Programming Biomolecules That Fold Greedily During Transcription*

Cody Geary[1], Pierre-Étienne Meunier[2], Nicolas Schabanel[3], and Shinnosuke Seki[4]

1 **California Institute of Technology, Pasadena, CA, USA.** codyge@gmail.com.
2 **Department of Computer Science, Aalto University, Finland and Aix Marseille Université, CNRS, LIF UMR 7279, 13288, Marseille, France.** http://users.ics.aalto.fi/meunier/
3 **CNRS, Université Paris Diderot, France and IXXI, Université de Lyon, France.** http://www.irif.univ-paris-diderot.fr/users/nschaban/
4 **University of Electro-Communications, Tokyo, Japan.** s.seki@uec.ac.jp

Abstract

We introduce and study the computational power of Oritatami, a theoretical model to explore greedy molecular folding, by which a molecule begins to fold before awaiting the end of its production. This model is inspired by a recent experimental work demonstrating the construction of shapes at the nanoscale by folding an RNA molecule during its transcription from an engineered sequence of synthetic DNA.

An important challenge of this model, also encountered in experiments, is to get a single sequence to fold into different shapes, depending on the surrounding molecules. Another big challenge is that not all parts of the sequence are meaningful for all possible inputs. Hence, to prevent them from interfering with subsequent operations in the Oritatami folding pathway we must structure the unused portions of the sequence depending on the context in which it folds.

Next, we introduce general design techniques to solve these challenges and program molecules. Our main result in this direction is an algorithm that is time linear in the sequence length that finds a rule for folding the sequence deterministically into a prescribed set of shapes, dependent on its local environment. This shows that the corresponding problem is fixed-parameter tractable, although we also prove it NP-complete in the number of possible environments.

1998 ACM Subject Classification F.1.1 Models of Computation, J.3 Life and Medical Sciences

Keywords and phrases Natural computing, Self-Assembly, Molecular Folding

Digital Object Identifier 10.4230/LIPIcs.MFCS.2016.43

1 Introduction

The process by which one-dimensional sequences of nucleotides or amino-acids acquire their complex three-dimensional geometries, which are key to their *function*, is a major puzzle of biology today. Understanding molecular folding will not only shed light on the origin and functions of the molecules existing in nature, it will also enable us to *control* the process

* C. Geary is a Carlsberg Postdoctoral Fellow supported by the Carlsberg Foundation and is supported by USNSF Grant no. 1317694 (http://molecular-programming.org). N. Schabanel is supported by Grants ANR-12-BS02-005 RDAM and IXXI-MOLECAL. S. Seki is in part supported by: the Academy of Finland; Postdoctoral Researcher Grant 13266670/T30606; JST Program, MEXT, Japan, No. 6F36; and JSPS Grants No. 15H06212 and No. 16H05854.

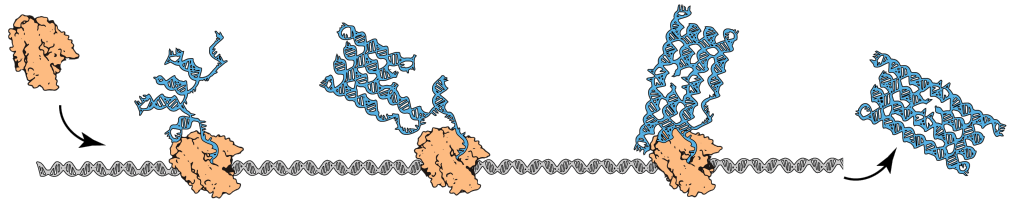


Figure 1 An RNA molecule folding over itself while being transcribed, as the experiments in [14].

more finely, and engineer artificial molecules with a wide range of uses, from performing missing functions inside living organisms, to producing precisely targeted drugs.

Biomolecular nano-engineering includes DNA self-assembly, which gave rise to an impressive number of successful experimental realizations, from arbitrary 2D shapes [23] to molecule cyclic machines [28], or counters [11]. First pioneered by Seeman [24], DNA nanotechnologies only really started to take off once a computer science model was devised by Winfree [26] to *program* molecular self assembly in a computer science way.

Since then, many models have been designed to refine different features of experiments: hierarchical self-assembly [4, 5], modeling the absence of a *seed*, kinetic tile assembly [26, 13], 3D and probabilistic tile assembly [6], among others.

The applications of DNA are limited by the high-fidelity of DNA base-pairing. The elaborate structures that researchers are able to produce with DNA are typically formed by the thermodynamically-driven assembly process of *annealing*, where many DNA strands are brought near equilibrium by heating them up to a high temperature and then cooling them down at a controlled rate, a folding process that is essentially incompatible with living cells.

By contrast, the RNA nanostrutures are designed to mimic the natural folding process, and are designed to fold under conditions that are cell-like. However, their assembly process is harder to program: intuitively, the shape depends on both the sequence and the environment, and the sequence is read *linearly*, independent from the environment. This contrasts with more classical programming models such as Turing machines, or even tile assembly, which are able to *jump* to different parts of the program depending on the input.

Another important difficulty is that even predicting the final shape of a sequence is still the center of active research, especially for proteins [29, 17, 18, 8, 21, 22, 16]

In particular, a large body of computer science literature focused on energy optimization, one of the main drivers of folding. For example, in different variants of the *hydrophobic-hydrophilic (HP) model* [9], it has been shown that the problem of predicting the most likely geometry (or *conformation*) of a sequence is NP-complete [25, 20, 2, 3, 7, 1], both in two and three dimensions, and in different variants of the model.

A few years ago, the *kinetics* of folding, which is the step-by-step dynamics of the reaction, has been demonstrated by biochemists to play a role in the final shape of molecules [15], even a *prevalent role* in the case of RNA [12]. In recent experimental results [14], researchers have even been able to *control* this mechanism to engineer their own shapes out of RNA.

This paper introduces a new model of RNA folding, intended to capture the kinetics of folding and model the experiments in [14]. In particular, it focuses on the *co-transcriptional* nature of RNA folding, which is the fact that, in real conditions, molecules fold while being transcribed (see Figure 1): in computer science terms, the folding process is a *local energy optimization*, or otherwise put, a *greedy algorithm*.

