the toehold pair between domains 2 and 3 switch availabilities in one direction, while the pair between 4 and 5 switches in the other. We use 3 pairs because 2 alternating copies of a long domain and its complement have only two configurations, and not enough toeholds that change availability.

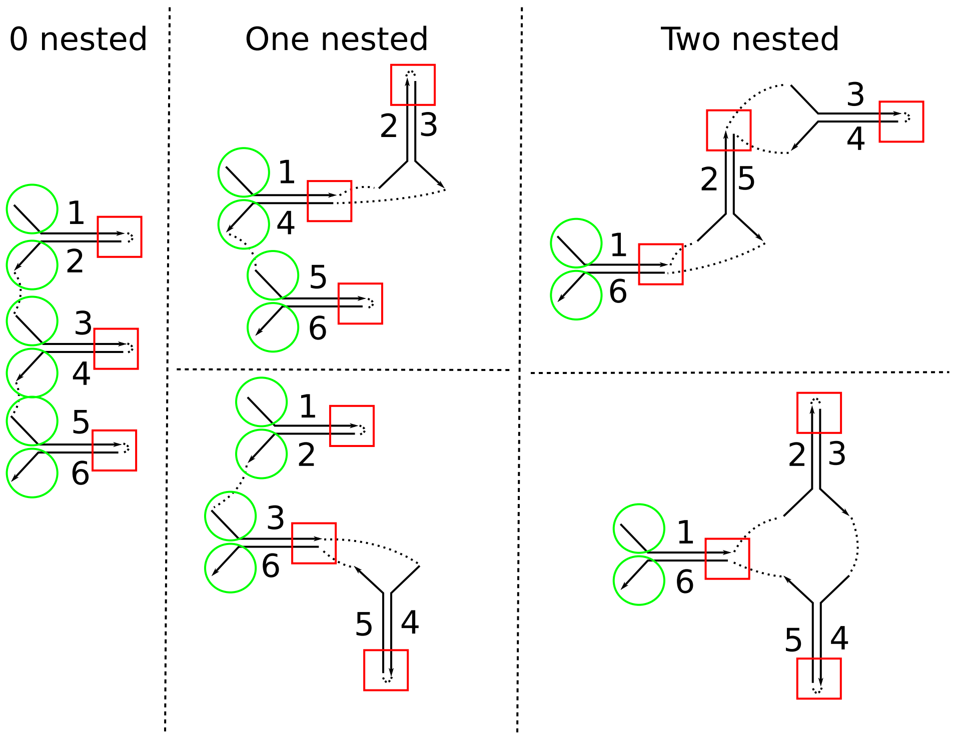


Figure 4 Non-pseudoknotted configurations of a strand with 6 long domains alternating between A and A^*, each flanked by toeholds with open toeholds facing "out". For this purpose we number the domains 1 through 6, odd numbers being A, even numbers A^*. Of the 6 possible pairings between two sets of 3 elements each, one of them (1-4, 3-6, 2-5) is pseudoknotted; the other five are shown. Note in each configuration which toeholds are both unbound and not nested (green circles), and which are bound (red squares). The CR_4 mechanism will cause each of its strands to switch between two of these configurations, and its meaningful effect is to turn some toeholds "on" and others "off".

We can then describe the CR_4 mechanism, shown in Figure 5. The mechanism involves two strands, one ("left strand", purple) reconfiguring between the two one-nested configurations in Figure 4, one ("right strand", blue) reconfiguring between a one-nested configuration and the linear two-nested configuration. So while the left strand has a pair of toeholds that changes from available to unavailable and a pair that does the reverse, the right strand only has a single toehold that changes from available to unavailable and none that do the reverse. There are multiple regions, indicated by dotted lines, that can contain any sequence without affecting the mechanism unless that sequence somehow interacts with the domains involved in the mechanism. This will be particularly useful for composing copies of the CR_4 mechanism.

The CR_4 mechanism as presented is physically reversible, with each step as well as the overall mechanism being fully reversible in the model. It can be made irreversible in either direction by removing a single toehold. Removing either of the two ω^* toeholds on the left strand will make the reaction irreversible in the direction towards which that ω^* ends up unbound. (Removing the ρ^* or ν^* toehold can also do so, but those are used for composition.) Similarly on the right strand, in each direction one of the three σ toeholds changes from bound to unbound, and removing it will make the reaction irreversible in that direction.

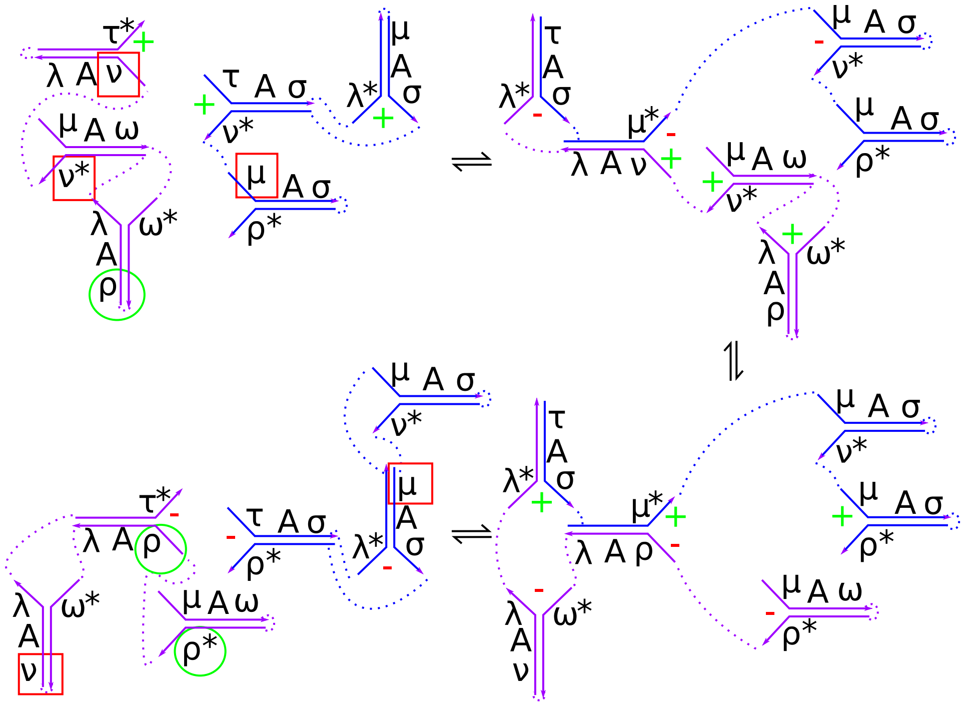
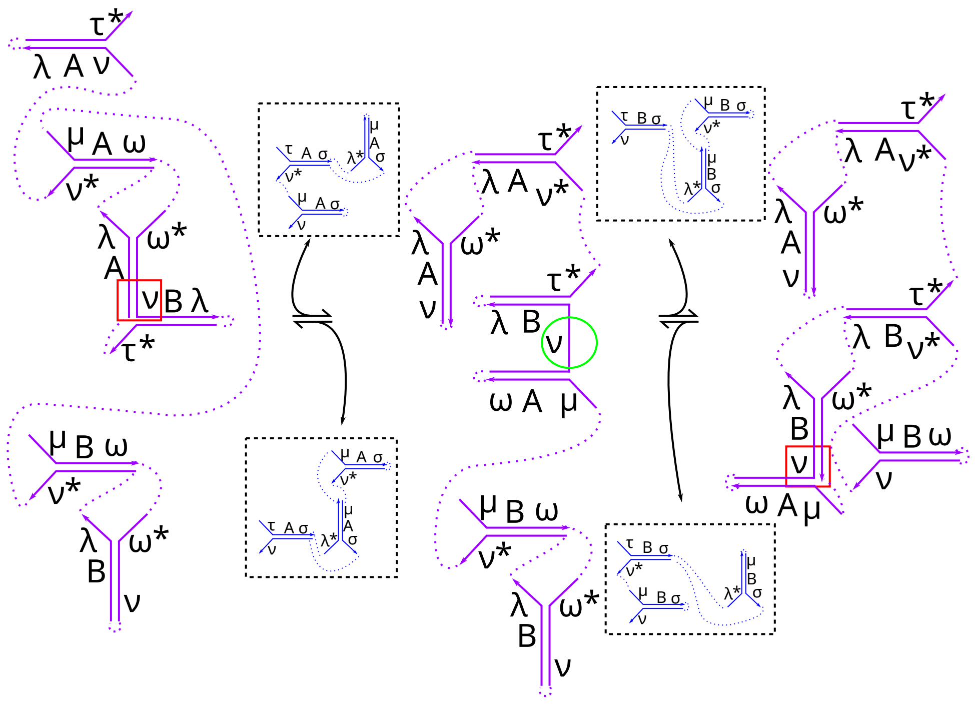


