Shape Reachability in Oritatami Is Complete for P

Sota Kano
The University of Electro-Communications, Tokyo, Japan

Shinnosuke Seki[1]
The University of Electro-Communications, Tokyo, Japan

Abstract

RNA co-transcriptional folding is a process in which an RNA sequence, or *transcript*, folds upon itself while being synthesized nucleotide by nucleotide according to its DNA template. Geary, Rothemund, and Andersen have demonstrated how to program a rectangular tile-like shape into (the DNA template of) an RNA transcript that folds co-transcriptionally into the shape *in vitro*. Using the oritatami model of co-transcriptional folding, we launch the study on the verification of co-transcriptionally folding systems. Shapes are the primary verification target of practical signification; it is abstracted as a set S of points on the 2D triangular grid. Thus, the shape reachability problem asks if the transcript of a given oritatami system goes through all and only the points in S and halts. We demonstrate a logspace reduction from the circuit value problem (`CVP`), a well-known **P**-complete problem, into a subproblem of the shape reachability whose input oritatami system is promised to be deterministic and at delay 3 (abstraction of the relative speed of local optimization to that of transcription), thus concluding that this subproblem `DSR`(3) is also **P**-complete. What to be verified may specify not only which points to be visited, but also which route should be taken (path reachability), and, moreover, how abstract nucleotides (beads) along the transcript should bind with each other (conformation reachability). We show that as long as the delay is bounded from above by a constant, the conformation reachability can be solved in logspace, and to this problem, path reachability can be reduced in logspace under the promise that a given oritatami system lets beads form as many bonds as possible (maximum arity).

2012 ACM Subject Classification Theory of computation → Complexity classes; Theory of computation → Problems, reductions and completeness; Applied computing → Bioinformatics

Keywords and phrases RNA co-transcriptional folding, Oritatami model, Reachability problems, P-completeness

Digital Object Identifier 10.4230/LIPIcs.DNA.32.3

Supplementary Material *Dataset*: `https://github.com/kabegamikamio/Gates2Oritatami` archived at `swh:1:dir:4753e8fcffbae60d9d175e40cd82be6ec5b997f3`

Funding *Shinnosuke Seki*: His research is supported in part by Bourse Région Ambition Internationale de Auvergne-Rhône-Alpes, France, No. 00284230-1 (Rokam) and KAKENHI Grant-in-Aid for Scientific Research (C) No. 20K11672.

Acknowledgements We thank Florent Becker for fruitful discussions at the launch of the reported research, and Coda Bourotte for providing insightful comments on the limitations of the Common-rail architecture. Let us take opportunity to express our sincere gratitude towards anonymous referees of **DNA32** for their valuable and encouraging comments to improve the scientific rigor and readability of this manuscript.

[1] Corresponding author

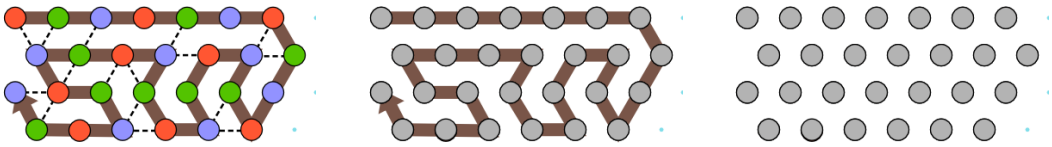


Figure 1 The three types of target considered in the reachability problems: `CfR` (conformation reachability), `PR` (path reachability), and `SR` (shape reachability).

