Automated, constraint-based analysis of tethered DNA nanostructures

Matthew R. Lakin[1,2,3] and Andrew Phillips[4]

[1] Department of Computer Science, University of New Mexico, NM, USA
[2] Center for Biomedical Engineering, University of New Mexico, NM, USA
[3] Department of Chemical & Biological Engineering, University of New Mexico, NM, USA
[4] Microsoft Research, 21 Station Rd, Cambridge CB1 2FB, UK

mlakin@cs.unm.edu aphillip@microsoft.com

Abstract. Implementing DNA computing circuits using components tethered to a surface offers several advantages over using components that freely diffuse in bulk solution. However, automated computational modeling of tethered circuits is far more challenging than for solution-phase circuits, because molecular geometry must be taken into account when deciding whether two tethered species may interact. Here, we tackle this issue by translating a tethered molecular circuit into a constraint problem that encodes the possible physical configurations of the components, using a simple biophysical model. We use a satisfaction modulo theories (SMT) solver to determine whether the constraint problem associated with a given structure is satisfiable, which corresponds to whether that structure is physically realizable given the constraints imposed by the tether geometry. We apply this technique to example structures from the literature, and discuss how this approach could be integrated with a reaction enumerator to enable fully automated analysis of tethered molecular computing systems.

1 Introduction

Molecular computing using solution-phase DNA circuits is a powerful method for implementing nanoscale information processing systems. In particular, DNA strand displacement has emerged as a proven method of engineering networks of programmable computational elements [1, 2] and a powerful theoretical framework has built up around it [3, 4]. In the computational modeling of DNA strand displacement networks, the assumption that all components are freely diffusing in bulk solution means that the physical conformations of interacting molecular species may be neglected when enumerating the possible reactions. This assumption is heavily exploited in existing modeling systems for solution-phase DNA strand displacement networks, such as the DSD programming language [5] and the associated Visual DSD software [6].

However, the fact that any pair of components in a freely diffusing molecular circuit may potentially diffuse into proximity and interact means that the entire circuit must be designed so as to eliminate crosstalk from all possible undesired interactions. Thus, even if the circuit is designed in a modular fashion, each instance of a given module must be implemented using different nucleotide sequences, which limits the scalability of circuits. Indeed, the largest DNA computing circuit built to date is the “square root”

circuit reported by Qian and Winfree [7], which required 74 initial DNA species (excluding input signals) for a total of 130 different DNA strands. Reducing the number of distinct molecular species required to implement complex computational systems is therefore a crucial direction for future research in molecular computing.

Advances in the field of structural DNA nanotechnology, in particular, the development of DNA origami [8] as a reliable method for building DNA tile nanostructures [9], has raised the possibility of implementing molecular computers using components that are tethered to a DNA origami tile surface. This would enable nucleotide sequences to be shared between components that are tethered far enough apart to not interact, which means that large-scale circuits could be constructed using many copies of standardized components. Furthermore, tethering circuit components in close proximity increases the speed of circuit computation because components do not need to diffuse in three dimensions prior to interacting [10]. Indeed, several proposals for implementing molecular computers using surface-bound DNA strand displacement networks based on hairpin-opening reactions have been previously published [11, 12]. A further motivation for studying molecular circuits tethered to DNA tiles is that DNA nanostructures show great promise as a vehicle to deliver theranostic molecular devices to cells and tissues [13].

From a modeling perspective, however, the move to implementing molecular computers using components that are tethered to a surface presents some challenges. In particular, tethering components to the surface at particular locations makes it necessary to consider the geometry of the system when deciding whether two components can actually interact with each other. For example, it might be the case that two components contain complementary single-stranded domains, but are tethered to the surface too far apart to actually be able to hybridize. In certain cases it may be possible to derive simple expressions to compute whether domains can interact [12], but this is not possible in the general case. Our previous work on modeling tethered strand displacement networks [14] avoided this issue by relying on the programmer to specify which components could interact, however, in many cases this requires additional biophysical calculations to be made separately. Thus, in order to build an automated compiler for tethered molecular circuits, we require an automated, general purpose method for computing whether the geometry of two tethered molecular species permits them to interact.

In this paper, we present an automated solution to the problem of reaction enumeration in tethered strand displacement systems. We translate tethered structures into sets of arithmetic constraints on variables that represent the physical locations of their components. If the constraint problem associated with a given structure is *satisfiable*, it means that there is a plausible physical configuration of the components that can be adopted by the system. Conversely, if the constraint problem is *unsatisfiable*, there is no physical configuration of the components that can produce that structure. By solving the constraints using an off-the-shelf satisfaction modulo theories (SMT) solver, we obtain a procedure that allows us to automatically detect whether a structure is physically plausible, given the constraints imposed by the tethers. To our knowledge, this is the first fully automated procedure for analyzing molecular geometry in the context of a tethered molecular computing system.

2 Localized processes

In this section, we present our language of localized processes for representing tethered DNA structures. Definition 1 extends the process calculus-based presentation of DNA nanostructures from our previous work [15] with syntax for tethers on strands that attach them to a surface (representing, e.g., a DNA origami tile) at particular coordinates, and annotations to represent the physical coordinates of different parts of the structure.