Figure 5 The pseudoknot-robust coupled reconfiguration CR_4 mechanism. Each step is a $2tx_4$ motif as in Figure 3. Dotted lines indicate strand connections that can contain arbitrary sequences; note that the reaction as drawn starts and ends with two single strands. + and − indicate forks that interact in the forward and reverse directions, respectively. Green circles indicate toeholds that are available at the end and bound at the beginning, for composition with other reactions. Red squares do the same for the opposite direction. Note that only one of the two strands has both.

4 Implementing CRNs with four-way coupled reconfiguration

To implement CRNs, we first need to figure out how to compose modules of the CR_4 mechanism. Ideally, we would like what we call a *systematic, $O(1)$ toeholds* implementation, which we defined in our previous work [16]. This means that the "signal strands" for formal species e.g. A and B have the same structure and the same toehold domains, differing only in the identity of their long domain(s), which should be species-specific. To get interaction between independent long domains we want to use a shared toehold, where one of the toeholds that changes availability in one mechanism is the same as one of the ones that changes availability in the other. Given the properties of the mechanism, we need to use the left strand for this, and equate the ρ^* of one mechanism with the ν of the other. So while Figure 5 had ρ be a unique toehold, for the CRN implementation $\rho = \nu^*$ and $\rho^* = \nu$; the CR_4 mechanism still works with this replacement. The composition is shown in Figure 6.

Now that we can connect multiple CR_4 modules, we can use this to make CRN implementations. Similar to our previous mechanism [15], we use left strands as gates and right strands as species strands, where one of the two configurations of a species strand is the "on" state and the other is the "off" state. The on state of a species strand represents the actual species, while the off state is a fuel that represents nothing, unless something turns it on. Then a gate strand will in sequence turn each reactant species off then turn each product species on, each time reconfiguring itself in a way that enables the next interaction.



■ **Figure 6** A strand (purple) with two copies of the coupled reconfiguration mechanism from Figure 5 connected by a shared toehold. This strand with the shared toehold open (green circle) can interact with the counterpart (blue) of either mechanism, but only with one of them; either reaction will close off the toehold needed for the other (red squares). The toehold ρ in Fig. 5 is given the sequence of ν^* to make this connection work.

In contrast to the previous mechanism, this one has a polarity issue: one could not, for example, equate the ρ of one CR_4 mechanism in Figure 5 with the ν of another, because the ρ is 5' of a long domain in its mechanism while the ν is 5' of a different long domain in its mechanism. Nor can one connect a CR_4 mechanism that turns a B off with one that turns a C on, if signals B and C are from the same template. So instead of implementing $A + B \rightleftharpoons C$ in one gate, we implement $A + B \rightleftharpoons int_r$. Here int_r is an intermediate, specific to reaction r, and while the species strands use the one-nested state as on and the linearly nested state as off, int_r uses one-nested as off and linearly nested as on. Figure 7 shows this gate. The intermediate strand would then be shared with a second gate for the products, so $A + B \rightleftharpoons C + D$ would be implemented as $A + B \rightleftharpoons int_r$ and $C + D \rightleftharpoons int_r$. For irreversible reactions, ω^* toeholds should be removed from the gates to make the int_r module of the reactant gate, and all modules of the product gate, irreversible as described in Section 3.

We coded this implementation as a translation scheme in the Nuskell language [4], and ran it in that program to check whether it works. In fact it does on two simple CRNs we tested. At least, it works in the pseudoknot-free model; Nuskell does not calculate pseudoknots, and thus cannot check whether our system is pseudoknot-robust. In general, our argument for correctness is that the same combination of two open toeholds and a long domain does not appear in our system except in the desired reactions, namely the forward and reverse reactions at any given step. Thus no m_4 step can happen except for the intended tx_4 motifs. This can be observed from the figures.