1 Introduction

It is computationally hard to predict thermodynamical folding of RNA [1, 5, 6], wherein an RNA sequence is expected to reach a structure that minimizes free energy, but MFE structures cannot be computed easily unless certain types of non context-free structures, so-called *pseudoknots*, are left out of the picture [23, 27]. RNA sequences also fold isothermally, in particular while being transcribed, that is, synthesized sequentially nucleotide by nucleotide from their DNA templates; the RNA sequences thus synthesized are called *transcripts*. Both this process of *co-transcriptional folding* (CF) and thermodynamical annealing have been engineered as a programming platform for self-assembly of single-stranded RNAs. Annealing is more reliable as of date but seems hardly applicable *in vivo*, where natural RNAs are transcribed and fold to perform various computations, and considerable efforts have been devoted in the last decade to the CF-driven self-assembly of structures and computations [7, 8, 9, 13, 14, 16, 20, 22, 24, 26]. A transcript may not reach its MFE structure co-transcriptionally by being blocked in midway kinetically or topologically. Geary, Rothemund, and Andersen took the initiative in the molecular programming of self-assembly of CF-driven ssRNAs in 2014 [13, 16] by demonstrating how to program a rectangular tile-like shape in a transcript and a sequence of events at the oligonucleic domain-level through which the transcript folds co-transcriptionally into the shape. This methodology has been extended further to 3D wireframe polyhedra *in vitro* [7, 20] and even to arbitrary graphs *in silico/papyro* [8, 24]. Events considered explicitly therein are the hybridization of domains into a helix and the coaxial junction of two helices via a kissing loop (KL) pseudoknot. A target shape is abstracted as a graph G, one of the spanning trees of G is chosen as T, and a transcript is designed such that it traces the contour of T and forms each edge of T as a helix and each *co*-edge (of G but not on T) as a KL. In [20], 5-petal flowers and tetra squares, both of which are 2-dimensional, as well as tetrahedra, have thus been designed and assembled *in vitro* by folding co-transcriptionally and then being annealed afterward. This means that these structures are thermodynamically favored, and they are reachable even solely by co-transcriptional folding according to their AFM images taken prior to post-process annealing. As they are designed so as to fold hierarchically, that is, helices first and then KLs, a certain proportion of successful instances may suggest that global reconfigurations are hardly involved there.

Suppose that the technology of programming a global-reconfiguration-free CF pathway were established. As such a pathway is a coarse-grained specification of events only at the level of oligonucleic domains, a transcript behaves differently depending on what RNA sequences are assigned to its domains. Once fully specified thus at the nucleotide level, the transcript can be verified for an aimed shape efficiently, in linear time in principle. In case of failure, can programmers install a local patch or do they need to overhaul the sequence design? The answer to this question should depend on how much they know about the target shape. If the shape is not connected, a single transcript is certainly not enough; even if it is,

it is computationally hard to test if the shape can be traced, by a Hamiltonian path (see [12]). What if the shape is promised to be traceable, or the programmers are provided with an actual Hamiltonian path? The goal of this paper is to settle these questions in the framework of a reachability test using a formal *oritatami* model of co-transcriptional folding [15]. In this model, a sequence of abstract molecules, or *beads*, folds co-transcriptionally on the triangular grid. This is a discrete-time model; the sequence is transcribed one bead per step, and at every step, only the most nascent δ beads at the end of the thus elongating "transcript" are allowed to reconfigure for more bonds (between adjacent beads), where δ is a system parameter called *delay*. A structure is formalized as a triple of a bead sequence, a self-avoiding path, and a set of bonds between adjacent beads that "like each other," and called a *conformation*, see Figure 1 (left). Once an oritatami system M is designed, it takes only as many time steps as the length of the transcript of M to see how it folds, or to turn down M because it cannot stabilize a bead uniquely, or *deterministically*. Once the determinism is confirmed, it should be verified if it "reaches" its aimed target (actually, the check of determinism and reachability test can be done in parallel). The system M can be verified for a conformation in logarithmic space, i.e., parallelizable, as long as δ is bounded by a constant (Proposition 6). However, engineers may be informed of the target only partially as noted above, motivating path and shape reachability. Having been promised to be deterministic, M can be tested even for these targets in time exponential in δ but linear in the length of its transcript.