The first experimental results have used a standard benchmark: making simple shapes, such as squares (as shown for instance on Figure 2). With the new model introduced in

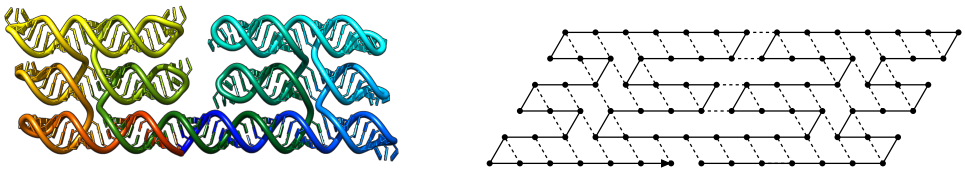


Figure 2 The design of a rectangle, co-transcriptionally folded with RNA, and the corresponding path on the triangular lattice, where each bead corresponds to two to four nucleotides.

this paper, our goal is twofold: first, explore the engineering possibilities of this mechanism, in order to make arbitrary shapes and structures. Then, the other aim of our study is to understand the complexity of sequence operations, to understand the computational processes which led to the creation of complex molecular networks.

Main contributions. In our model, called Oritatami, we consider a sequence of "beads", which are abstract basic components, standing for nucleotides or even sequences of nucleotides (also called *domains*). In Oritatami, only the latest produced beads of the molecules are allowed to move in order to adopt a more favorable configuration. The folding is driven by the respective attraction between the beads.

Our main construction is a *binary counter*. Counters are an essential component of many sophisticated constructions in biological computing, in particular in tile assembly [10, 19]. Counters are also an important benchmark in experiments [11].

▸ **Theorem 1.** *There is a fixed periodic sequence* $0, 1, \ldots, 59, 0, 1, \ldots$ *of period 60 whose rule is given in Fig. 11, which, when started from a seed encoding an integer* x *in binary with at most* $2k + 1$ *bits for some* k*, folds into a structure encoding* $x + 1$*,* $x + 2$*,* $\ldots$*,* $2^{2k+1} - 1$*, on the successive rows of the triangular grid.*

We *prove the correctness* of this construction by designing an abstract module system to handle the complexity of the base mechanism of the model, which is about as low-level as assembly code in more standard computing models.

We then show a generic construction method in this model, which we applied to automate parts of the design of the counter. Moreover, this result helps understanding the computational complexity of sequence programming. Precisely, we prove two results in this direction:

▸ **Theorem 2.** *Designing a single sequence that folds into different target shapes in a set of surrounding environments, is NP-complete in the number of environments.*

More surprisingly, it turns out that there is an algorithm to solve this problem in time *linear in the length of the sequence*. This algorithm is also practical, as we were able to use it to find sequences for our main construction:

▸ **Theorem 3.** *The sequence design problem is FPT with respect to the length* ℓ *of the sequence: there is an algorithm linear in* ℓ *(but exponential in the number of environments) to design a single sequence that folds into the target shapes in the given environments.*

2 Model and Main Results

2.1 Model

Oritatami system. Oritatami is about the folding of finite sequences of beads, each from a finite set B of *bead types*, using an attraction rule , on the triangular lattice graph $\mathbb{T} = (\mathbb{Z}^2, \sim)$

where $(x, y) \sim (u, v)$ if and only if $(u, v) \in \{(x-1, y), (x+1, y), (x, y+1), (x+1, y+1), (x-1, y-1), (x, y-1)\}$.

A *conformation* c of a sequence $w \in B^*$ is a self-avoiding path of length ℓ labelled by w in $\mathbb{T}$, i.e. a path whose vertices $c_1, \ldots, c_\ell$ are pairwise distinct and labelled by the letters of w. A *partial conformation* of a sequence w is a conformation of a prefix of w. For any partial conformation c of some sequence w, an *elongation* of c by k beads is a partial conformation of w of length $|c| + k$. We denote by $\mathcal{C}_w$ the set of all partial conformations of w (the index w will be omitted when the context is clear). We denote by $c^{\rhd k}$ the set of all elongations by k beads of a partial conformation c of a sequence w and by $c^{\lhd k}$ the singleton containing the prefix of length $|c| - k$ of c.

An *Oritatami system* $\mathcal{O} = (p, \heartsuit, \delta)$ is composed of (1) a (possibly infinite) *primary structure* p, which is a sequence of *beads*, of a type chosen from a finite set B, (2) an *attraction rule*, which is a symmetric relation $\heartsuit \subseteq B^2$ and (3) a parameter δ called the *delay time*.

Given an attraction rule $\heartsuit$ and a conformation c of a sequence w, we say that there is a *bond* between two adjacent positions c_i and c_j of c in $\mathbb{T}$ if $w_i \heartsuit w_j$. The *energy* of a conformation c of w, written $\mathrm{E}(c)$, is the negation of the number of bonds within c: formally, $\mathrm{E}(c) = -|\{(i, j) : c_i \sim c_j,\ j > i+1,\ \text{and } w_i \heartsuit w_j\}|$.

Oritatami dynamics. A *dynamics* for a sequence w is a function $\mathcal{D} : 2^{\mathcal{C}} \to 2^{\mathcal{C}}$ such that for all subset S of partial conformations of length ℓ of w, $\mathcal{D}(S)$ is a subset of the elongations by one bead of the partial conformations in S (thus, partial conformations of length $\ell + 1$).

Given an Oritatami system $\mathcal{O} = (p, \heartsuit, \delta)$ and a *seed conformation* σ of a seed sequence s of length ℓ, the set of partial conformations of the primary structure p at time t under dynamics $\mathcal{D}$ is $\mathcal{D}^t_{sp}(\{\sigma\})$, [1] i.e. the set of all elongations by t beads of the seed conformation prolongated by the primary structure according to dynamics $\mathcal{D}$.