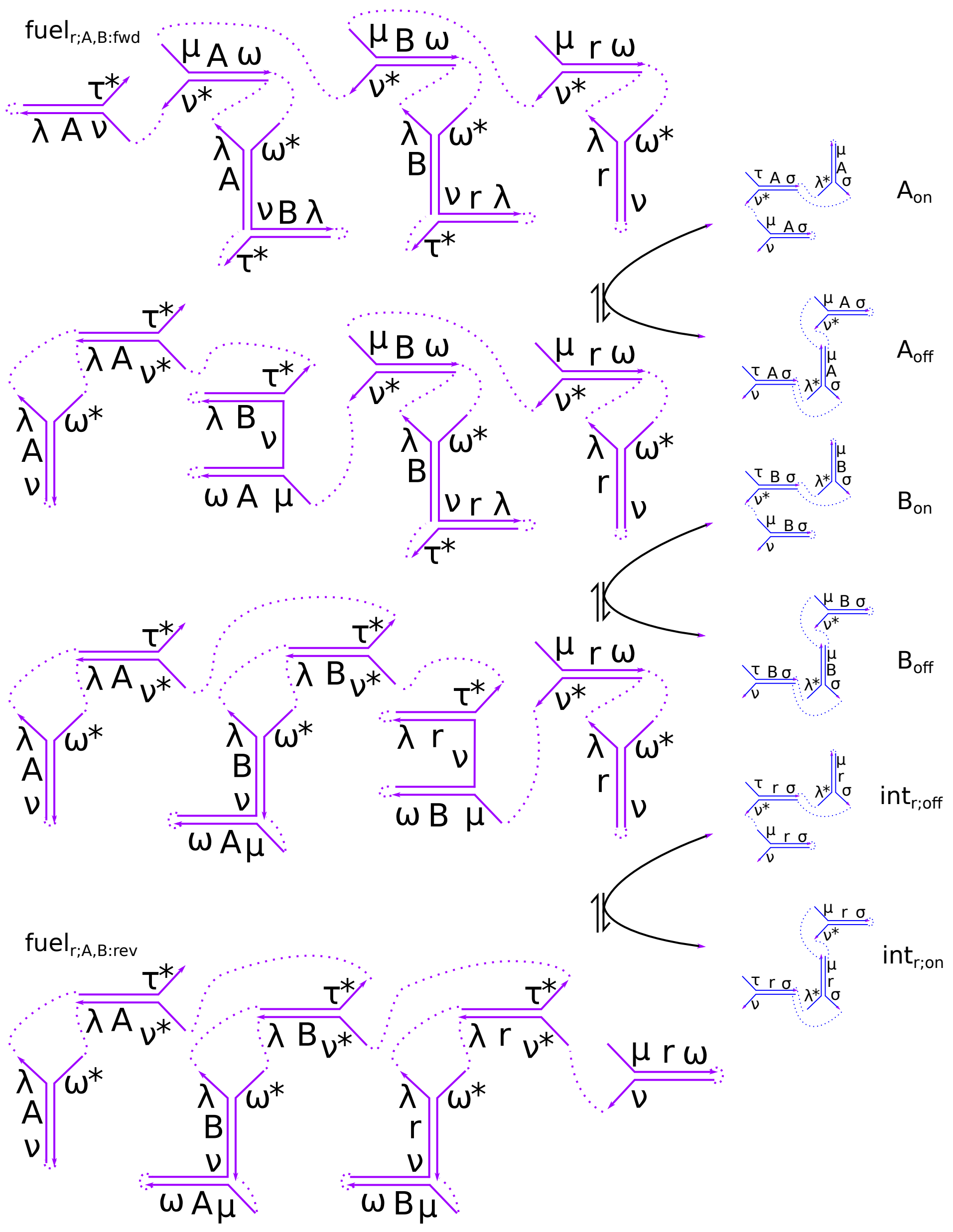


Figure 7 An implementation of a CRN reaction using shared toeholds as shown in Figures 5 and 6. This figure shows the reactants half of $A + B \rightleftharpoons \dots$; for a reversible reaction $A + B \rightleftharpoons C + D$, the products half is an exact copy of the mechanism with A and B replaced by C and D. A single gate strand (purple) reconfigures the reactant strands (blue) to their "off" states, then converts an intermediate with a reaction-specific long domain r to its "on" state. That intermediate can then interact with the product gate.

5 Polymer extensions: stacks and surface CRNs

While well-mixed CRNs can do many interesting tasks, they are provably less powerful than classical computers [2, 22, 27, 29]. Adding some form of geometry, such as unbounded polymerization or a surface grid, makes up the difference in computational power. To that end, there have been designs for DSD implementations of polymer CRN-like systems, such as stack machines [24] and surface CRNs [10, 26], and some general polymer models [6, 19].

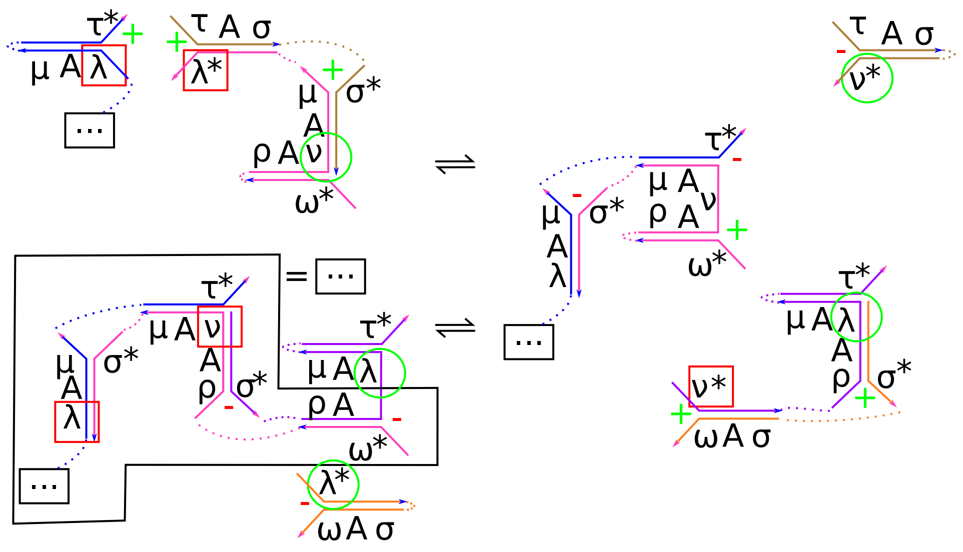


Figure 8 An implementation of stacks (reversible push and pop reactions) using $2tx_4$ motifs. The ... box represents the rest of the stack; note that the end state has the same pattern as the start state, with an additional unit on the stack, drawn enclosed in the larger box. All four non-stack complexes have toeholds composable with the previous CRN mechanism (red squares, green circles).

It turns out that the $2tx_4$ motif is convenient for polymerization mechanisms: of the three sequences of 2 long domains (plus flanking toeholds) involved, the motif takes one of them from bound to the second to bound to the third. This can be used to add or remove strands from a polymer. For example, Figure 8 shows a mechanism for adding an additional unit to a linear stack polymer. If necessary this mechanism can be made irreversible by removing appropriate σ^* toeholds, but this is not usually desirable for stack machines. It is worth noting that this mechanism is not, technically, a coupled reconfiguration mechanism, since there is no strand or complex that has multiple configurations for the same set of strands.

The four non-stack complexes could be considered fuels, but if instead their presence is controlled by some other system, they can control whether or what gets added to the stack. For example, if their composition toeholds are shared with a species strand from the well-mixed CRN mechanism, that would make a CRN augmented by controllable stacks. Note that the toehold identities here have no correlation to those in Figure 7, so they can be renamed as necessary to be compatible with that mechanism. The stack machine implementation from [24] is, on the formal level, a CRN augmented by controllable stacks, and their implementation could be replaced by ours with no change to the formal augmented CRN. Thus our mechanism can make stack machines with only 2-stranded input complexes.

Another type of polymer system is surface CRNs, given in the same notation as CRNs but where the species are understood to occupy points on a grid [26]. Then a reaction $A + B \to C + D$ means that an A and B at adjacent points to each other interact and become a C and D in the respective same points on the surface. Implementing this generally requires a "replacement-in-place" mechanism, where a species strand bound to a generic base strand is identified by an incoming fuel, and then replaced on the same base strand by a different strand. Though complex, this is possible with the $2tx_4$ motif, as shown in Figure 9. Specifically, that figure shows the $B \rightleftharpoons I_r$ half of a $B \rightleftharpoons C$ reaction; the $C \rightleftharpoons I_r$ half is symmetric. Multiple involved strands have toeholds suitable for composition, so the replacement could be controlled by e.g. an A species in a well-mixed CRN.