The main contribution of this paper is to prove that the shape reachability problem is **P**-complete; this means that the test is highly unlikely to be parallelizable, even if an input oritatami system is promised to be deterministic and at delay 3 (Theorem 16). The proof consists of demonstrating how to encode in logarithmic space an instance of `CVP` (circuit value problem) [19], which is the most classical **P**-complete problem, primarily as the transcript of an oritatami system to be verified, but also as part of a target shape. This result does not contradict the Turing universality of the oritatami model at delay 3 [25]. Indeed, the point reachability problem is undecidable, but it limits neither time nor space until the transcript reaches a target point, while all the reachability problems studied in this paper delineate the space for computation by an input shape.

2 Preliminaries

Let Σ be a finite set of types of abstract molecules, or *beads*, and let Σ^* be the set of finite sequences of bead types. For a sequence $w = a_1a_2\cdots a_n \in \Sigma^*$ of n beads $a_1, \cdots, a_n \in \Sigma$, its length is denoted by $|w|$, i.e., $|w| = n$. We refer to its i-th bead a_i as $w[i]$, and to its subsequence $a_ia_{i+1}\cdots a_j$ as $w[i..j]$; if the subsequence is a prefix of w, that is, $i = 1$, we also use $w[..j]$. This way of referencing is also applied to a self-avoiding path P on the 2-dimensional triangular grid $\mathbb{T} = (\mathbb{Z}^2, \sim)$, where $p \sim q$ means that the grid points p, q are next to each other, that is, $p - q$ is in $\hat{V} = \{(1,0), (0,1), (-1,1), (-1,0), (0,-1), (1,-1)\}$, which is the set of the six unit vectors of $\mathbb{T}$.

Oritatami [15] is a discrete-time model of co-transcriptional folding on the triangular grid. In the model, a sequence of bead types is transcribed, that is, elongates one bead per step, and folds on the triangular grid in such a way that at every step, only the most nascent δ beads are allowed to reconfigure so as to form as many bonds as possible between adjacent beads without breaking the transcript's (covalent) backbone or having the transcript intersect with itself, where δ is a parameter of an oritatami system, and called *delay*. Another system parameter called *(binding) rule* $\heartsuit$ specifies which types of beads are allowed to bind with

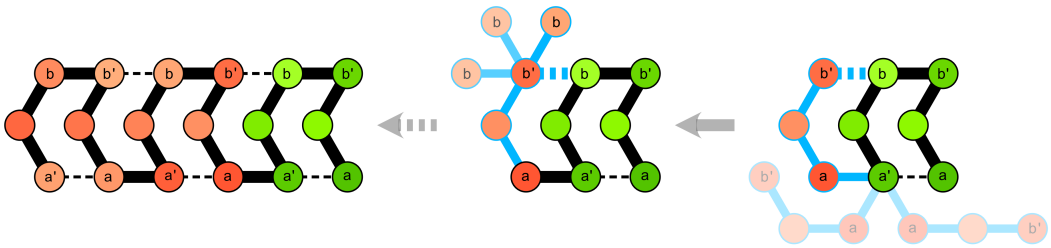


Figure 2 A glider and its folding at delay 3. Its transcript is of period 6 as a $\mathtt{a} \bullet \mathtt{b}'\mathtt{b} \bullet \mathtt{a}'\mathtt{a} \bullet \cdots$ and its rule is $\mathtt{a}\heartsuit\mathtt{a}', \mathtt{b}\heartsuit\mathtt{b}'$.

each other as long as they are at unit distance. That is, $\heartsuit$ is a symmetric relation on Σ. A *(valid) conformation (with respect to $\heartsuit$)* is a triple $c = (w, P, H)$ of a self-avoiding path P on $\mathbb{T}$, a sequence w of as many beads as the points along P, and a set H of pairs of bead indices representing a bond, that is:

$$H \subseteq \left\{\{i,j\} \;\middle|\; P[i] \sim P[j], 1 \le i,\ i+2 \le j < |w|,\ w[i]\heartsuit w[j]\right\}.$$