We explore greedy folding dynamics where only the most recently transcribed beads can move, all other beads remain in place. These are controlled by integer parameter δ (in this article, $\delta \leqslant 4$). Several dynamics could model the "greedy" nature of the process. We choose the following dynamics, called the *hasty dynamics*:

The hasty dynamics does not question previous choices but chooses the energy-minimal positions for the δ last beads among all elongations of the previously adopted partial conformations. It lets the $\delta - 1$ already placed last beads where they are and abandons the extension of a conformation if no extension with the newly transcribed bead allows to reach a lowest energy conformation available for the δ last beads. Formally, $\mathcal{H}$ starts from a set of partial conformations, elongates each of them by one bead, and keeps the elongated conformations that have minimal energy among those who share the same prefix of length $|\sigma| + t - \delta$:

$$\mathcal{H}(S) = \bigcup_{\gamma \in S^{\lhd(\delta-1)}} \left(\operatorname*{arg\,min}_{c \in (S^{\rhd 1}) \cap (\gamma^{\rhd \delta})} \mathrm{E}(c) \right)$$

An Oritatami system $\mathcal{O} = (p, \heartsuit, \delta)$ is *deterministic* for dynamics $\mathcal{D}$ and seed σ of sequence s if for all $i \geqslant 1$, the position of the i-th bead of p is deterministic at time $i - 1 + \delta$, i.e. if for all $i \geqslant 1$, $|\{c_{|\sigma|+i} : c \in \mathcal{D}^{i-1+\delta}_{sp}(\{\sigma\})\}| = 1$. We say that $\mathcal{O}$ *stops* at time t with seed σ and dynamics $\mathcal{D}$ if $D^t_{sp}(\{\sigma\}) = \varnothing$ and $\mathcal{D}^z_{sp}(\{\sigma\}) \neq \varnothing$ for $z < t$. The folding process may

[1] Given two words $a, b \in B^*$, we denote by ab their concatenation.

only stop because of a geometric obstruction (no more elongation are possible because the conformation gets trapped in a closed area).

3 Folding a binary counter

3.1 General idea of the construction

Our construction works with $\delta = 4$. The counter is implemented by folding the periodic sequence of bead types $0, 1, \ldots, 58, 59, 0, 1, \ldots$ with period 60. Our construction proceeds in zig-zags as the classic implementation of a counter with a sweeping Turing machine whose head goes back and forth between the two ends of the coding part of the tape. Each pass is 3-rows thick and folds each part of the molecule into a parallelogram of size 4×3 or 6×3 except for the last and the first parts of each pass which are folded into parallelograms of size 3×6 to accomplish the U-turn downwards and start the next pass. The *zig pass*, folding three rows at a time from right to left, computes the carry propagation in the current value of the counter. The *zag pass*, folding three rows at a time from left to right, writes down the bits of the newly incremented value, and gets the folding to resume at the right-hand side of the conformation.

The molecule is semantically divided into 4 parts, called *modules*:

- Module A (beads 0–11, in blue in all figures): the First Half-Adder
- Module B (beads 12–29, in red in all figures): the Left-Turn module
- Module C (beads 30–41, in blue in all figures): the Second Half-Adder
- Module D (beads 42–59, in red in all figures): the Right-Turn module

Encoding. The current value of the counter is encoded in standard binary with the most significant bit to the left. Each bit is encoded into a specific folding of the modules A and C of the molecule in the rows corresponding to a zag pass: namely folding A0 and C0 for 0, and A1 and C1 for 1. During the zig pass, the *value of the carry* is encoded by the position of the molecule when it starts to fold Module A or C: `carry` $= 0$ if Module A or C starts to fold in the top row; `carry` $= 1$ if Module A or C starts to fold from the bottom row.

In the zig pass (←), modules A and C "read" from the row above the value encoded into the folding in the row above during the previous zag-phase (or in the seed conformation for the first zig pass), and fold into a shape (called a *brick*, see Section 4) A00, A10, A01, A11 or C00, C10, C01, C11 accordingly where Axc is the brick corresponding to the case where x is the bit read in the row above and c is the carry. In the zig pass, modules B and D just propagate the carry value (0 or 1, i.e. start from top or bottom row) output by the preceding module A or C to the next.

When the zig pass reaches the leftmost part of the row on top, the beads there forces the module B to adopt the Left-turn shape which reverses the folding direction and starts the next zag pass.

In the zag pass (→), modules A and C "read" the bricks above Axc or Cxc and folds into the bricks that encodes the corresponding bits, namely Ay or Cy where $y = (x + c) \mod 2$. There are no carry propagation and all the modules B and D fold into the same brick B2 or D2 in this pass.

When the molecule reaches the rightmost part of the row on top of it, the special beads there force the module D to fold into the Right-turn brick which reverses the folding direction and starts the next zig pass.

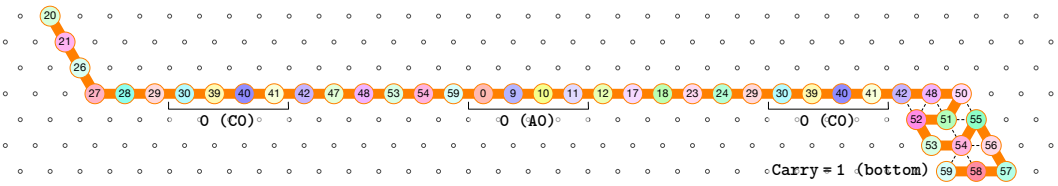


Figure 3 The seed conformation for the 3-bits counter encoding the three bits 000 as the initial value of the counter.

3.2 The first two passes of the folding

Let's run the first passes of the 3 bits counter to get acquainted with the process.

The seed conformation. is shown in Fig. 3. The seed conformation for the $(2k+1)$-bit counter is composed of $4k+3$ parts:

- The first part $20 \overset{\searrow}{\text{SE}} 21 \overset{\searrow}{\text{SE}} 26 \overset{\searrow}{\text{SE}} 27 \overset{\rightarrow}{\text{E}} 28 \overset{\rightarrow}{\text{E}} 29$, made of beads from Module B, encodes a sequence that will trigger the carriage return at the end of the next zig pass.
- The central part consists in k repetitions of the same sequence of 4 patterns, plus an extra repetition of the first pattern at the end (the central part consists thus in $4k+1$ parts in total):
  - $30 \overset{\rightarrow}{\text{E}} 39 \overset{\rightarrow}{\text{E}} 40 \overset{\rightarrow}{\text{E}} 41$ encoding a bit 0 using beads from Module C,
  - followed by $42 \overset{\rightarrow}{\text{E}} 47 \overset{\rightarrow}{\text{E}} 48 \overset{\rightarrow}{\text{E}} 53 \overset{\rightarrow}{\text{E}} 54 \overset{\rightarrow}{\text{E}} 59$ encoding nothing but padding using beads from Module B,
  - followed by $0 \overset{\rightarrow}{\text{E}} 9 \overset{\rightarrow}{\text{E}} 10 \overset{\rightarrow}{\text{E}} 11$ encoding a bit 0 using beads from Module A,
  - followed by $12 \overset{\rightarrow}{\text{E}} 17 \overset{\rightarrow}{\text{E}} 18 \overset{\rightarrow}{\text{E}} 23 \overset{\rightarrow}{\text{E}} 24 \overset{\rightarrow}{\text{E}} 29$ encoding nothing but padding using beads from Module D.