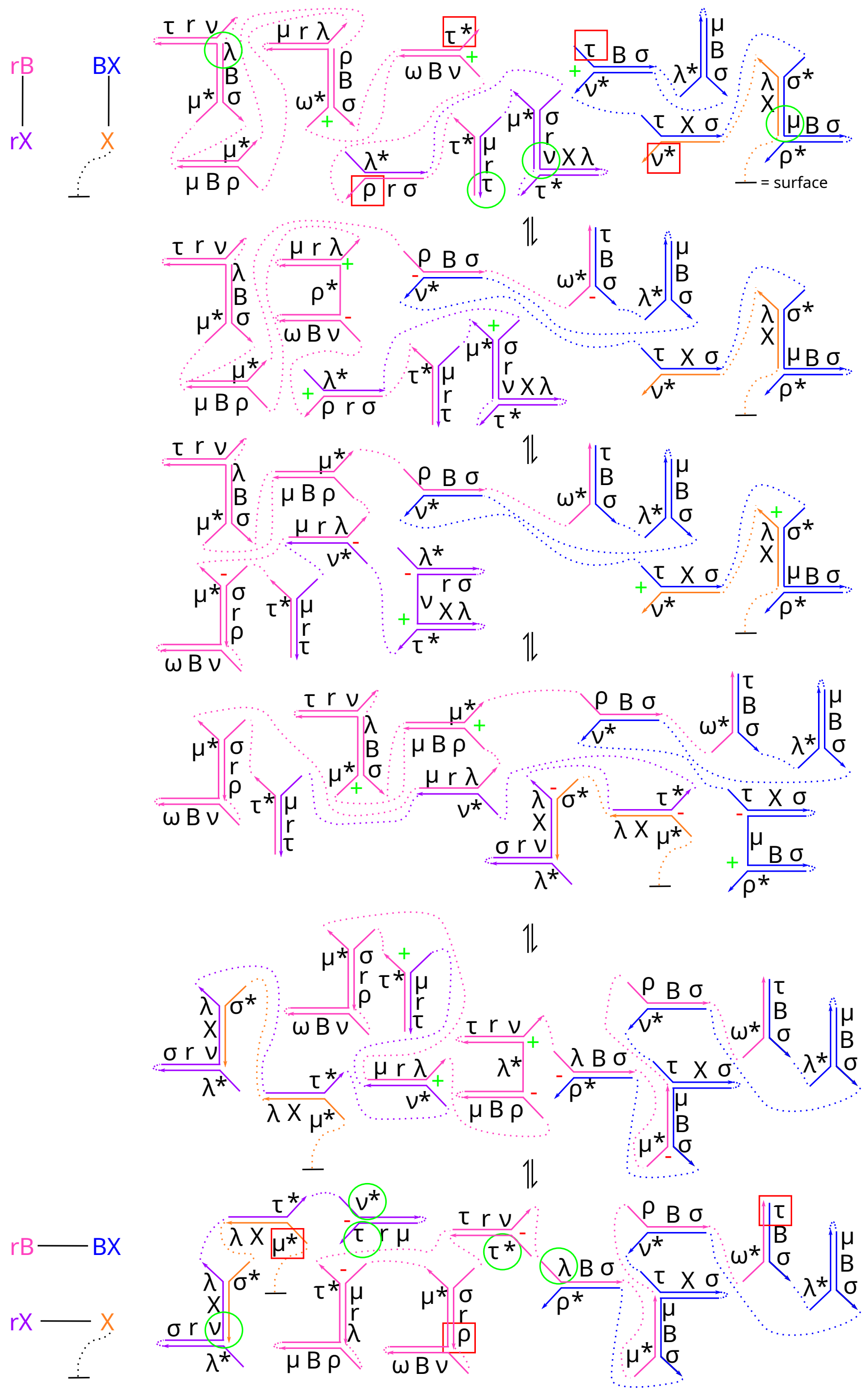


Figure 9 A replacement-in-place mechanism that can be used for surface CRN implementations. r is a reaction-specific domain, X is universal. The goal is to replace what's bound to the base strand, connected to some surface. Shared toeholds allow interaction between B, r, and X domains.

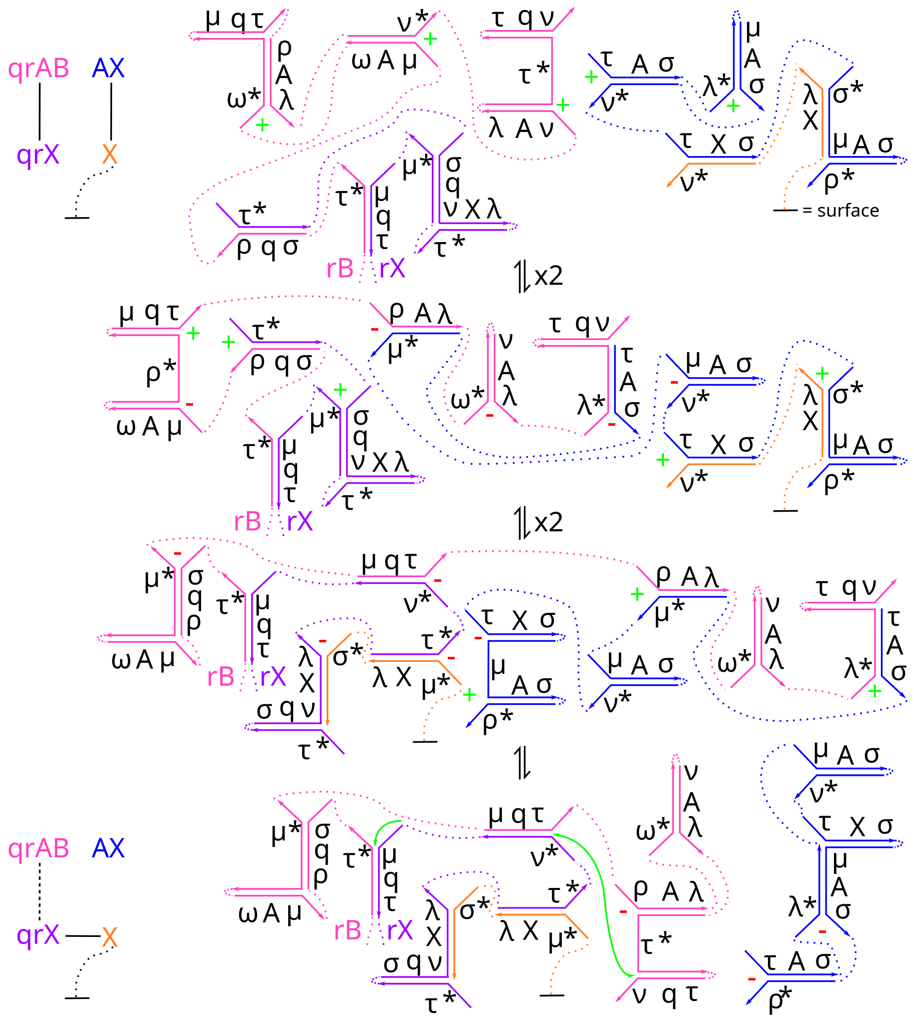


Figure 10 A replacement-in-place mechanism for the first reactant of a bimolecular surface CRN reaction. This mechanism can be best understood as a combination of those from Figures 5 and 9, so some steps were skipped in this figure. The rB and rX ... indicate the left reactant complex shown in Figure 9. A reversible $2tx_4$ motif within one product (green arrows) can separate the displayed parts of the $qrAB$ and qrX strands, allowing that reaction with the second reactant.

Strands in Figure 9 are identified by their long domains: the species-specific domain B, the reaction-specific domain r, and the universal attachment domain X. The rX strand has no species-specific domains, so it can interact with the reactant B or the product C. The rB and (not shown) rC strands then mediate between the rX strand and the species strands.

To implement the bimolecular $A + B \rightleftharpoons I_r$ surface CRN half-reaction, one could just put together two copies of this mechanism. However, the rX strand would then either have to be in a 3-stranded fuel complex with the rA and rB strands, or start out with the rA strand and have the rB strand enter later. But the second case would not let it remember, halfway through a reaction, whether it's interacting with the reactants or the products; and what should be $A + B \rightleftharpoons C + D$ could become $A + B \rightleftharpoons A + D$ or any of a few other combinations. Instead, we use a variant mechanism where a $qrX - qrAB$ 2-stranded fuel causes the A strand to be released on its own, rather than attached to the $qrAB$ strand. (We use q as a second reaction-specific long domain to minimize crosstalk potential.) This mechanism is shown in Figure 10. The τ^* domain at the 5' end of the $qrAB$ strand is a shared toehold, the same as the τ^* at the 3' end of the rB strand as shown in Figure 9. Thus in the full reaction, a $qrAB - qrX$ fuel will find an $AX - X$ complex on the surface, displace the A according to Fig. 10, at which point the newly opened half of those strands will interact with and displace a neighboring $BX - X$ complex on the surface, releasing a $qrAB - BX$ reverse fuel complex and leaving a $qrX - X$ complex on the surface to receive the products.