Here, the inequality $i + 2 \le j$ implies that a bond cannot form between two consecutive beads along the transcript, which are bonded covalently rather. By the *length* of c, we mean $|w|(= |P|)$, and denote it by $|c|$. The set of all valid conformations w.r.t. $\heartsuit$ is denoted by $\mathcal{C}_\heartsuit$. In the following, we always clarify which rule $\heartsuit$ is under consideration and do not bother specifying with respect to which rule a conformation under consideration is valid. The stability of c is measured as $E(c) = -|H|$; that is, the more bonds a conformation has, the more stable it becomes. Its *arity* is the maximum number of bonds per bead, and formally defined as

$$\max_{1 \le k \le |w|} \left|\left\{\{i,j\} \in H \;\middle|\; i = k \text{ or } j = k\right\}\right|.$$

A conformation is said to be of *maximum arity* if its arity is equal to the maximum degree of the underlying grid. The set of all conformations of arity at most α is denoted by $\mathcal{C}_{\heartsuit,\le\alpha}$, or simply by $\mathcal{C}_{\le\alpha}$.

Let us introduce two operations between conformations. One is to take a *prefix* of a conformation $c = (w, P, H)$; this operation keeps only a prefix w' of w in the classical sense, leaves the same number of points at the beginning of P, and extracts from H only the bonds between beads on the prefix. The other is elongation. Elongating c by a bead $b \in \Sigma$ yields the following set of conformations:

$$c^{\triangleright b} = \left\{(wb, Pv, H \cup H') \in \mathcal{C}_\heartsuit \;\middle|\; H' \subseteq \left\{\{i, |w|{+}1\} \;\middle|\; P[i] \sim v, i \le |w| - 1,\ w[i]\heartsuit b\right\}\right\}.$$

For elongating a conformation c rather by a sequence of bead types $b_1 b_2 \cdots b_n$, it suffices to first elongate c by the first bead b_1, then elongate each of the resulting conformations, that is, those in $c^{\triangleright b_1}$, by b_2, and so on.

An *oritatami system* M is a 4-tuple $M = (t, \heartsuit, \delta, \alpha)$ of a transcript $t \in \Sigma^*$, a rule $\heartsuit$, and two integer parameters δ (delay) and α (arity). An input is given as a *seed* conformation σ of arity at most α. Its computation begins with such a seed, $c_0 = \sigma$, and consists of conformations $c_1, c_2, \cdots$ that satisfy:

$$c_i \in \operatorname*{argmin}_{c \in c_{i-1}^{\triangleright t[i]} \cap \mathcal{C}_{\le\alpha}} \left(\min_{c' \in c^{\triangleright t[i+1..i+\delta-1]} \cap \mathcal{C}_{\le\alpha}} E(c') \right). \tag{1}$$

The i-th bead $t[i]$ is thus stabilized, according to c_i that satisfies (1) (see in Figure 2 how a bead is thus stabilized). With more than one candidate for c_i, M is *nondeterministic on the seed* σ. On the other hand, if for all time steps $i \leq |t|$, c_i is uniquely determined or there is no candidate, M is *deterministic on* σ. Particularly, the lack of candidate aborts M; this happens when all the neighbors of $t[i-1]$ have been already occupied. Otherwise, the system fully transcribes and stabilizes t as $c_{|t|}$, and this *terminal* conformation serves as its outcome on the input σ. If an oritatami system gets nondeterministic on the seed σ, it may have multiple terminal conformations; let us denote their set by $T(M, \sigma)$; this notation will be abused to refer to the unique terminal conformation if it is clear from the context that M is deterministic and does not abort on σ.

The bead stabilization (1) is local in the sense that it is affected by only at most 6δ beads inside the circle with radius $\delta + 1$ centered around $t[i-1]$, which was already stabilized at the previous step. Let us informally call this region the (i-th) *universe*. The circumference of the circle is too far for nascent beads to be placed, but a bead there can affect the bead stabilization by attracting the nascent segment $t[i..i+\delta-1]$ unless it has already formed α bonds.