  Note the symmetry by a shift of 30 of the beads values in the patterns involving Modules A and C, and Modules B and D.
- The last part $42 \overset{\rightarrow}{\text{E}} 48 \overset{\rightarrow}{\text{E}} 50 \overset{\swarrow}{\text{SW}} 51 \overset{\leftarrow}{\text{W}} 52 \overset{\searrow}{\text{SE}} 53 \overset{\rightarrow}{\text{E}} 54 \overset{\nearrow}{\text{NE}} 55 \overset{\searrow}{\text{SE}} 56 \overset{\searrow}{\text{SE}} 57 \overset{\leftarrow}{\text{W}} 58 \overset{\leftarrow}{\text{W}} 59$, made of beads from Module D, encodes a sequence that will first initiate the next zig pass and later trigger the carriage return at the end of the next zag pass.

Note that the seed conformation ends at the bottom row of the upcoming zig pass, which encodes thus that initially the carry is 1.

The first zig pass (←). Each zig pass starts with a carry equal to 1, i.e. starts folding from the bottom row. In the first zig pass, the first module A (see Figure 4) folds into the brick A01, encoding the bit $1 = 0 + 1$ with no carry propagation, as a consequence of the carry being 1 and of reading the first bit, 0, from the seed above. Note that the folding A01 ends at the top row, encoding that the carry is now 0. The reading of the bit from the seed is accomplished by the way the module binds to the seed which shapes the module accordingly as we will see in details in Section 3.3.

Then, as illustrated in Fig. 5, the next modules B, C, D, and A fold into shapes B0, C00, D0 and A00 respectively: B0 and D0 meaning that no carry is propagated (they start and end on the top row); and C00 and A00 meaning that the (input) carry is 0 and the bit read from the seed is 0.

Finally, as illustrated in Fig. 6, the last module, B, of the zig pass binds to the 3-beads long carriage-return pattern at the leftmost part of the seed and folds into the shape BT

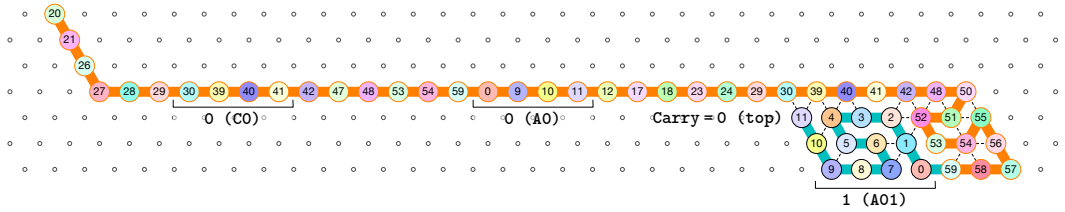


Figure 4 The folding of the first module, A: starting with a carry 1, encoded by the position of the first bead (on the bottom row), this module "reads" a 0 from the seed by binding to the seed, and folds into A01, encoding a 1 with no carry propagation, as encoded by the position of the last bead (on the top row of the module).

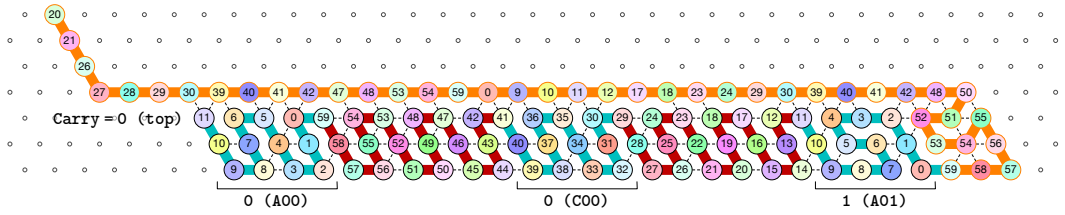


Figure 5 The folding of the central part of the first zig pass in the 3-bits counter.

conducing the molecule to go down by 6 rows, reverse direction and start the following zag pass. Note that the bottom of the shape BT contains the exact same carriage-return pattern.

The first zag pass (→). The zag pass is mostly a normalization pass as illustrated in Fig. 7–8. It progresses from left to right and computes the new value of each bit by rewriting each shape A00 and A11 as C0, C00 and C11 as A0, A10 and A01 as C1, and C10 and C10 as A0. Shapes A0 and C0 encode 0, and Shapes A1 and C1 encode 1, both to be read during the upcoming zig pass. Modules B and D just fold into the shapes B2 and D2 respectively, encoding nothing but padding.

Finally, as illustrated in Fig. 9, the last module, D, of the zag pass binds to the 3-beads long carriage-return pattern in the rightmost part of the seed and folds into the shape DT conducing the molecule to go down by 6 rows, reverse direction and start the next zig pass. Note that, as for the shape BT, the bottom of the shape DT contains the exact same carriage-return pattern.

Figure 10 shows the 3-bits counter folded upto the value 3 = 011 in binary. One can observe the shape A11 in the second zig pass. A11 corresponds to reading a 1 with a carry 1 which propagates the carry: indeed, the folding ends at the bottom row which propagates the carry to the next module C which folds into C01 as it reads a 0 from above with carry 1. Note that shape A11 is then rewritten as C0 in the following zag pass below.