In our previous work on theoretical polymer systems, we showed that all "single-locus" polymer networks could be implemented by the following five types of reaction schema [19]:

$$\begin{aligned} *_1AB*_2 &\to *_1CD*_2 \\ *_1A*_2 + B &\rightleftharpoons *_1C*_2 \\ F*_1 \rightleftharpoons *_1 \quad &, \; *_1 \rightleftharpoons *_1E \\ *_1 + *_2 &\rightleftharpoons *_1I*_2 \end{aligned}$$

Of those, the stack mechanism in Fig. 8 implements $*_1 \rightleftharpoons *_1E$ and, with modifications, $F*_1 \rightleftharpoons *_1$. The surface mechanism in Fig. 9 implements $*_1A*_2 + B \rightleftharpoons *_1C*_2$ when combined with the well-mixed CRN mechanism, and with the mechanism in Fig. 10 implements $*_1AB*_2 \to *_1CD*_2$ (reversibly or irreversibly). The last of those reactions is only necessary for reactions that look like it, but might be doable by a variant of the stack mechanism.

6 Discussion and further work

Experimental considerations

The mechanisms we presented work in the formal model of DSD we use, but being different from previously tested DSD mechanisms, leaves open the question of whether they would work in physical DNA molecules. There are four specific questions possibly interesting as general questions about DSD, beyond this mechanism: length of the mechanism; fuel synthesis; internal toehold unbinding and binding; and whether m_4 steps require two matching toeholds.

Like our previous work [15], this mechanism involves a much longer sequence of steps between two complexes coming together and the result eventually separating than most experimentally tested DSD systems. The more complex a system is, the more that can go wrong, even if we don't know what. So we'd like to know whether DSD mechanisms are robust enough for a reaction of this length. Relatedly, we argued that 1- or 2-stranded input complexes would be easier to synthesize than 3- or more-stranded complexes, but we gained that at the expense of making those strands much longer, which also raises concern.

In DSD systems in general, the usual way to make the fuel complexes is to put the strands together in solution and anneal them [31]. This requires any particular multiset of strands to have one minimum free energy configuration, the desired fuel. In a coupled reconfiguration mechanism, the same strand has multiple "equal"-energy configurations, only some of which are fuels. This is likely solvable; in our previous coupled reconfiguration work [15], we speculated about co-transcriptional folding mechanisms. In this mechanism, there are regions where it doesn't matter what's in them; it might be possible to add sequences in those regions that slightly bias an annealed strand to form in one configuration over the others, but gently enough that the coupled reconfiguration mechanism can still complete.

In the DSD model we use, any domain is either short (toehold) or long; toehold domains can unbind whenever they're not anchored, then bind again to a different copy of their complement. In experimental use, "toehold" means a domain whose binding energy is roughly equal to the energy from entropy of complexes dissociating, in the current conditions. Toeholds unbinding and re-binding all within a single complex is, to our knowledge, not studied, but our mechanism depends on it. We suspect that, even if internal toehold unbinding doesn't work with 6-base toeholds, there will be some length of toehold that does behave the way we require. Part of testing this mechanism would involve finding that length.

In the anchored model, we assume an (m_4) step can only happen if the branch migration domains are flanked by two pairs of bound domains (usually toeholds); if only one side or none are bound, the m_4 step does not happen. Our mechanism relies heavily on this, as we use pairs of toehold identities to carry information; in Figure 5, for example, there are steps where a (ρ, μ^*) pair needs to interact with either of two (μ, ρ^*) pairs, but not the (μ, ν^*) pair next to them. But again, this isn't well tested. Dabby's experiments show a difference in rates between an m_4 step with both sides bound and an m_4 step with one side bound and the other nonexistent [11], which is promising. But in our system the second side will have two non-complementary toeholds, which in practice will have nonzero interaction. And our system needs 6-7 orthogonal toeholds. Again, we suspect that even if 6-base toeholds don't have the desired behavior, there will be some design of toeholds that will.

Theoretical implications

We have for a while been wondering if a general CRN implementation with DSD using only 2-stranded inputs, with a certain set of other desirable conditions, was possible. Those conditions were: the system is physically reversible, is a systematic implementation, has $O(1)$ toeholds, is made of macroscopically bimolecular reactions ("condensed reactions" as defined in [3] or [16]), and is correct according to modular CRN bisimulation. We define those conditions properly in our previous work, where we also prove that with the additional condition of "no remote toehold m_3 steps are ever possible", no such scheme can exist [16]. (The middle of a $2tx_4$ motif involves toeholds unbinding and binding to a different copy of their complement, which is equivalent to a remote toehold m_3 step. We intended that to define m_4-only systems, but the $2tx_4$ motif shows that the condition was too strong.)

We have a number of examples of general CRN implementations that meet all but one of those conditions: the state-of-the-art CRN implementations use 3- or more-stranded fuels [5, 8, 24, 30], while we have come up with a scheme that uses $O(n)$ toeholds and one that relies on macroscopically trimolecular reactions [18]. Our previous m_3-based coupled reconfiguration mechanism meets all of those conditions [15], but relies on the assumption that pseudoknots physically don't happen. We found this questionable, and defined a new desired condition, pseudoknot-robustness. This mechanism meets all of those conditions including pseudoknot-robustness, closing the question of whether such a mechanism is possible.

There are still interesting directions to go from this on the theoretical level. Intuitively, there still seems to be a difference between 2-stranded and 3-stranded DSD input complexes: our 2-stranded mechanisms require long sequences of steps for $A + B \rightleftharpoons \ldots$ logic, which 3-stranded mechanisms can do with two tx_3 [30] or tx_4 [14] motifs. It might be possible to explain why in formal terms, or discover a more efficient 2-stranded mechanism. Relatedly, the coupled reconfiguration mechanism was important for us, and the conditions under which it's possible can be further explored. Our mechanism relies on using pairs of toeholds to enable or disable m_4 steps, while m_3 steps only rely on one toehold, so it's an interesting question whether the same mechanisms can be done with only m_3 steps. And finally, if any experimental testing shows that DNA doesn't behave the way our model assumes, defining the difference might open up further theoretical exploration.

References

1 Dana Angluin, James Aspnes, Zoë Diamadi, Michael J. Fischer, and René Peralta. Computation in networks of passively mobile finite-state sensors. *Distributed Computing*, pages 235–253, March 2006. doi:10.1007/s00446-005-0138-3.

2 Dana Angluin, James Aspnes, and David Eisenstat. Stably computable predicates are semilinear. In *Proceedings of the twenty-fifth annual ACM symposium on Principles of distributed computing*, pages 292–299. ACM, 2006. doi:10.1145/1146381.1146425.

3 Stefan Badelt, Casey Grun, Karthik V Sarma, Brian Wolfe, Seung Woo Shin, and Erik Winfree. A domain-level DNA strand displacement reaction enumerator allowing arbitrary non-pseudoknotted secondary structures. *Journal of the Royal Society Interface*, 17(167):20190866, 2020. doi:10.1098/rsif.2019.0866.

4 Stefan Badelt, Seung Woo Shin, Robert F Johnson, Qing Dong, Chris Thachuk, and Erik Winfree. A general-purpose CRN-to-DSD compiler with formal verification, optimization, and simulation capabilities. In Damien Woods and Yannick Rondelez, editors, *DNA Computing and Molecular Programming*, volume 9818 of Lecture Notes in Computer Science, pages 232–248. Springer, 2017. doi:10.1007/978-3-319-66799-7_15.