Definition 1 (Localized processes). *The syntax of localized processes, P, is expressed in terms of* domain names *a, b, ...,* bond names *i, j, k, ...,* variables *x, y, z, ..., and* real-valued constants $r \in \mathbb{R}$. *Then, the grammar of localized processes is as follows.*

Coordinate c	$::= x$	*Variable*
	$\mid r$	*Real-valued constant*
Tether t	$::= tether$	*Tethered*
	$\mid \varepsilon$	*Untethered*
Position π	$::= t(c_1, c_2, c_3)$	*Position with coordinates*
Domain d	$::= a$	*Domain name*
	$\mid a^*$	*Complemented domain name*
(Un)bound domain o	$::= d$	*Free domain*
	$\mid d!i$	*Bound domain with bond i*
Strand S	$::= <\pi_0\ o_1\ \pi_1\ \cdots\ \pi_{M-1}\ o_M\ \pi_M>$	*Strand with M domains,* $M \geq 1$
Process P	$::= (S_1 \mid \cdots \mid S_N)$	*Multiset of N strands,* $N \geq 0$

We write $strands(P)$, $bonds(P)$, $posns(P)$, and $vars(P)$ for the sets of *strands*, *bonds*, *positions*, and *variables* that occur in P, respectively. We consider processes *equal* up to re-ordering of strands, renaming of bonds, and renaming of variables, and we only consider processes that are *well-formed*, by which we mean that each bond in $bonds(P)$ appears *exactly twice* and is shared between complementary domains, and that each variable in $vars(P)$ appears *exactly once*.

We assume the existence of a predicate *toehold* such that $toehold(a)$ returns true if and only if a is a toehold domain. For convenience, we also write $a\hat{}$ iff $toehold(a)$ holds. We also assume the existence of a function *len* such that $len(a)$ returns the length in nucleotides of the DNA sequence for the domain name a (and thus returns a positive integer). We assume that these functions are also lifted to possibly-complemented domains d, so that $f(a^*) = f(a)$, for $f \in \{toehold, len\}$. Similarly, we lift the complementation syntax to possibly-complemented domains, by defining $(a^*)^* = a$.

Between each pair of domains on every strand, and on each strand terminus, is a *position* that represents the location of that part of the structure in 3D space. The position $\varepsilon(c_x, c_y, c_z)$, which we may abbreviate as simply (c_x, c_y, c_z), means that part of the structure is untethered and located at x-coordinate c_x, y-coordinate c_y, and z-coordinate c_z. (Each of the coordinates may be a variable that can be assigned a value during the constraint solving procedure, or a real-valued constant that represents a specific location.) Similarly, the position $tether(c_x, c_y, c_z)$ represents a tether that attaches that part of the strand to the underlying surface at the location (c_x, c_y, c_z) where, typically, $c_z = 0$ and c_x and c_y are real-valued constants. In practice, we may elide certain positions from

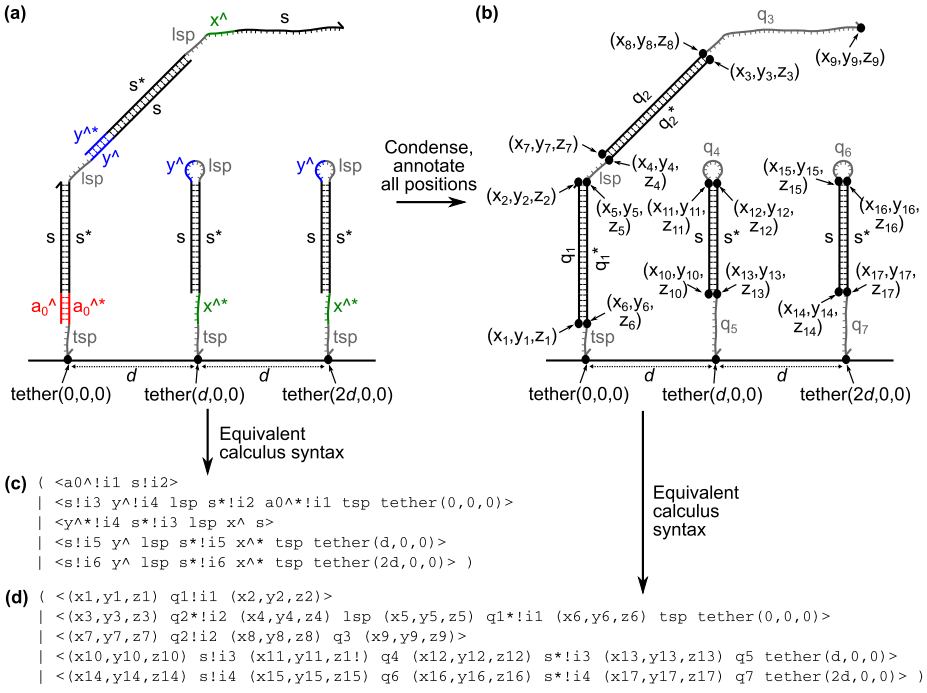

Fig. 1. Example of representing a tethered structure. **(a)** Secondary structure of a three-stator transmission line after [11], with the three stators arranged in a straight line. **(b)** Condensed version of the structure from (a), with all positions fully annotated. Here, the q domains are the freshly generated domain names. **(c)** Calculus syntax representing the initial structure from (a). **(d)** Calculus syntax representing the condensed structure from (b).

the process syntax altogether, with the understanding that the physical position of junction between the domains on either side (or the strand terminus, if at the end of the strand) will be represented by a freshly generated position with syntax $(c_x{}^\dagger, c_y{}^\dagger, c_z{}^\dagger)$, where $c_x{}^\dagger$, $c_y{}^\dagger$, and $c_z{}^\dagger$ are unique, freshly generated variables.