3.3 How does computation take place: modules, functions, states and environment

Each module A, B, C and D implement various "functions" that are "called" when the molecule is folded. Which function is called depends on two things:

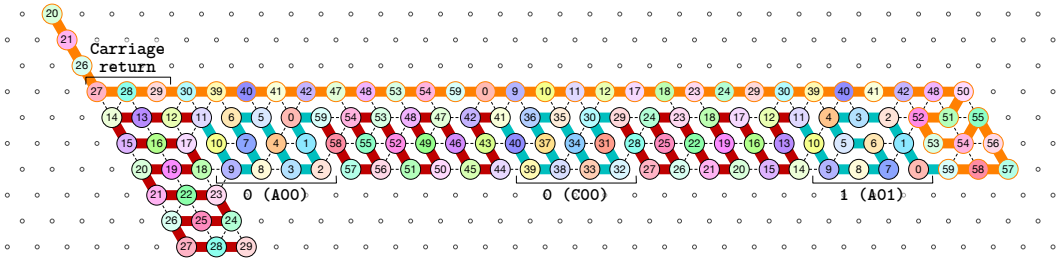


Figure 6 In our construction, the leftmost three beads of any conformation are different from the other beads the left U-turn module binds to inside the zig or zag pass: when the left U-turn module folds next to these bead types, it "triggers" the production of an actual U-turn.

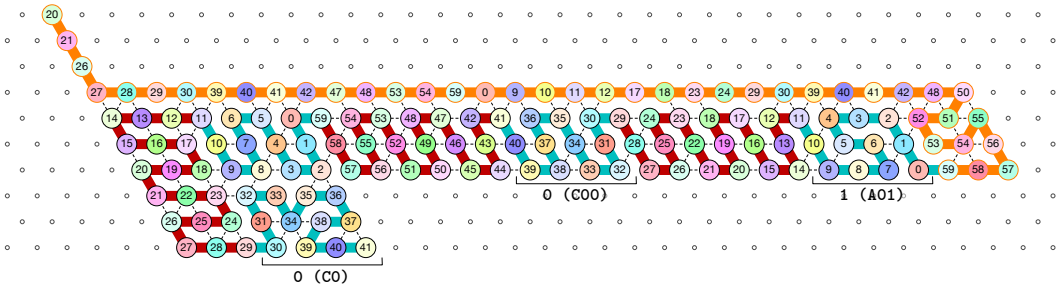


Figure 7 During the zag pass, all modules start from the bottom row, computing the value of each new bit by rewriting shapes A00 and A11 as C0, C00 and C11 as A0, A10 and A01 as C1, and C10 and C10 as A1.

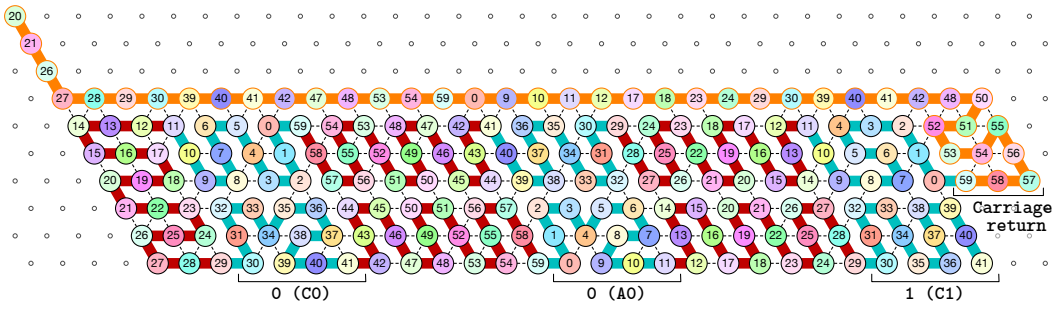


Figure 8 At the end of the first zag pass, the new value of each bit have been encoded into shapes: A0 or C0 for bits equal to 0, A1 or C1 for bit equal to 1.

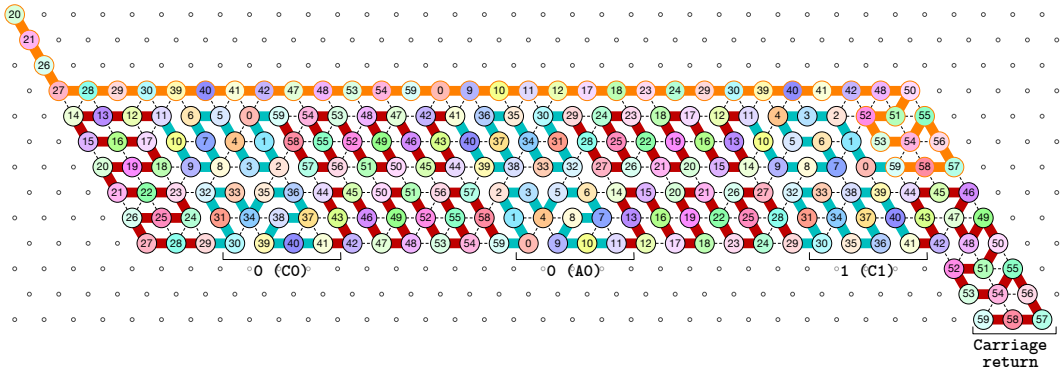


Figure 9 Finally, at the end of the first zag pass, the last module D binds to the carriage-return pattern in the seed and fold into the shape DT to accomplish the right U-turn from which the next zig pass can start.

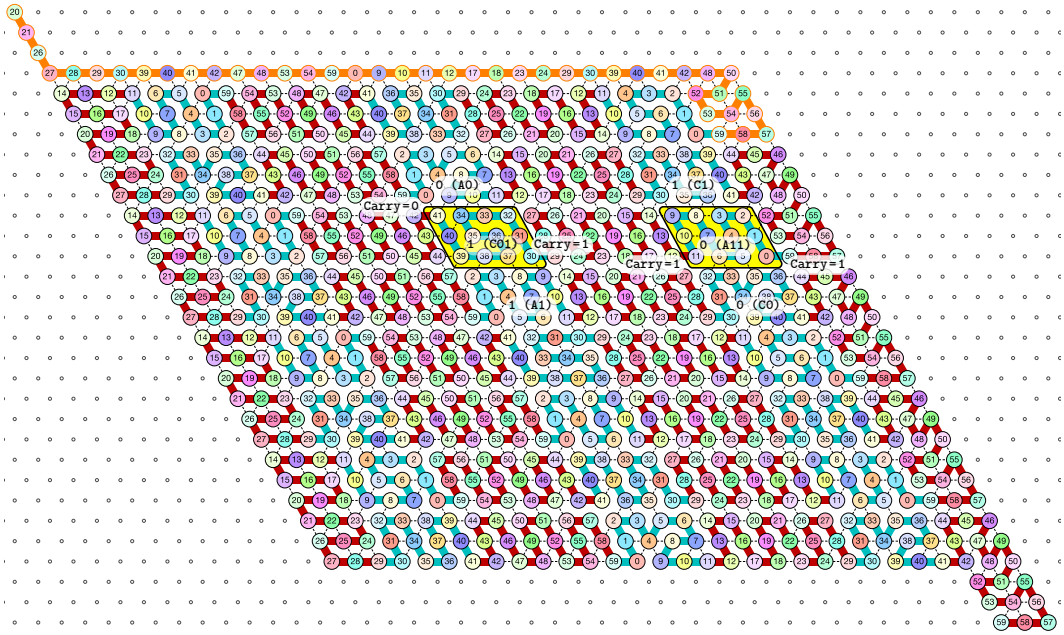


Figure 10 The folding of the 3-bits counter upto value $4 = 100$ in binary. Observe the carry propagation in the second zig pass.