5 Luca Cardelli. Two-domain DNA strand displacement. *Mathematical Structures in Computer Science*, 23(02):247–271, 2013. doi:10.1017/S0960129512000102.

6 Luca Cardelli and Gianluigi Zavattaro. On the computational power of biochemistry. In *International Conference on Algebraic Biology*, pages 65–80. Springer, 2008. doi:10.1007/978-3-540-85101-1_6.

7 Ho-Lin Chen, David Doty, and David Soloveichik. Deterministic function computation with chemical reaction networks. *Natural Computing*, 13(4):517–534, 2014. doi:10.1007/s11047-013-9393-6.

8 Yuan-Jyue Chen, Neil Dalchau, Niranjan Srinivas, Andrew Phillips, Luca Cardelli, David Soloveichik, and Georg Seelig. Programmable chemical controllers made from DNA. *Nature Nanotechnology*, 8(10):755–762, 2013. doi:10.1038/nnano.2013.189.

9 Kevin M Cherry and Lulu Qian. Scaling up molecular pattern recognition with DNA-based winner-take-all neural networks. *Nature*, 559(7714):370, 2018. doi:10.1038/s41586-018-0289-6.

10 Samuel Clamons, Lulu Qian, and Erik Winfree. Programming and simulating chemical reaction networks on a surface. *Journal of the Royal Society Interface*, 17(166):20190790, 2020. doi:10.1098/rsif.2019.0790.

11 Nadine L Dabby. *Synthetic molecular machines for active self-assembly: prototype algorithms, designs, and experimental study.* PhD thesis, California Institute of Technology, 2013. doi:10.7907/T0ZG-PA07.

12 Robert M Dirks, Justin S Bois, Joseph M Schaeffer, Erik Winfree, and Niles A Pierce. Thermodynamic analysis of interacting nucleic acid strands. *SIAM review*, 49(1):65–88, 2007. doi:10.1137/060651100.

13 David Doty and Monir Hajiaghayi. Leaderless deterministic chemical reaction networks. In David Soloveichik and Bernard Yurke, editors, *DNA 2013: Proceedings of The 19th International Meeting on DNA Computing and Molecular Programming*, volume 8141 of *Lecture Notes in Computer Science*, pages 46–60. Springer International Publishing, 2013. doi:10.1007/978-3-319-01928-4_4.

14 Benjamin Groves, Yuan-Jyue Chen, Chiara Zurla, Sergii Pochekailov, Jonathan L Kirschman, Philip J Santangelo, and Georg Seelig. Computing in mammalian cells with nucleic acid strand exchange. *Nature nanotechnology*, 11(3):287–294, 2016. doi:10.1038/nnano.2015.278.

15 Hope Amber Johnson and Anne Condon. A Coupled Reconfiguration Mechanism for Single-Stranded DNA Strand Displacement Systems. In Thomas E. Ouldridge and Shelley F. J. Wickham, editors, *28th International Conference on DNA Computing and Molecular Programming (DNA 28)*, volume 238 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 3:1–3:19, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.DNA.28.3.

16 Robert F. Johnson. Impossibility of sufficiently simple chemical reaction network implementations in DNA strand displacement. In Ian McQuillan and Shinnosuke Seki, editors, *Unconventional Computation and Natural Computation*, pages 136–149. Springer International Publishing, 2019. doi:10.1007/978-3-030-19311-9_12.

17 Robert F Johnson, Qing Dong, and Erik Winfree. Verifying chemical reaction network implementations: A bisimulation approach. *Theoretical Computer Science*, 2018. doi:10.1016/j.tcs.2018.01.002.

18 Robert F. Johnson and Lulu Qian. Simplifying Chemical Reaction Network Implementations with Two-Stranded DNA Building Blocks. In Cody Geary and Matthew J. Patitz, editors, *26th International Conference on DNA Computing and Molecular Programming (DNA 26)*, volume 174 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:14, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.DNA.2020.2.

19 Robert F Johnson and Erik Winfree. Verifying polymer reaction networks using bisimulation. *Theoretical Computer Science*, 843:84–114, 2020. doi:10.1016/j.tcs.2020.08.007.

20 Matthew R. Lakin and Andrew Phillips. Modelling, simulating and verifying Turing-powerful strand displacement systems. In Luca Cardelli and William Shih, editors, *DNA Computing and Molecular Programming*, pages 130–144, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. doi:10.1007/978-3-642-23638-9_12.

21 Matthew R. Lakin, Simon Youssef, Filippo Polo, Stephen Emmott, and Andrew Phillips. Visual DSD: a design and analysis tool for DNA strand displacement systems. *Bioinformatics*, 27(22):3211–3213, 2011. doi:10.1093/bioinformatics/btr543.

22 Jérôme Leroux. Vector addition systems reachability problem (a simpler solution). In *EPiC*, volume 10, pages 214–228. Andrei Voronkov, 2012. doi:10.29007/BNX2.

23 Rasmus L Petersen, Matthew R Lakin, and Andrew Phillips. A strand graph semantics for DNA-based computation. *Theoretical computer science*, 632:43–73, 2016. doi:10.1016/j.tcs.2015.07.041.

24 Lulu Qian, David Soloveichik, and Erik Winfree. Efficient Turing-universal computation with DNA polymers. In Yasubumi Sakakibara and Yongli Mi, editors, *DNA Computing and Molecular Programming*, volume 6518 of Lecture Notes in Computer Science, pages 123–140. Springer, 2011. doi:10.1007/978-3-642-18305-8_12.

25 Lulu Qian and Erik Winfree. Scaling up digital circuit computation with DNA strand displacement cascades. *Science*, 332(6034):1196–1201, 2011. doi:10.1126/science.1200520.

26 Lulu Qian and Erik Winfree. Parallel and scalable computation and spatial dynamics with DNA-based chemical reaction networks on a surface. In Satoshi Murata and Satoshi Kobayashi, editors, *DNA Computing and Molecular Programming*, volume 8727 of Lecture Notes in Computer Science, pages 114–131. Springer, 2014. doi:10.1007/978-3-319-11295-4_8.

27 Charles Rackoff. The covering and boundedness problems for vector addition systems. *Theoretical Computer Science*, 6(2):223–231, 1978. doi:10.1016/0304-3975(78)90036-1.

28 Friedrich C. Simmel, Bernard Yurke, and Hari R. Singh. Principles and applications of nucleic acid strand displacement reactions. *Chemical Reviews*, 119(10):6326–6369, 2019. PMID: 30714375. doi:10.1021/acs.chemrev.8b00580.

29 David Soloveichik, Matthew Cook, Erik Winfree, and Jehoshua Bruck. Computation with finite stochastic chemical reaction networks. *Natural Computing*, 7(4):615–633, 2008. doi:10.1007/s11047-008-9067-y.

30 David Soloveichik, Georg Seelig, and Erik Winfree. DNA as a universal substrate for chemical kinetics. *Proceedings of the National Academy of Sciences*, 107(12):5393–5398, 2010. doi:10.1073/pnas.0909380107.

31 Niranjan Srinivas, James Parkin, Georg Seelig, Erik Winfree, and David Soloveichik. Enzyme-free nucleic acid dynamical systems. *Science*, 358, 2017. doi:10.1126/science.aal2052.

32 Allison Tai and Anne Condon. Error-free stable computation with polymer-supplemented chemical reaction networks. In Chris Thachuk and Yan Liu, editors, *DNA Computing and Molecular Programming*, pages 197–218, Cham, 2019. Springer International Publishing. `doi:10.1007/978-3-030-26807-7_11`.