Note that we use the term "process" here for historical reasons. Although the long-term goal of our work is to enable automated enumeration of interactions in localized processes, in this paper we focus solely on the problem of analyzing the geometry of a single state of the localized process, and we use the process syntax solely as a means of representing the structure in question. Figure 1 presents graphical and syntactic representations of the tethered system that will serve as our running example throughout the paper—a three-stator transmission line after [11].

3 Biophysical model

Before we present our translation of localized processes into constraint sets, we first present the assumptions about the biophysics of DNA that will underlie the transla-

tion. Crucially, we will model double-stranded DNA duplexes as rigid rods and single-stranded DNA as infinitely flexible, freely jointed chains. We also assume that joints (nicks) between double-stranded duplexes are infinitely flexible. We will neglect the thickness of DNA strands, and will also neglect the length of the bonds between complementary bases by requiring the ends of the two strands that make up a duplex to be positioned at exactly the same point in space. In computing whether structural components may interact, we will also not account for steric effects that could, for example, eliminate solutions to the constraint problem that would require part of the structure to pass through another to form a catenane. This model is clearly simplified, and we refer the user to Section 7 for further discussion of these assumptions and how this model might be made more realistic.

4 Condensing localized processes

When deriving geometric constraints from localized processes, we will model every double-stranded duplex as a single rigid rod, as described in Section 3 above, even when that duplex consists of multiple sequence domains. Thus, before converting a localized process into a set of constraints we must first *condense* the process by combining all domains on a given strand that represent a continuous duplex into a single extended domain. To reduce the size of the resulting constraint problem, we will also condense continuous single-stranded regions into extended domains. The condensing process is a straightforward binary relation $\longrightarrow_{condense}$ on localized processes that can be defined by the following rewrite rules:

$$(<\cdots\ \pi_j\ d_1!i_1\ \pi_{j+1}\ d_2!i_2\ \pi_{j+2}\ \cdots> \mid <\cdots\ \pi_k\ {d_2}^*!i_2\ \pi_{k+1}\ {d_1}^*!i_1\ \pi_{k+2}\ \cdots> \mid P)$$
$$\longrightarrow_{condense} (<\cdots\ \pi_j\ q!i_1\ \pi_{j+2}\ \cdots> \mid <\cdots\ \pi_k\ q^*!i_1\ \pi_{k+2}\ \cdots> \mid P)$$

$$(<\cdots\ \pi_j\ d_1!i_1\ \pi_{j+1}\ d_2!i_2\ \pi_{j+2}\ \cdots\ \pi_k\ {d_2}^*!i_2\ \pi_{k+1}\ {d_1}^*!i_1\ \pi_{k+2}\ \cdots> \mid P)$$
$$\longrightarrow_{condense} (<\cdots\ \pi_j\ q!i_1\ \pi_{j+2}\ \cdots\ \pi_k\ q^*!i_1\ \pi_{k+2}\ \cdots> \mid P)$$

$$(<\cdots\ \pi_j\ d_1\ \pi_{j+1}\ d_2\ \pi_{j+2}\ \cdots> \mid P) \longrightarrow_{condense} (<\cdots\ \pi_j\ q\ \pi_{j+2}\ \cdots> \mid P)$$

where, in each case, q is a freshly-chosen domain name, for which we assume that $len(q) = len(d_1) + len(d_2)$. We also require that any position removed by the condensing process has the form $\varepsilon(x,y,z)$, that is, the position is untethered and is specified solely by variables and not by constants. In the rule definitions, these positions are referred to as π_{j+1} and π_{k+1}. This condition essentially says that no information is lost by discarding that position, because it was not previously constrained to a specific location in space. If this condition is violated, the condensing process fails and we cannot proceed. However, for practical purposes this is a reasonable condition to impose, since to our knowledge there are no proposed or implemented designs for tethered molecular circuits that include a tether part way along a DNA duplex.

Thus, a single condensing step takes two neighbouring bound domains, either on two strands (the first rule) or on a single strand (the second rule), or two neighbouring single-stranded domains on the same strand (the third rule) and converts them into a single domain with a freshly-generated name (which avoids conflicts with other domain names in the system) whose length is the sum of the lengths of the two domains it

replaces. If the domains were initially bound, they remain so in the condensed version of the process. Furthermore, the coordinates that represent the positions of the endpoints of the new bound domains are the same as those that represented the endpoints of the neighbouring bound domains that were replaced. Figure 1 presents an example of condensing, as applied to our running example system.

It is not hard to see that the condensing process preserves well-formedness of processes (since, in the first two rules, both occurrences of i_2 are removed and i_1 connects two complementary domains in the resulting process), that it is terminating (since every rule application removes a pair of neighbouring bound domains) and that it produces a unique normal form when no more rule applications are possible (modulo the choice of freshly-named new domains that are introduced).

Henceforth we will assume that all processes have been condensed via this procedure, so that all duplexes are represented by a single domain on each bound strand and all neighbouring single-stranded domains have been similarly collapsed into a given domain. This ensures that all domain junctions in the process correspond to points where the structure is flexible.

5 Geometric constraints for localized processes

This section details the main technical contribution of the paper, where we define a translation of a (condensed) localized process into a *set* of arithmetic constraints that represent the possible geometry of the structure according to the biophysical model outlined in the previous section. This set of constraints will collectively encode the geometry of the structure, and our intention is that the set structure represents an implicit conjunction, so that the set of constraints is satisfied iff *all* constraints within the set are simultaneously satisfied (under some mapping of variables to values).