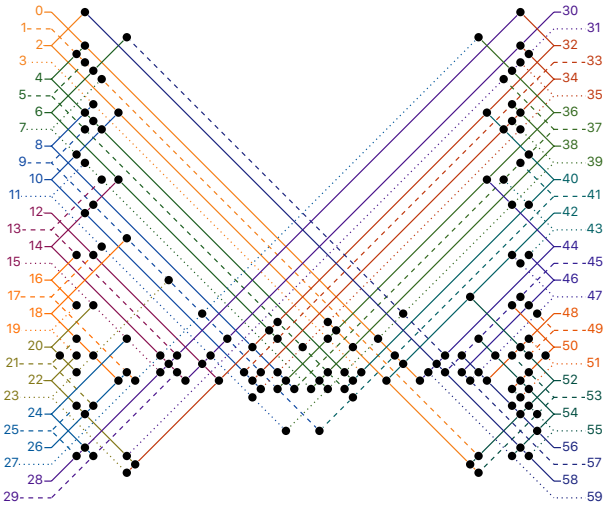


Figure 11 *The rule* 💗 *of the Counter Oritatami system*: in this diagram, we have b 💗 b' iff there is a bullet • at the intersection of one the two lines coming from b and from b'; for instance, we have 4 💗 8 but not 4 💗 7.

the current "state" of the molecule: here, the *state* is whether the carry is 0 or 1. As mentionned earlier this is encoded in the position of the molecule when the module starts to fold: it starts in the top row if the carry is 0; in the bottom row if the carry is 1.

the local environment of the molecule: the *environment*, i.e. the beads already placed around the current area where the folding take place, acts as the memory in the computation.

The position where the folding of a module starts, determines which beads of a given module will be exposed to and interact with the environment. Then, by creating bonds (or not) with the environment, each module will adopt a specific shape. Therefore, the possible binding schemes will be different depending of this initial position. Similarly, depending on the beads already placed in the environment, the part of the module exposed to it will adopt one form or another depending on how many bonds it can create with the environment. Adopting the language of computer science: the position at which a module starts to fold, determines which *function* of the module is called; the function then *reads the input* encoded by the beads already placed in the environment.

Fig. 12 provides a precise description on how the function of the Half-Adder Module C are implemented in the zig pass. As the zig pass goes from right to left, the figure is meant to be read from right to left. In the zig pass, Module C implements two functions: 1) Add 1 to the bit above and propagate the carry if needed, or 2) Copy the bit above unchanged. Add is called if the carry is 1 at the beginning of the folding and Copy is called if the carry is 0. The following step-by-step description of the folding explains how:

Beads 30-33 (rightmost column in Fig. 12): *if the carry is* 0 *at start,* then bead 30 is able to bind with beads 11 and 12 from the environment and depending on whether the input encodes a bit 0 or 1, bead 32 will be able to bind to bind either to 28 or to 5 and 6 respectively. *Whereas if the carry is* 1*,* then bead 30 cannot reach beads 11 and 12. Thus, these are beads 31 and 32 that will bind with beads 10 and 12 from the environment, giving to the molecule a completely different shape.

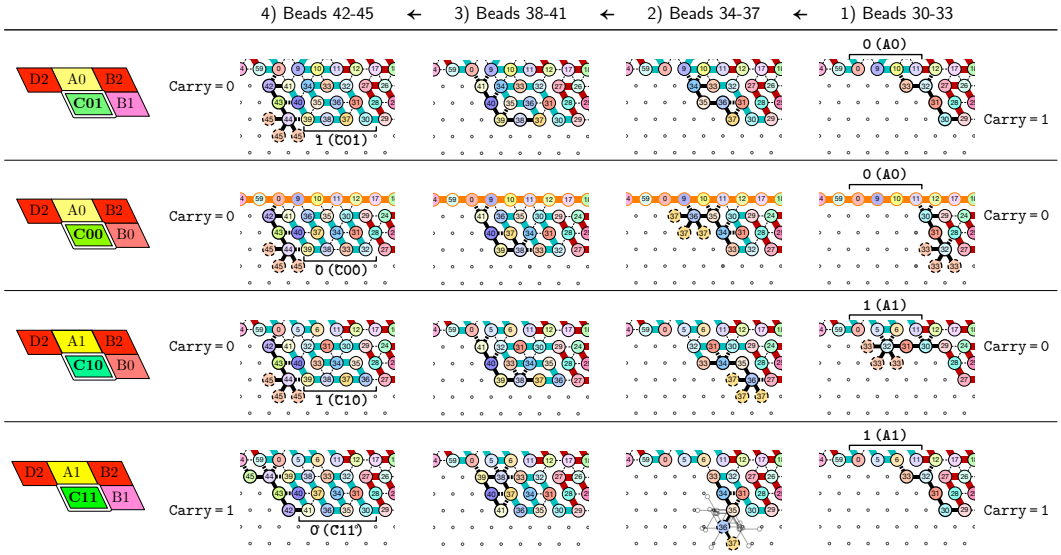


Figure 12 An illustration of how the module C applies a different function which results in different foldings according to the initial state of the molecule (carry = 0 or 1) at the beginning of the folding of the module, and to the environment (the bit 0 or 1 encoded) read above by the function. This figure is meant to be read from right to left (zig pass ←).

Beads 34-37 with carry = 0 at start: as bead 34 is attracted by both beads 30 and 31, the molecule folds upon itself similarly but with a different rotation depending on whether it had read a 0 or a 1 in the environement above: vertically if it has read a 0, horizontally if it has read a 1. Bead 36 attracted by beads 9 and 27 fixes the end of the tip in place leaving bead 37 free to move for now.