When an oritatami system $M = (t, \heartsuit, \delta, \alpha)$ is given as input, the amount of space to store it matters. If all beads along t are pairwise distinct, then the rule $\heartsuit$ can contain $|\Sigma|^2$ elements; however, this is unrealistic and it is more realistic to treat the size of Σ as a constant. Since δ larger than $|t|$ makes no mathematical sense and α is bounded by the maximum degree of the underlying grid (6 for the triangular grid), when Σ is fixed, M can be stored within $O(|t|)$ bits.

2.1 Parallel feasibility: P-complete vs NC

Class **P** is the set of decision problems that can be solved with a deterministic Turing machine in polynomial time. It is regarded as a class of practically feasible problems.

Nick's class, in short **NC**, is a subset of **P** and consists of the problems which can be solved with a deterministic parallel Turing machine with a polynomial number of processors in polylogarithmic time. Problems in **NC** are regarded as *efficiently parallelised*. L is the class of problems that are computable in logarithmic space. These problems can also be efficiently parallelised. For these complexity classes and Log-space reduction, see [2].

▶ **Lemma 1** ([3, 17]). $\mathbf{L} \subseteq \mathbf{NC}$.

A problem is **P**-complete if it is in **P** and any problem in **P** can be reduced to it with an appropriate reduction such that the reduction itself is efficiently parallelised. **P**-complete problems are believed to be *inherently sequential*; thus, there may not be efficient parallel algorithms to solve them.

2.2 Circuit value problem

The circuit value problem, `CVP`, is the problem to decide the terminal output of a given Boolean circuit consisting of gates for inputs, AND, OR and NOT operations. A `CVP` instance is shown in Figure 3. It is formally defined as follows.

▶ **Definition 2** (CVP).

- Input: a Boolean circuit with $n+1$ gates $G = (g_0, g_1, ..., g_n)$, where each g_i $(0 \leq i \leq n)$ is either an input or a gate of AND, OR or NOT.
- Output: yes if the terminal output g_n is true.

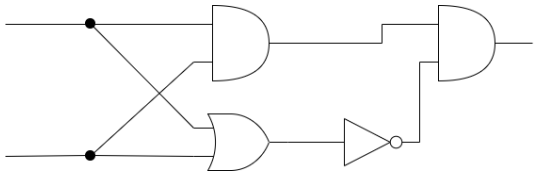


Figure 3 An example of CVP.

▶ **Lemma 3** ([19]). `CVP` *is* **P***-complete.*

Just as the `SAT` is often used to prove **NP**-completeness, `CVP` is the standard source for proving **P**-completeness.

2.3 Reachability problems in Oritatami

Reachability problems in the oritatami model in general ask: given an oritatami system $M = (t, \heartsuit, \delta, \alpha)$, a seed σ, and a "target" as input, are terminal conformations in $T(M, \sigma)$ equal to the target in a problem-dependent sense? As a target, we shall consider conformations, paths, and shapes, which are a set of grid points. Let us first settle the issue of input encoding. Assume that the alphabet Σ is fixed, and recall first that M can be encoded in linear space in $|t|$. A target conformation $c = (w, P, H)$, with $|w| = n$, can be encoded in $O(n)$ bits. Indeed, P consists of as many points as beads in w, its first bead can be placed at $(0, 0)$ without loss of generality, and the rest of P can be rather described as a sequence of $n - 1$ vectors $v_1, v_2, \cdots, v_{n-1}$ taken from $\hat{V}$ such that $P[i] = (0, 0) + v_1 + \cdots + v_{i-1}$. The set H can contain at most $6n$ bonds, and each of them can be encoded in $\lceil \log 6 \rceil = 3$ bits as an index for the six possible unit vectors on the triangular grid. Lastly, as the seed σ must be a prefix of c (otherwise, how can M reach c, starting from σ?), it can be encoded as compactly as c. In order for $T(M, \sigma) = C$ to hold, the length of the seed plus $|t|$ must be equal to n. Therefore, the whole input can be encoded in $O(n)$ space. All alternative targets considered in this paper can be obtained by somehow simplifying conformations so that in all problems formalized from now on, the size of an input is $O(n)$.