We will use a 3D Cartesian coordinate system that allows us to locate each domain junction with respect to the surface to which the tethered components are attached. For simplicity, we assume that all tethered components are attached to the *same* surface, though that restriction could be straightforwardly relaxed if we assume that there are no interactions between components tethered to different surfaces. By convention, we will assume that the surface (which could model, e.g., a DNA origami tile) lies in the $z = 0$ plane, and that all components are tethered to the same side of the surface and protrude on the positive z side. We now begin to define the different classes of constraints that will make up the constraint-based representation of structure geometry. First, we define constraints that represent the physical length of each domain in a process.

Definition 2 (Domain length constraints). *For a localized process P we define the corresponding domain length constraints,* $\mathsf{dlc}(P)$*, as follows:*

$$\mathsf{dlc}(P) = \bigcup_{S \in \mathit{strands}(P)} \mathsf{dlc}(S)$$

$$\textit{where} \quad \mathsf{dlc}(<\pi_0\ o_1\ \pi_1\ \cdots\ \pi_{N-1}\ o_N\ \pi_N>) = \mathsf{dlc}(\pi_0, o_1, \pi_1) \cup \mathsf{dlc}(\pi_1, o_2, \pi_2) \cup \cdots \cup \mathsf{dlc}(\pi_{N-1}, o_N, \pi_N)$$

$$\textit{and} \quad \mathsf{dlc}(t(c_x, c_y, c_z), o, t'(c'_x, c'_y, c'_z)) = \begin{cases} \{(c'_x - c_x)^2 + (c'_y - c_y)^2 + (c'_z - c_z)^2 = (\mathit{len}(a) \times \mathsf{L_{ds}})^2\} & \textit{if } o = a!i \\ \{(c'_x - c_x)^2 + (c'_y - c_y)^2 + (c'_z - c_z)^2 \leq (\mathit{len}(a) \times \mathsf{L_{ss}})^2\} & \textit{if } o = a. \end{cases}$$

In this definition, we must distinguish between single-stranded and double-stranded domains. Thus, $\mathit{len}(a) \times \mathsf{L_{ds}}$ is the (fixed) length of a double-stranded domain a and $\mathit{len}(a) \times \mathsf{L_{ss}}$ is the *maximum* extended length of a single-stranded domain a. The constraints then specify the fixed distance between the two ends of a double-stranded domain (which we model as a rigid rod) or the maximum distance between the two ends of a single-stranded domain (which we model as an infinitely flexible freely-jointed chain), in terms of the variables used to represent the coordinates of each end of the domain. We now define the constraints that encode hybridization between domains.

Definition 3 (Hybridization constraints). *For a localized process P we define the corresponding hybridization constraints,* $\mathsf{hc}(P)$*, as follows:*

$$\mathsf{hc}(P) = \bigcup_{i \in \mathit{bonds}(P)} \mathsf{hc}_P(i)$$

$$\textit{where} \quad \mathsf{hc}_P(i) = \{c_x = c'''_x, c_y = c'''_y, c_z = c'''_z, c'_x = c''_x, c'_y = c''_y, c'_z = c''_z\}$$

$$\textit{and} \quad <\cdots t(c_x, c_y, c_z)\ a!i\ t'(c'_x, c'_y, c'_z) \cdots> \in \mathit{strands}(P)$$

$$\textit{and} \quad <\cdots t''(c''_x, c''_y, c''_z)\ a^*!i\ t'''(c'''_x, c'''_y, c'''_z) \cdots> \in \mathit{strands}(P).$$

As one of the key simplifications in our biophysical model, we assume that the diameter of a DNA duplex is negligible. Thus, if the two (complementary) domains are hybridized, the 5' end of one domain must be colocated with the 3' end of the other, and vice versa. Note that, in the above definition, since $i \in \mathit{bonds}(P)$ and since we assume that P is well-formed, it follows that there exists precisely one pair of complementary domains connected via the bond i, which will satisfy the side conditions.

Finally, we define constraints that situate the structure with respect to the tile surface. This includes the constraints imposed by tethers, which anchor one end of a domain at a fixed location on the tile surface, and constraints that stipulate that no part of the structure may pass through the tile surface.

Definition 4 (Tile constraints). *For a localized process P we define the corresponding tile constraints,* $\mathsf{tc}(P)$*, as follows:*

$$\mathsf{tc}(P) = \bigcup_{\pi \in \mathit{posns}(P)} \mathsf{tc}(\pi)$$

$$\textit{where} \quad \mathsf{tc}(\mathit{tether}(c_x, c_y, c_z)) = \{c_z = 0\}$$

$$\textit{and} \quad \mathsf{tc}(\varepsilon(c_x, c_y, c_z)) = \{c_z \geq 0\}.$$

Thus, positions on a strand where a tether appears (in practice this is typically at the end of the strand, though our model does not require this) are constrained to specific

coordinates, with a z coordinate of 0. Non-tethered positions are simply constrained to have a non-negative z coordinate, i.e., they cannot be "below" the tile. We now combine the different kinds of constraint defined above to create a constraint-based model that encodes the possible geometric configurations of a localized process.

Definition 5 (Constraint-based biophysical models). *For a condensed, localized process P, we write $\mathcal{M}_P$ for the full, constraint-based model, which is defined as follows:*

$$\mathcal{M}_P = \mathsf{dlc}(P) \cup \mathsf{hc}(P) \cup \mathsf{tc}(P).$$

Thus, all constraints from domain lengths, hybridization, and tethers are unioned together to produce a single set of constraints that represents the possible geometric configurations of the structure of the localized process.

Definition 6 (Plausibility of localized processes). *We say that a localized process P is* plausible *iff its associated constraint set is satisfiable, that is, if there exists a mapping from the variables to real numbers such that all constraints in the constraint set are simultaneously satisfied when the mapping is applied. A localized process that is not plausible is* implausible.