Beads 34-37 with carry = 1 at start: Bead 34 is attracted by beads 9 and 10 encoding a bit 0 above which allows beads 36 and 37 to bind with 31 as well, and but prefers to bind with 31 together with 35 otherwise. This induces two different shapes: the beginning of a "wave" pattern ( ) if the bit read above is 0; or the beginning of a "switchback" pattern ( ) if the bit read is 1.

Beads 38-41, without carry propagation (carry = 0, or carry = 1 and bit read = 0): in these three cases the folding of the beads 38-41 starts from the same position. As the environment is different for each of them, we could design the rule so that this part of the module prefers to adopt the same shape, climing along the part already folded to the top row to start the next module with a carry = 0.

Beads 38-41, with carry propagation (carry = 1 and bit read = 1): because the switchback pattern is upside down in this case, bead 37 stays low and bead 38 can firmly attach to the top with beads 5, 6 and 33, and the tip of the module folds downwards as 40 and 41 are attracted by 37. This ensures that the folding of the module ends at the bottom row, propagating the carry = 1 to the next module.

Note that the bottom rows of the four resulting foldings differ significantly: 39–38–37–30 for C01, 39–38–33–32 for C00, 39–38–37–36 for C10, and 41–36–35–30 for C11. This will allow to distinguish between them in the following zag pass to write the correct bit for the new value of the counter.

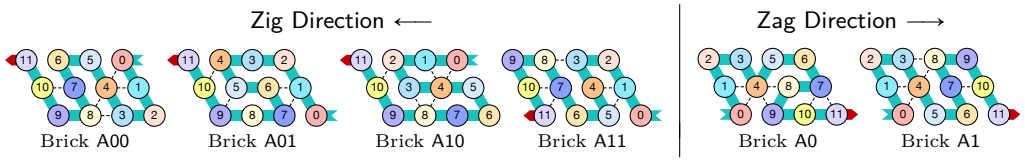


Figure 13 The six different bricks for Module A, First Half-Adder (beads 0–11).

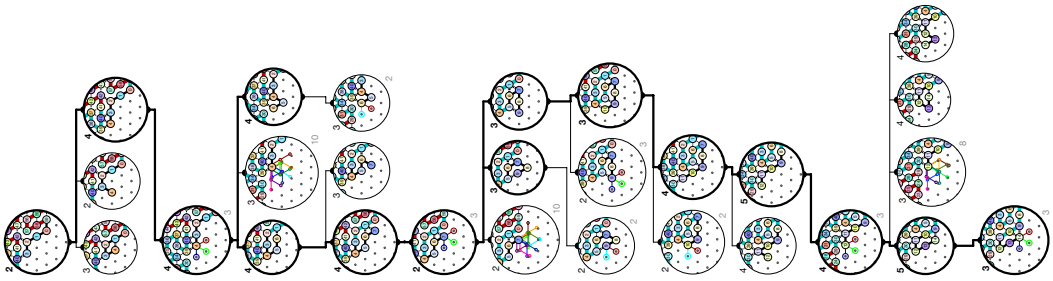
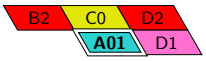


Figure 14 The folding certificate for the brick A01 in the environment:



4 Proof Sketch of Theorem 1

We call *bricks* the various conformations each module will adopt in the final folding, as they are the bricks upon which the whole folding is built. Designing the bricks is one of the biggest challenges when programming Oritatami. There are six bricks for Modules A and C: Axc and Cxc in the zig pass where $x, c \in \{0,1\}$ are the bit read from the brick above (Ax or Cx) and the (input) carry from the preceding module (Bc or Dc); and Ay and Cy in the zag pass where y is the bit written: namely $y = x + c \bmod 2$ if the brick above is Axc or Cxc. There are four bricks for Modules B and D: Bc' and Dc' in the zig pass where $c' \in \{0,1\}$ is the carry output by the preceding brick Cxc or Axc, namely $c' = x \wedge c$; B2 and D2 in the zag pass. Figure 13 lists the six bricks for Module A.

The proof works in two stages. First, we describe the final target folding and show that it implements correctly a binary counter: i.e. that the bricks implement correctly the relationships between the y, x, c and c' described above. Second, we show that indeed, each module folds into the expected brick in each given environment. For that matter, we produce a *folding certificate*, which shows for each step all the possible extension of the current conformation together with the number of bonds created (the number in the north-east corner), grouping together several extensions when they share a common path (the number of paths groupes are in the south-east corner). Figure 14 displays the folding certificate for Brick A01 surrounded by the bricks B2, C0, D2 and D1.

5 Rule design is NP-hard and FPT

Once we have agreed on the desired conformations in our construction, an important issue is to find an attraction rule such that a primary structure folds into its correct functions. The *rule design problem* consists in: given a delay time δ, a list of $n > 0$ seeds $\sigma_1, \sigma_2, \ldots, \sigma_n$, and a list of n conformations $c_1, c_2, \ldots, c_n$ of the same length ℓ, output an attraction rule 💗 such that for all $i \in \{1, 2, \ldots, n\}$, Oritatami system $\mathcal{O}_i = (s, \sigma_i, 💗, \delta)$ deterministically folds into conformation c_i, where $s = \langle 0, 1, \ldots, \ell - 1 \rangle$.

Theorem 2 (proof omitted) shows by a reduction from 3-SAT that this problem is NP-hard. However, Theorem 3 (proof omitted) shows using the locality of the bindings that it is fixed-parameter tractable with respect to the length of the sequence ℓ. Then, as n and δ are usually small for the designs we considered, we could use this algorithm to help us designing an attraction rule compatible with our brick designs.

6 Perspectives

The purpose of our new model is not to be entirely accurate with respect to phenomena observed in nature, but instead to start developing an intuition about the *kind of problem* that need to be solved in order to engineer RNA shapes, and later, even proteins.

In the future, a number of extensions of this model seem natural. In particular, extending it with a more realistic notion of *thermodynamics and molecular agitation*. Using existing works in molecular dynamics [27], would allow to explore stochastic optimization processes.

Acknowledgments. The authors thank Abdulmelik Mohammed, Andrew Winslow, Damien Woods for discussions and encouragements.