33 Chris Thachuk, Erik Winfree, and David Soloveichik. Leakless DNA strand displacement systems. In Andrew Phillips and Peng Yin, editors, *DNA Computing and Molecular Programming*, volume 9211 of Lecture Notes in Computer Science, pages 133–153. Springer, 2015. `doi:10.1007/978-3-319-21999-8_9`.

34 Anupama J Thubagere, Chris Thachuk, Joseph Berleant, Robert F Johnson, Diana A Ardelean, Kevin M Cherry, and Lulu Qian. Compiler-aided systematic construction of large-scale DNA strand displacement circuits using unpurified components. *Nature Communications*, 8:14373, 2017. `doi:10.1038/ncomms14373`.

35 Boya Wang, Chris Thachuk, Andrew D Ellington, Erik Winfree, and David Soloveichik. Effective design principles for leakless strand displacement systems. *Proceedings of the National Academy of Sciences*, 115(52):E12182–E12191, 2018. `doi:10.1073/pnas.1806859115`.

A Formal definitions of DSD

Here we give formal definitions of various aspects of DSD systems that were omitted from Section 2 for lack of space. When discussing the definition of correctness in terms of CRN bisimulation, we also discuss how our CRN mechanism satisfies it.

Anchored and semi-anchored bonds

In Definition 3 the u, m_3, and m_4 steps had conditions based on whether certain bonds were "anchored by an adjacent bond as shown in Figure 2". This refers to a formal definition of anchored, which is as follows:

▶ **Definition 6.** *A bond (x, x^*) is* anchored *if, for some other bond (y, y^*), either x y and y^* x^* are each adjacent 5' to 3' on their respective strands, or y x and x^* y^* are. We also in that case say (x, x^*) is* anchored by *(y, y^*), and note that if so, then (y, y^*) is also anchored by (x, x^*).*

Fig. 2 contains many examples of anchored bonds. In particular, the resulting bonds of m_3 and m_4 steps, and the cases where a u step can't happen, are examples of anchored bonds as described in the rules of Definition 3.

One can also define a *semi-anchored* model, as well as a remote toehold model, by relaxing those conditions:

▶ **Definition 7.** *A bond (x, x^*) is* semi-anchored *if, for some other bonds (y_i, y_i^*), $1 \leq i \leq n$, y_i^* y_{i+1} are adjacent 5' to 3', as are either x y_1 and y_n^* x^* or x^* y_1 and y_n^* x. Note that an anchored bond is semi-anchored, satisfying this definition for $n = 1$.*

The semi-anchored pseudoknot-allowed or pseudoknot-free DSD models *are defined from the corresponding anchored model by modifying the m_3 and m_4 steps to require the resulting bond(s) to be semi-anchored instead of anchored. The u step is unchanged, still able to happen if the bond is not fully anchored.*

The remote-toehold pseudoknot-allowed DSD model *is defined from the anchored pseudoknot-allowed model by removing the requirement in the m_3 and m_4 steps that the resulting bond(s) be anchored at all. The* remote-toehold pseudoknot-free DSD model *is defined from the anchored pseudoknot-free model by, for m_3 and m_4 steps, removing the requirement that the resulting bond(s) be anchored but adding a requirement that the resulting complex be non-pseudoknotted, since adding a pseudoknot would now otherwise be possible.*

It is also possible to mix-and-match conditions, e.g. requiring m_4 steps to be fully anchored while allowing remote-toehold m_3 steps. In fact, there are a lot of different possible DSD models, of which these are just a few; for example, Petersen et al. define a model that forbids u steps if the bond is semi-anchored (which they refer to as "anchored") [23]. We disagree with that because such a model would cause difficulties for tx_4 motifs, which seem to be experimentally possible [11, 14].

A good example of the semi-anchored model is the 3-way-initiated 4-way branch migration mechanism used in the original surface CRN paper [26]; the m_3 step that forms the 4-way junction is valid in the semi-anchored model, but not the anchored model. In our work, the semi-anchored model is referenced only for comparison with the anchored model. The remote-toehold model is mentioned in passing related to the conditions in the impossibility proof of our previous work: that work used a mixed model where m_3 was remote-toehold while m_4 was anchored only [16]. Then, for the class of DSD systems it proved could not exist, they were required to implement CRNs in such a way that no remote-toehold m_3 steps would ever be enumerated. In this work, the $2tx_4$ motif (Figure 3) contains two u, b sequences each of which is equivalent to a remote-toehold m_3, so our system would not meet the condition of the impossibility proof; thus this mechanism's existence suggests those conditions were too strict.

We mentioned that the anchored, pseudoknot-free model is equivalent to Peppercorn's model with the `--reject-remote` option used [3]. Peppercorn's standard model, without that flag, is equivalent to the remote-toehold pseudoknot-free model.

Condensed reactions

One assumption we often make is that, in the (relatively low) concentrations typical for DSD, if two complexes come together and interact, they will have enough time to reach a stable state (possibly involving separating into two or more complexes) before any third complex comes by and interacts with them. Thus the sequence of steps from the first (necessarily) b step to the eventual separation can be treated as a single reaction, called a *condensed reaction*, following [3]:

▶ **Definition 8.** *In a given DSD system, a* resting set *(sometimes* resting state*) is a set of complexes such that any complex in the set can transform into any other via unimolecular steps, and no complex in the set can transform into any complex* not *in the set via a unimolecular step. A* condensed reaction *is a sequence of steps, starting with a bimolecular b step whose reactants are both elements of some resting sets, followed by 0 or more unimolecular steps, resulting in one or more complexes each of which is an element of some resting set. More precisely, a condensed reaction is an equivalence class of such pathways, where two pathways are equivalent if they are the same reaction when considered as reactions from resting sets to resting sets.*

For example, the entire pathway of Figure 5 is a single condensed reaction. It is worth noting that in a fully reversible system, any condensed reaction must have more than one product, since any single complex formed after an initial b step could separate, via the reverse of the pathway up to that point, into the original complexes, and thus that intermediate complex cannot be an element of a resting set.

The assumption that only condensed reactions can happen, if it holds, is particularly useful in claiming that no undesired cross-reactions can happen in a given system. Usually there aren't actually any trimolecular undesired reactions that could happen, but the assumption lets us avoid checking a large amount of combinations that don't go anywhere. This assumption specifically doesn't hold for cooperative hybridization mechanisms, including but not limited to our cooperative 4-way branch migration-based CRN implementation [18].

CRN implementations and CRN bisimulation

We use the formal verification method of CRN bisimulation [17] to determine whether our DSD system correctly implements the CRN it claims to implement. CRN bisimulation is defined as a relation between two CRNs, so here we take the abstract CRN as the *formal* CRN, and the enumerated CRN of the DSD system as the *implementation* CRN. CRN bisimulation then says the implementation is correct if there is an *interpretation* of implementation species that satisfies certain conditions, from [17]:

▶ **Definition 9.** *Given a formal CRN and an implementation CRN, an* interpretation *is a function from implementation species to multisets of formal species. An interpretation is a* CRN bisimulation *if it satsifies the following three conditions:*

- Atomic condition: *Any formal CRN species has an implementation species interpreted as the singleton multiset of that formal species.*
- Delimiting condition: *Any implementation reaction, when interpreted, is either a formal reaction or trivial (a "reaction" whose reactants equal its products). ("Anything that can happen, should.")*
- Permissive condition: *In any implementation state, for any formal reaction that can happen in its interpretation, a reaction interpreted as that reaction can happen after zero or more trivial reactions. ("Anything that should happen, can.")*

The implementation CRN is correct according to CRN bisimulation *if there exists an interpretation that is a CRN bisimulation.*

In general with DSD systems, CRN bisimulation is done after removing fuel species from the enumerated CRN, and the interpretation is required to interpret each signal species as the singleton multiset of its formal species (thus satisfying the atomic condition automatically). We note that as far as CRN bisimulation is concerned, removing a fuel species f is equivalent to adding a reaction $\emptyset \to f$ and requiring the interpretation of f to be $\emptyset$.