If a localized process is implausible, this means there is no way to arrange the domains in space to form the specified structure without breaking one or more of the geometric constraints imposed by that structure. For example, in the case of a tethered DNA circuit, this may mean that the tethered locations of two components may leave them too far apart to interact with each other. Thus, we can enumerate reactions between tethered components while respecting the tether geometry by checking plausibility of the localized process P that represents the candidate product structure, i.e., by checking satisfiability of the corresponding constraint problem $\mathcal{M}_P$.

6 Results

6.1 Prototype implementation

We implemented a prototype system for checking satisfiability of constraint-based geometric models using the Z3 SMT solver [16]. The prototype is written in Python, and takes as input a localized process, expressed in the syntax from Definition 1. This is then condensed and converted into the corresponding constraint problem (as defined in Definition 5) using the Z3 Python API which can then be checked for satisfiability.

Our preliminary experiments represented domain positions using real-valued variables, because Z3 includes a complete algorithm for solving non-linear constraints over real variables exactly [17]. However, while this algorithm is complete it can be very computationally expensive, and even just changing the values of certain constants within a constraint problem (without changing the structure of the problem) can cause the time taken to solve the constraints to increase from microseconds to many hours. Therefore, we adopted an alternative approach to encode the constraints using floating-point variables, which can be represented within Z3 as bit-vectors of a fixed width. This makes the time taken to solve the constraints far more predictable, on the order of

thirty seconds to one minute depending on the particular problem. However, floating-point representations are inexact, which introduces the possibility of constraints being incorrectly deemed unsatisfiable if any of the required variable assignments cannot be represented exactly using the floating-point representation in question. (The width of the bit vector, and the numbers of bits used to represent the significand and exponent, can be tuned to reduce numerical inaccuracy at the cost of increased compute time to solve the constraints.) We addressed this issue by modifying our constraints slightly, by introducing a *tolerance* parameter ε, so that variables will be considered equal if their values fall within that range. This addresses the issue of numerical inaccuracy because, for a judicious choice of ε, it is highly likely that there will be representable values within the tolerance range that satisfy the constraints. To implement this (approximate) solving algorithm, we simply modify the generated constraints in the model as follows, where e is an arithmetic expression and c is a constant. Thus:

- $e \oplus c$ becomes $e \oplus (c - \varepsilon)$, for $\oplus \in \{>, \geq\}$;
- $e \oplus c$ becomes $e \oplus (c + \varepsilon)$, for $\oplus \in \{<, \leq\}$; and
- $e = c$ becomes $\{e \geq (c - \varepsilon), e \leq (c + \varepsilon)\}$.

This transformation from exact to inexact constraints causes the number of constraints to double at most, and does not increase the number of variables at all. Furthermore, the terms of the form $c \pm \varepsilon$ are constants that can be pre-computed.

6.2 Example

We tested our approach using an example taken from the repeater system in the Seelig lab on hairpin-based circuits on DNA origami tiles [11], introduced in Figure 1. We are particularly interested in the reaction between two tethered species, i.e., when the first stator hairpin has been opened, the freely diffusing fuel hairpin has bound and opened, and the opened fuel hairpin is trying to bind to the second stator hairpin, which is still closed. Because the stator hairpins share common nucleotide sequences, there are two possible interactions, which are shown in Figure 2(a),(b). Figure 2(a) shows the desired interaction, where the opened fuel hairpin binds to the second stator hairpin (S_1), and Figure 2(b) shows an undesirable interaction, where the opened fuel hairpin binds to the third stator hairpin (S_2). In our presentation, every hairpin has a "loop spacer" domain (lsp), and every *tethered* hairpin has a "tether spacer" domain (tsp). Adjusting the lengths of these domains and the inter-stator distance (d) changes the possible behaviors of the system, and the goal of our analysis is to find the parameter sets that enable correct signal transmission, i.e., so that the opened stator S_0 can bind to S_1, resulting in the desired product structure shown in Figure 2(a), but *not* to S_2, which would result in the undesired product structure shown in Figure 2(b).

Results from constraint-based analysis of this example system with the stators arranged in a straight line are presented in Figure 2(c). As described above, we used a floating-point representation for our constraint variables, with an 8 bit exponent and an 8 bit significand. The ε value was 10^{-5}. Values for L_{ds} and L_{ss} were as used in previous work [12], that is, $\mathsf{L}_{\mathsf{ds}} = 0.34\,\mathrm{nm}$ and $\mathsf{L}_{\mathsf{ss}} = 0.68\,\mathrm{nm}$. The lengths of all toehold domains were 5 nucleotides and the lengths of all long domains were 30 nucleotides.