References

1 O. Aichholzer, D. Bremner, E. D. Demaine, H. Meijer, V. Sacristán, and M. Soss. Long proteins with unique optimal foldings in the H-P model. *Computational Geometry*, 25(1–2):139–159, 2003.

2 J. Atkins and W. E. Hart. On the intractability of protein folding with a finite alphabet of amino acids. *Algorithmica*, 25(2–3):279–294, 1999.

3 Bonnie Berger and Tom Leighton. Protein folding in the hydrophobic-hydrophilic (HP) model is NP-complete. *Journal of Computational Biology*, 5(1):27–40, 1998.

4 Sarah Cannon, Erik D. Demaine, Martin L. Demaine, Sarah Eisenstat, Matthew J. Patitz, Robert Schweller, Scott M. Summers, and Andrew Winslow. Two hands are better than one (up to constant factors). In *STACS: Proceedings of the Thirtieth International Symposium on Theoretical Aspects of Computer Science*, pages 172–184, 2013. Arxiv preprint: `1201.1650`.

5 Ho-Lin Chen and David Doty. Parallelism and time in hierarchical self-assembly. In *SODA 2012: Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1163–1182. SIAM, 2012.

6 Matthew Cook, Yunhui Fu, and Robert T. Schweller. Temperature 1 self-assembly: Deterministic assembly in 3D and probabilistic assembly in 2D. In *SODA 2011: Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM, 2011. Arxiv preprint: `arXiv:0912.0027`.

7 Pierluigi Crescenzi, Deborah Goldman, Christos Papadimitriou, Antonio Piccolboni, and Mihalis Yannakakis. On the complexity of protein folding. *Journal of computational biology*, 5(3):423–465, 1998.

8 R. Das and D. Baker. Automated de novo prediction of native-like RNA tertiary structures. *PNAS*, 104:14664–14669, 2007.

9 K.A. Dill. Theory for the folding and stability of globular proteins. *Biochemistry*, 24(6):1501–1509, 1985.

10 David Doty, Jack H. Lutz, Matthew J. Patitz, Robert T. Schweller, Scott M. Summers, and Damien Woods. The tile assembly model is intrinsically universal. In *FOCS 2012*, pages 439–446, October 2012.

11 Constantine Glen Evans. *Crystals that count! Physical principles and experimental investigations of DNA tile self-assembly*. PhD thesis, California Institute of Technology, 2014.

12 Kirsten L. Frieda and Steven M. Block. Direct observation of cotranscriptional folding in an adenine riboswitch. *Science*, 338(6105):397–400, 2012.

13 Kenichi Fujibayashi, David Yu Zhang, Erik Winfree, and Satoshi Murata. Error suppression mechanisms for dna tile self-assembly and their simulation. *Natural Computing: an international journal*, 8(3):589–612, 2009. doi:10.1007/s11047-008-9093-9.

14 Cody Geary, Paul W. K. Rothemund, and Ebbe S. Andersen. A single-stranded architecture for cotranscriptional folding of RNA nanostructures. *Science*, 345:799–804, 2014.

15 Boyle J, Robillard G, and Kim S. Sequential folding of transfer RNA. a nuclear magnetic resonance study of successively longer tRNA fragments with a common 5' end. *J Mol Biol*, 139:601–625, 1980.

16 Hosna Jabbari and Anne Condon. A fast and robust iterative algorithm for prediction of rna pseudoknotted secondary structures. *BMC bioinformatics*, 15(1):147, 2014.

17 D. H. Mathews, M. D. Disney, J. L. Childs, S. J. Schroeder, M. Zuker, and D. H. Turner. Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure. *PNAS*, 101:7287–7292, May 2004. doi:10.1073/pnas.0401799101.

18 D.H. Mathews. Revolutions in rna secondary structure prediction. *Journal of molecular biology*, 359(3):526–32, 2006. doi:10.1016/j.jmb.2006.01.067.

19 Pierre-Étienne Meunier, Matthew J. Patitz, Scott M. Summers, Guillaume Theyssier, Andrew Winslow, and Damien Woods. Intrinsic universality in tile self-assembly requires cooperation. *SODA 2014: Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, 2014.

20 M. Paterson and T. Przytycka. On the complexity of string folding. In F. Meyer and B. Monien, editors, *ICALP 1996*, volume 1099 of *LNCS*, pages 658–669. Springer Berlin Heidelberg, 1996.

21 M. Popenda, M. Szachniuk, M. Antczak, K.J. Purzycka, P. Lukasiak, N. Bartol, J. Blazewicz, and R.W. Adamiak. Automated 3D structure composition for large RNAs. *Nucleic Acids Research*, 40(14):e112, 2012. doi:doi:10.1093/nar/gks339.

22 Elena Rivas. The four ingredients of single-sequence rna secondary structure prediction. a unifying perspective. *RNA Biol*, 10(7):1185–1196, Jul 2013. 23695796[pmid]. doi:10.4161/rna.24971.

23 Paul W. K. Rothemund. Folding DNA to create nanoscale shapes and patterns. *Nature*, 440(7082):297–302, March 2006. doi:10.1038/nature04586.

24 Nadrian C. Seeman. Nucleic-acid junctions and lattices. *Journal of Theoretical Biology*, 99:237–247, 1982.

25 R. Unger and J. Moult. Finding the lowest free energy conformation of a protein is an NP-hard problem: proof and implications. *Bulletin of Mathematical Biology*, 55(6):1183–1198, 1993.

26 Erik Winfree. *Algorithmic Self-Assembly of DNA*. PhD thesis, Caltech, June 1998.

27 Damien Woods, Ho-Lin Chen, Scott Goodfriend, Nadine Dabby, Erik Winfree, and Peng Yin. Active self-assembly of algorithmic shapes and patterns in polylogarithmic time. In *Proceedings of ITCS 2013: Innovations in Theoretical Computer Science*, pages 353–354, 2013.

28 Bernard Yurke, Andrew J Turberfield, Allen P Mills, Friedrich C Simmel, and Jennifer L Neumann. A DNA-fuelled molecular machine made of DNA. *Nature*, 406(6796):605–608, 2000.

29 Michael Zuker and David Sankoff. Rna secondary structures and their prediction. *Bulletin of Mathematical Biology*, 46(4):591–621, 1984. doi:10.1007/BF02459506.