Logarithmic space in n is enough to retrieve from the above encoding of the conformation c the labels of all the beads next to the i-th point of P. The index $i \leq n$ can be stored in $\lceil \log n \rceil$ bits. The processor computes all the indices $1 \leq m < n$ such that $\sum_{j=i}^{m} v_j \in \hat{V}$; then the bead $w[j]$ is placed at a point next to $P[i]$; note that there exist at most 6 such j's. In this way, the processor can even retrieve in $O(\log n)$ space all the pieces of information about the i-th universe necessary and sufficient to decide how the next bead $t[i]$ should be stabilized.

If a conformation is given as a target (`CfR` in Definition 4), it uniquely determines all intermediate conformations $c_1, c_2, \cdots$, and requires M to be deterministic on the seed c_0. For the other reachability problems (`PR` and `SR` in Definitions 7 and 13), it makes at least mathematical sense to formalize their deterministic variants that promise that a given oritatami system folds deterministically on a given seed. It can be tested in $O(6^\delta \cdot |t|)$ time whether M is deterministic or not on a given seed; for $1 \leq i \leq |t|$, the search for c_i considers at most 6^δ elongations of c_{i-1} and if the size of the right-hand side of (1) is 2 or greater, then M is not deterministic.

2.3.1 Conformation reachability (CfR)

Let us begin with the following reachability problem whose target is a conformation.

▶ **Definition 4** (CfR).
- Input: an oritatami system M, a seed σ, and a conformation c.
- Output: yes if $T(M, \sigma) = c$.

The $O(6^\delta \cdot |t|)$-time determinism test mentioned above actually simulates the folding by M fully, and the only thing required to solve `CfR` is to verify that the simulated terminal conformation is equal to c.

▶ **Proposition 5.** `CfR` *can be solved in sequential time of* $O(6^\delta \cdot n)$*; hence, it is fixed-parameter-linear.*

`CfR` can be solved efficiently in parallel within the class of delay-δ oritatami systems for a fixed δ. Let `CfR`(δ) be the set of `CfR` instances whose input oritatami system is of delay at most δ; we will denote likewise the subproblem of other reachability problems by bounding the delay of input oritatami systems by a constant δ. In light of Lemma 1, it suffices to demonstrate how a processor simulates the stabilization of $t[i]$ in the logarithmic space in n. As explained earlier, the i-th universe can be retrieved from the encoding of the target conformation c using $O(\log n)$ space, and can be stored in space $O(6\delta \cdot \log n)$. This universe enables the processor to decide how $t[i]$ should be stabilized. The amount of bits in use remain logarithmic in n as long as δ is bounded by a constant.

▶ **Proposition 6.** *For all integer* δ, `CfR`(δ) *is in* **L**.

2.3.2 Path reachability (PR)

▶ **Definition 7** (PR).
- Input: An oritatami system M, a seed σ, and a path P.
- Output: Yes if the path of *all* terminal conformations is equal to P.

Unlike the conformation, an oritatami system does not have to follow a path deterministically. If an oritatami system given as input is promised to be deterministic, simulating one run of M is enough; during the run, it suffices to check if $t[i]$ is stabilized at the point $P[i]$ for all time steps $i \leq n$. Let us formalize the deterministic variant of `PR` as follows:

▶ **Definition 8** (DPR).
- Input: An oritatami system M and a seed σ such that $|T(M, \sigma)| = 1$, along with a path P.
- Output: Yes if the path of the *unique* terminal conformation is equal to P.

▶ **Proposition 9.** `DPR` *can be solved in sequential time of* $O(6^\delta \cdot n)$.