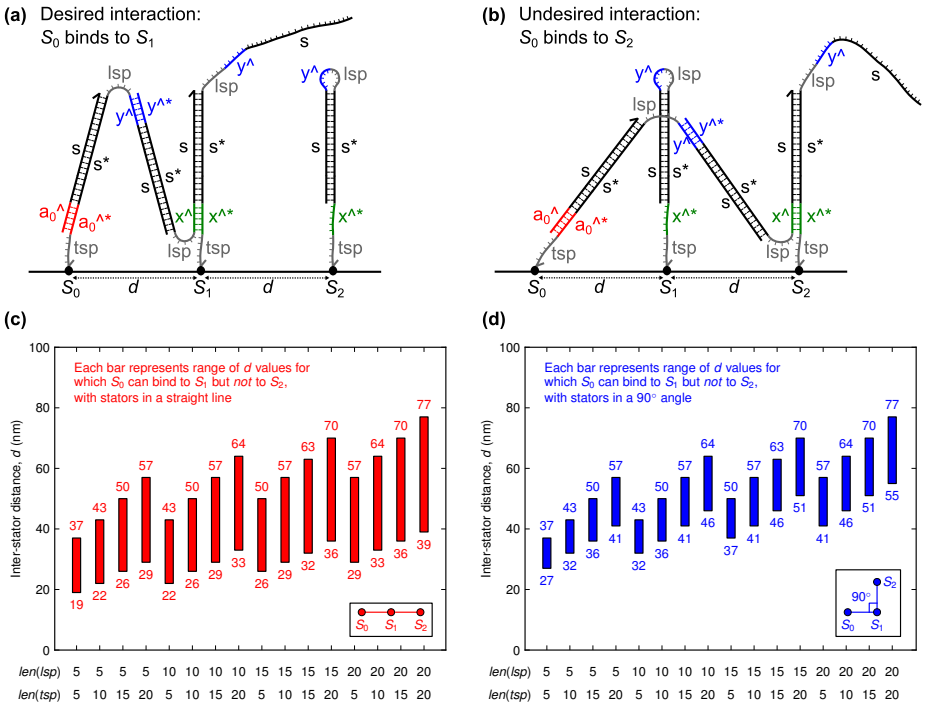

Fig. 2. Results on constraint-based structure modeling. **(a)** Illustration of the product structure for a three-stator transmission line after [11], when S_0 binds to S_1, as desired. **(b)** Illustration of the product structure for the undesirable interaction of the three-stator transmission line, when S_0 binds to S_2. **(c)** Results from constraint-based analysis of the transmission line system when all three stators are arranged in a straight line (bird's eye view shown in inset). Each bar shows the range of integer values of d for which S_0 can bind to S_1 but *not* to S_2, as required for correct signal transmission, for various combinations of loop spacer (lsp) and tether spacer (tsp) domain lengths. **(d)** Results from constraint-based analysis of the transmission line system when all three stators are arranged in a 90° degree angle (bird's eye view shown in inset), with data analysis carried out as specified in part (c).

For each combination of loop spacer (lsp) and tether spacer (tsp) lengths (5, 10, 15, and 20 nucleotides each) we used our prototype system to construct the condensed, localized process representation of the candidate structures formed by interactions between S_0 and S_1 and between S_0 and S_2, convert them to our constraint representation, and test plausibility of each using Z3, for values of the inter-stator distance (d) chosen at 1 nm intervals ranging from 1 nm to 100 nm. ⋆ Each bar in Figure 2(c) represents the range of d values for which S_0 can bind to S_1 but *not* to S_2, as required for correct signal transmission. Below the bar, the stators are close enough together that S_0 can bind directly

⋆ See the Supporting Information (available from the first author's web page) for full details of the examples and corresponding constraints.

to S_2, and above the bar, the stators are so far apart that S_0 cannot even reach S_1. These results show that the range of acceptable values for d increases as the lengths of the lsp and tsp domains increase, as we would expect.

We also analyzed a similar example system that used the same structures, except that the three stator hairpins are arranged to form a 90° angle on the tile surface. In the straight line system, we added extra constraints that all y-coordinates equal zero, as this decreases solving time, but this was *not* done for the 90° angle system. Hence, this example illustrates the applicability of our approach to processes whose components occur in non-trivial geometric arrangements. Results from constraint-based analysis of the 90° angle system are presented in Figure 2(d). These results show that, for identical loop spacer (lsp) and tether spacer (tsp) domain lengths, the maximum safe inter-stator distances are the same in both the straight-line and 90° angle cases, but the minimum safe distance is larger in the 90° angle case. This is because S_2 can be reached diagonally from S_0 when the stators are not arranged in a straight line. Thus, our constraint-based analysis can determine the feasible range of inter-stator distances that enable correct signal transmission without skipping any stators in the sequence, and thereby serves as a proof of concept for automated analysis of tethered molecular circuits.

7 Discussion

The main contribution of this paper is the development of a constraint-based methodology for *automatically* analyzing molecular geometry to determine if certain interactions between species are possible, given the physical constraints imposed by tethers that attach components to a surface (e.g., a DNA origami tile) at specific locations. Our approach therefore offers a principled, general-purpose alternative to the somewhat *ad hoc* rules adopted to handle molecular geometry in other reaction enumerators, e.g., our strand graph system [15] or the DyNAMiC Workbench [18].

7.1 Reaction enumeration

The key advantage of our fully automated analysis is that it could be used in the main loop of a compiler for tethered molecular reaction networks. In this vein, Algorithm 1 presents a pseudocode algorithm for enumerating the state space of a tethered molecular reaction network, in which our constraint-based analysis of the plausibility of tethered molecular structures is used as a filter to prevent the addition of a transition to the state space, even if the domains match, if the reaction would yield a product process whose structure is implausible. We note that, strictly speaking, the constraint-solving process is only required for interactions between different parts of a single tethered structure since, for bimolecular reactions where one or both of the reactants is not tethered, one may assume that the non-tethered species can diffuse and adopt any conformation required to react.