Most CRN implementations are *modular* in a certain sense: the implementation consists of distinct modules, one for each formal reaction, each of which implements that formal reaction starting and ending with a set of common "signal" species. This motivates a definition of *modular CRN bisimulation*, which is often easier to check for both computers and humans, again from [17]:

▶ **Definition 10.** *Given a formal and implementation CRN divided into a set of pairs of formal and implementation sub-CRNs, a* modular bisimulation interpretation *is an interpretation of the implementation species such that:*

- *the interpretation, restricted to any pair of sub-CRNs, is a CRN bisimulation on those sub-CRNs (i.e., satisfies the atomic, delimiting, and permissive conditions);*
- *there is a set of* signal species, *where each signal species is an implementation species interpreted as one copy of one formal species and appears in each implementation sub-CRN for which its formal species appears in the corresponding formal sub-CRN, with at least one signal species for each formal species; and*
- *for each implementation species in any sub-CRN, there is a sequence of trivial reactions that turns that species into a set of signal species corresponding to its interpretation, plus possibly some species whose interpretation is empty.*

In other words, each "module" must implement any formal reaction in it, and also be able to turn any of its implementation species into the "signal species" that can then go on to implement any formal reaction in any other module. In DSD implementations you might have one module for each formal reaction plus one "crosstalk module", containing no formal reactions and all the implementation reactions that might occur between species belonging to two different modules. The modular bisimulation condition then checks that those crosstalk reactions neither turn any species into something they shouldn't turn into, nor destroy any species in a way that can't be turned back into its signal species.

In Figure 7, the A_{on} and A_{off} complexes are interpreted as A and $\emptyset$, respectively, with A_{on} the modular signal species for A; and similar for B. The gate strands are interpreted as, from top to bottom, $\emptyset$, A, $A + B$, and $\emptyset$. Finally, $int_{r;off}$ is interpreted as $\emptyset$, and $int_{r;on}$ as the products of the formal reaction, e.g. $C + D$ if the reaction is $A + B \rightleftharpoons C + D$. So the first two reactions are trivial, and the third is interpreted as the formal reaction. Any point in the process can reverse itself to produce the signals A_{on} and B_{on}, and the (not shown) product half can turn $int_{r;on}$ into $C_{on} + D_{on}$, satisfying modularity. While this looks like a proof of correctness, the actual hard work is in reaction enumeration, proving that the reactions shown in the figure are the only ones that happen. Since CRN bisimulation happens after enumeration, this is beyond its scope.

Pseudoknot-robustness is then defined in terms of CRN bisimulation. First, note that if the same initial complexes are enumerated separately in the pseudoknot-allowed and pseudoknot-free anchored models, the pseudoknot-allowed species and reactions will be supersets of their pseudoknot-free counterparts. Then a DSD implementation of a formal CRN is *pseudoknot-robustly correct* if there is a single interpretation that is a CRN bisimulation on the pseudoknot-allowed enumerated CRN, and its restriction to the pseudoknot-free enumerated species is a CRN bisimulation on that CRN. Equivalently, it is correct if the atomic and permissive conditions hold in the psuedoknot-free model, the delimiting condition holds in the pseudoknot-robust model, and (similar to the modularity condition) any implementation species in the pseudoknot-robust model can, via trivial reactions, turn into only species that exist in the pseudoknot-free model.

B Further discussion of correctness

We did code the well-mixed CRN implementation scheme (Figure 7) into Nuskell [4] and verified that it satisfies modular bisimulation on two small formal CRNs, a binary counter and an oscillator. However, Nuskell (in particular, Peppercorn, which it uses for reaction enumeration) only handles psuedoknot-free systems, and cannot confirm that our implementation scheme is pseudoknot-robust.

Our argument for correctness is, first, from Figures 5-7 we clearly see that starting from the signal species $A_{on} + B_{on}$ with appropriate fuel species ($fuel_{r;A,B:fwd}$, $int_{r;off}$, $fuel_{r;C,D:rev}$, C_{off}, and D_{off}, with those last three being the products copy of the mechanism that is implied but not shown in Figure 7), the reaction can progress, consuming A_{on} and B_{on} and producing C_{on} and D_{on}. For CRN bisimulation purposes, the step that produces $int_{r;on}$ is the "turning point" where the interpretation (as described in the previous appendix) changes from $A + B$ to $C + D$. This argument is effectively the permissive condition of CRN bisimulation.

Second, the modularity condition: given that A_{off} and B_{off} are fuels, any intermediate complex can reverse the process, which produces the signal species (A_{on} and/or B_{on}) of its interpretation; and similarly $int_{r;on}$ or any intermediate of the products half can complete the process to produce the signal species C_{on} and/or D_{on}. This shows that any possible combination of complexes *can* do anything its interpretation should be able to do, leaving only to show that no undesired reactions can happen.

For undesired reactions, our argument comes from checking all combinations of toeholds that appear together in a way that could allow a four-way branch migration step. We are assuming that no m_4 step occurs unless toeholds on both sides match; whether this is true is an experimental question. Since the entire system never has any unbound long domains anywhere, no m_3 steps are possible, and any combination of just b and u steps can't make a nontrivial condensed reaction (argument similar to one made in [16]). In the initial states, those toehold combinations are (from Figure 5, with $\rho = \nu^*$ applied, each given 5' first then 3'): (τ, ν^*), (λ^*, σ), (μ, ν), (μ, ν^*), (ω^*, λ), (ν, τ^*) on the top left, and (τ, ν), (λ^*, σ), (μ, ν^*), (μ, ν), (ν^*, τ^*), (ω^*, λ) on the bottom left. In various intermediate steps (including some instances of the intermediate of the $2tx_4$ motif, not shown in Figure 5), the following additional pairs appear: (σ^*, λ), (ν, μ^*), (λ^*, ω), (ν^*, μ^*). So for example a (τ, ν^*) can bind and enable an m_4 step with a (ν, τ^*) on the same long domain identity, but not any other combination, in the assumptions of the model we're using.

So, the possible interactions are: the $(\tau, \nu^*) + (\nu, \tau^*)$ interaction in the intial state, which is the intended first step; the $(\tau, \nu) + (\nu^*, \tau^*)$ interaction in the final state, which is the intended first step of the reverse pathway; and no other possible interactions between anything in the initial and/or final states. The four pairs that appear only in intermediates will be able to interact with various complementary pairs on the same intermediate complex; it can be seen from Figure 5 that the only such interactions will be the intended forward and reverse $2tx_4$ motifs. Finally, two copies of the mechanism at the same intermediate step will have complementary pairs of toeholds; however an interaction between them would be a violation of the condensed reaction (i.e. low-concentration) assumption. It's not actually clear whether any such effectively tetramolecular reactions would actually cause problems; two instances of an A intermediate interacting, if they eventually just disentangle and both complete their respective reactions, might just end up in the same place as each one completing independently. Or possibly there could be issues if an $A + B$ reactant side interacts with an $A + C$ reactant side, or other such crosstalk. But in the condensed reaction model, this mechanism unambiguously works.

The argument for correctness of the polymer mechanisms would be along the same lines, confirming that no complementary toehold pairs appear except for the intended forward and reverse steps.