7.2 Biophysical model

The procedure outlined here is for computing *whether* two domains may bind, and an extension of this problem would be to determine *at what rate* that binding reaction may

Algorithm 1: Pseudocode algorithm for constructing the reachable state space of a localized process. This pseudocode assumes the existence of several functions: *unprocessed_states*(S), which returns the set of states in S that have not yet been processed by the algorithm; *candidate_reactions*(P), which returns the set of possible reactions in the process P when considering only sequence complementarity and *not* molecular geometry; and *is_plausible*(P), which decides whether the localized process P is geometrically plausible, using our constraint-based technique.

```
begin
    initialize state space S with localized process P_init as the initial state;
    mark state P_init as unprocessed;
    while unprocessed_states(S) ≠ ∅ do
        let P_reactant be some element of unprocessed_states(S);
        for (P_reactant → P_product) ∈ candidate_reactions(P_reactant) do
            if is_plausible(P_product) then
                if P_product ∉ S then
                    add state P_product to S;
                add transition (P_reactant → P_product) to S;
                mark state P_reactant as processed;
```

occur. In solution-phase systems, one approach to approximate binding rates for particular domains is to use free energy models to compute the free energy of a bound toehold domain, and then make the assumption that binding rates of that domain are dominated by diffusion, which allows an approximation of the off-rate to be computed. However, in tethered systems we cannot necessarily assume that on-rates are primarily influenced by diffusion. An alternative is to use the local concentration approach [19], which corresponds to calculating the probability that the domains will be in a conformation such that they may bind. It is possible that the output from the constraint solver could be used to estimate the number of different coordinate variable instantiations and hence the size of the parameter space in which the constraints are satisfied, which one might be able to use to approximate the rate. This would require us to consider not just the maximum possible extension of single-stranded domains, but also the probability that they are extended to given lengths e.g., using a worm-like chain biophysical model [19].

Indeed, the biophysical model that we have used in this paper is deliberately simple, so that the structures can be compiled directly to tractable constraint problems. Some of our assumptions seem necessary for the technique to work, such as the assumption that hybridized domains are co-localized in space, because to do otherwise would require additional constraints on the angles of certain domains to ensure that they are parallel, which would greatly complicate the constraint problems to be solved. Other assumptions, such as the condensing of single-stranded domains, make sense as a simple optimization which reduces the number of variables present in the constraint problem. An interesting direction for future research will be to consider further optimizations that could speed up the analysis of structures and enable larger structures to be analyzed: one possibility might be to prune constraints that arise from parts of the

structure that are not actually essential to determining whether the structure is plausible or not, another might be to remove domain length constraints that are technically redundant because the length of those domains has already been constrained by hybridization to some other domain. The Z3 solver itself includes facilities for the simplification of constraint problems, which could be directed in specific ways to optimize solving the type of problems produced by our translation process. Another possibility might be to explore alternative methods for deciding plausibility that do not involve SMT solving, e.g., employing sampling, search, or genetic algorithms over the space of variable instantiations to try to find an instantiation that satisfies all of the constraints. These alternative approaches could enable larger constraint problems, corresponding to more sophisticated molecular structures, to be tackled.

While our biophysical model is clearly not suited to in-depth biophysical investigations of nucleic acid dynamics, it *is* useful as a simplified model for (automated) preliminary investigations, that can be used to rule out certain designs and to guide more in-depth analyses, e.g., using coarse-grained simulation models [20]. The key advantage of an automated structural analysis tool such as ours is that it may significantly reduce the number of iterative experimental design cycles that must be carried out in order to successfully implemented a tethered molecular circuit, thereby reducing wasted labour and reagents expended on unsuccessful designs.

A fruitful avenue for further research will be to determine how much more realistic our biophysical model can be made without sacrificing ease-of-use or constraint solving performance. One possibility would be to include additional constraints, e.g., to enforce a minimum extension on single-stranded domains, or to require that the lengths of double-stranded domains must be within some factor of the true value, to model slight fluctuations in the duplex structure. (The latter constraint could be imposed at no extra cost because our floating-point constraint expansion already includes an error term.) Another possibility would be to constrain the angles that can be formed at the junctions between domains, but this could adversely impact performance.

It would be an interesting future research direction to investigate whether our biophysical model is conservative, in the sense that if two strands cannot interact in our model, then they will not be able to interact in a more realistic model. It seems likely that certain assumptions would need to be made for this statement to be true, such as rigidity of the underlying origami tile. Furthermore, the biophysical constants, such as the internucleotide distances, would need to be chosen judiciously. It is worth pointing out that the converse may not hold, i.e., our system may permit certain reactions that a more detailed model might rule out, e.g., due to steric effects that might present the DNA from actually adopting the required conformation.

7.3 Extensions

The discussion above is phrased entirely in terms of tethered structures, however, similar techniques could be used in non-tethered but non-trivial DNA structures, e.g., those studied in our previous work on strand graphs [15]. In this context, we would simply pick an arbitrary position from the structure to serve as the origin and would omit the $\mathtt{tc}(P)$ term from Definition 5 (which would also remove the constraints that prohibit negative z-coordinates). This would enable us to use constraint solving to find reachable

domain bindings in a complex DNA nanostructure, with the caveat that issues such as potential steric hindrance between strands would not be modelled. Existing free energy approaches allow calculation of base-pairing probabilities [21] for nucleic acid nanostructures that include certain restricted classes of pseudoknots in polynomial time. As before, this thermodynamic model will be more physically accurate than our model, as it is based on thermodynamic experiments at the individual base-pair level. However, our work could similarly be used as a precursor to a more detailed analysis using free energy methods.

It is worth noting that, in all of the examples presented above, it is the case that for any position $\mathit{tether}(c_x, c_y, c_z)$, then $c_x = r_x$, $c_y = r_y$, and $c_z = 0$, for some constants r_x and r_y. It is also the case that for any position $\varepsilon(c_x, c_y, c_z)$, then $c_x = x$, $c_y = y$, and $c_z = z$, for some variables x, y, and z. That is, the locations of all tethers are fixed and are always in the $z = 0$ plane, and all untethered domain junctions are completely free to vary (subject to the other constraints). However, our approach is more general than this—in principle, we could leave certain tether coordinates unspecified and let the constraint solver search for satisfying instantiations. Thus, this work lays the foundation for the development of further automated tools for tethered molecular computing systems, such as circuit synthesis tools that take a logic function as input and return a layout for a molecular circuit that will implement that logic function without crosstalk. In this context, our constraint-based system would be used to determine the positioning and spacing of components that would be required to make the products of the desired reactions plausible but the products of the undesired reactions implausible. Ideally, any circuit synthesis algorithm of this type would keep the constraint problems that must be solved as small as possible for maximum efficiency, and might use some kind of global optimization procedure or genetic algorithm to converge towards a suitable design. We could also allow non-zero z-coordinates for tether locations, which would correspond to the underlying surface being a 3D nanostructure as opposed to a flat tile.

Acknowledgments. This material is based upon work supported by the National Science Foundation under grants 1525553, 1518861, and 1318833. The authors thank Neil Dalchau and Rasmus Petersen for productive discussions.

References

1. D. Y. Zhang and G. Seelig. Dynamic DNA nanotechnology using strand-displacement reactions. *Nat Chem*, 3(2):103–113, 2011.
2. Y.-J. Chen, N. Dalchau, N. Srinivas, A. Phillips, L. Cardelli, D. Soloveichik, and G. Seelig. Programmable chemical controllers made from DNA. *Nat Nanotechnol*, 8:755–762, 2013.
3. D. Soloveichik, G. Seelig, and E. Winfree. DNA as a universal substrate for chemical kinetics. *PNAS*, 107(12):5393–5398, 2010.
4. M. Cook, D. Soloveichik, E. Winfree, and J. Bruck. Programmability of chemical reaction networks. In *Algorithmic Bioprocesses*, pages 543–584. Springer, 2009.
5. M. R. Lakin, S. Youssef, L. Cardelli, and A. Phillips. Abstractions for DNA circuit design. *JRS Interface*, 9(68):470–486, 2012.
6. M. R. Lakin, S. Youssef, F. Polo, S. Emmott, and A. Phillips. Visual DSD: a design and analysis tool for DNA strand displacement systems. *Bioinformatics*, 27(22):3211–3213, 2011.

7. L. Qian and E. Winfree. Scaling up digital circuit computation with DNA strand displacement cascades. *Science*, 332:1196–1201, 2011.
8. P. W. K. Rothemund. Folding DNA to create nanoscale shapes and patterns. *Nature*, 440:297–302, 2006.
9. G. Tikhomirov, P. Petersen, and L. Qian. Programmable disorder in random DNA tilings. *Nat Nanotechnol*, 12:251–259, 2017.
10. H. Bui, V. Miao, S. Garg, R. Mokhtar, T. Song, and J. Reif. Design and analysis of localized DNA hybridization chain reactions. *Small*, 13(12):1602983, 2017.
11. R. A. Muscat, K. Strauss, L. Ceze, and G. Seelig. DNA-based molecular architecture with spatially localized components. In *Proceedings of ISCA 13*, 2013.
12. N. Dalchau, H. Chandran, N. Gopalkrishnan, A. Phillips, and J. Reif. Probabilistic analysis of localized DNA hybridization circuits. *ACS Synth Biol*, 4(8):898–913, 2015.
13. A. S. Walsh, H. Yin, C. M. Erben, M. J. A. Wood, and A. J. Turberfield. DNA cage delivery to mammalian cells. *ACS Nano*, 5(7):5427–5432, 2011.
14. M. R. Lakin, R. Petersen, K. E. Gray, and A. Phillips. Abstract modelling of tethered DNA circuits. In *Proceedings of DNA20*, volume 8727 of *LNCS*, pages 132–147. Springer, 2014.
15. R. L. Petersen, M. R. Lakin, and A. Phillips. A strand graph semantics for DNA-based computation. *Theor Comput Sci*, 632:43–73, 2016.
16. L. de Moura and N. Bjørner. Z3: An efficient SMT solver. In *Proceedings of TACAS 2008*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008.
17. D. Jovanović and L. de Moura. Solving non-linear arithmetic. In *Proceedings of IJCAR 2012*, volume 7364 of *LNCS*, pages 339–354. Springer, 2012.
18. C. Grun, J. Werfel, D. Y. Zhang, and P. Yin. DyNAMiC Workbench: an integrated development environment for dynamic DNA nanotechnology. *JRS Interface*, 12:20150580, 2015.
19. A. J. Genot, D. Y. Zhang, J. Bath, and A. J. Turberfield. Remote toehold: a mechanism for flexible control of DNA hybridization kinetics. *J Am Chem Soc*, 133:2177–2182, 2011.
20. J. P. K. Doye, T. E. Ouldridge, A. A. Louis, F. Romano, P. Šulc, C. Matek, B. E. K. Snodin, L. Rovigatti, J. S. Schreck, R. M. Harrison, and W. P. J. Smith. Coarse-graining DNA for simulations of DNA nanotechnology. *Phys Chem Chem Phys*, 15:20395–20414, 2013.
21. R. M. Dirks and N. A. Pierce. An algorithm for computing nucleic acid base-pairing probabilities including pseudoknots. *J Comput Chem*, 25:1295–1304, 2004.