As shown in the conformation reachability, within the class of delay-δ oritatami systems for a fixed δ, this problem can be solved using logarithmic space; let `DPR`(δ) be the set of `DPR` instances whose input oritatami system is of delay at most δ.

▶ **Proposition 10.** *For all* $\delta \geq 1$, `DPR`(δ) *is in* **L**.

There may exist a nondeterministic oritatami system which always follows the same path, but with different binding patterns. The problem `PR` is thus motivated at least theoretically. Leaving its computational complexity as an open problem, here we show that an instance of `PR` with a maximum-arity oritatami system can be reduced to an instance of `CfR`, and hence, can be solved efficiently in parallel. Let us denote the set of such instances of `PR` by $\texttt{PR}_{\max \alpha}$. It makes no sense for a maximum-arity oritatami system to prevent ♡-related beads from

forming a bond. This observation implies that if the system reaches the path P, its terminal conformation is the one obtained from P by binding all the beads that are ♡-related and adjacent in P.

▶ **Proposition 11.** *For all* $\delta \geq 1$, $\mathtt{PR}_{\max \alpha}(\delta) \in \mathbf{L}$.

Proof. It suffices to reduce an instance of $\mathtt{PR}_{\max \alpha}(\delta)$ to an instance of $\mathtt{CfR}(\delta)$. Let P be the target path and t be the transcript of the input oritatami system with $|t| = n$. For $i \leq n$, the i-th point of P should be labeled with $t[i]$, and this labeling can be done by storing the index i for reference using $\lfloor \log n \rfloor$ bits. In this way, we can determine which type of bead is supposed to stabilize at each point along P. Moreover, it can be determined uniquely how they are supposed to bind with each other because the input oritatami system is promised to be of maximum arity. Thus, an instance of $\mathtt{CfR}(\delta)$ has been obtained. ◀

What if the arity of a given oritatami system is not maximum? In this realistic scenario, once a bead exhausts its capability of forming α bonds, it becomes an obstacle that does not contribute to lowering the free energy, at least by binding, but stands in the way of the transcript. Therefore, it seems necessary to track and memorize how the prefix $t[..i-1]$ has folded co-transcriptionally so far in order to decide if each bead in the i-th universe is still active or already inert.

This problem is at least in **coNP**. Indeed, if the oritatami system of a given instance can follow another path to the target one, that path serves as a polynomial-size no certificate. It suffices to run the system and ensure that it can follow that path.

▶ **Proposition 12.** $\mathtt{PR}(\delta)$ *is in* **coNP** *for all* $\delta \geq 1$.

2.3.3 Shape reachability (SR)

A *shape* is a set of points on $\mathbb{T}$. Given a conformation $c = (w, P, H)$, its shape is a set of points along the path P.

▶ **Definition 13** (SR).
- Input: an oritatami system M, a seed σ, and a shape S.
- Output: yes if the shape of *all* the terminal conformations in $T(M, \sigma)$ is equal to S modulo translation.

Note that in this definition, the output verification allows for translation. This is because unlike the conformation or path reachability, the input shape S does not specify where the seed should be. As done for path reachability, here we define the deterministic variant of shape reachability (DSR) by requiring an input oritatami system M and a seed σ to satisfy $|T(M, \sigma)| = 1$.

As done for Proposition 12, we can show that without promise of determinism, the shape reachability problem is in **coNP** under any constant bound on the delay.

▶ **Proposition 14.** *For all* $\delta \geq 1$, $\mathtt{SR}(\delta)$ *is in* **coNP**.

In contrast, if a given oritatami system is promised to fold deterministically on a given seed, then the shape reachability can be solved in a quadratic time in n. Based on Proposition 9 on DPR, it should suffice to note that there are $O(n)$ possible arrangements of the seed in order to prove the following result.

▶ **Proposition 15.** DSR *can be solved in sequential time of* $O(6^\delta \cdot n^2)$. *Thus, for all* $\delta \geq 1$, $\mathtt{DSR}(\delta)$ *is in* **P**.

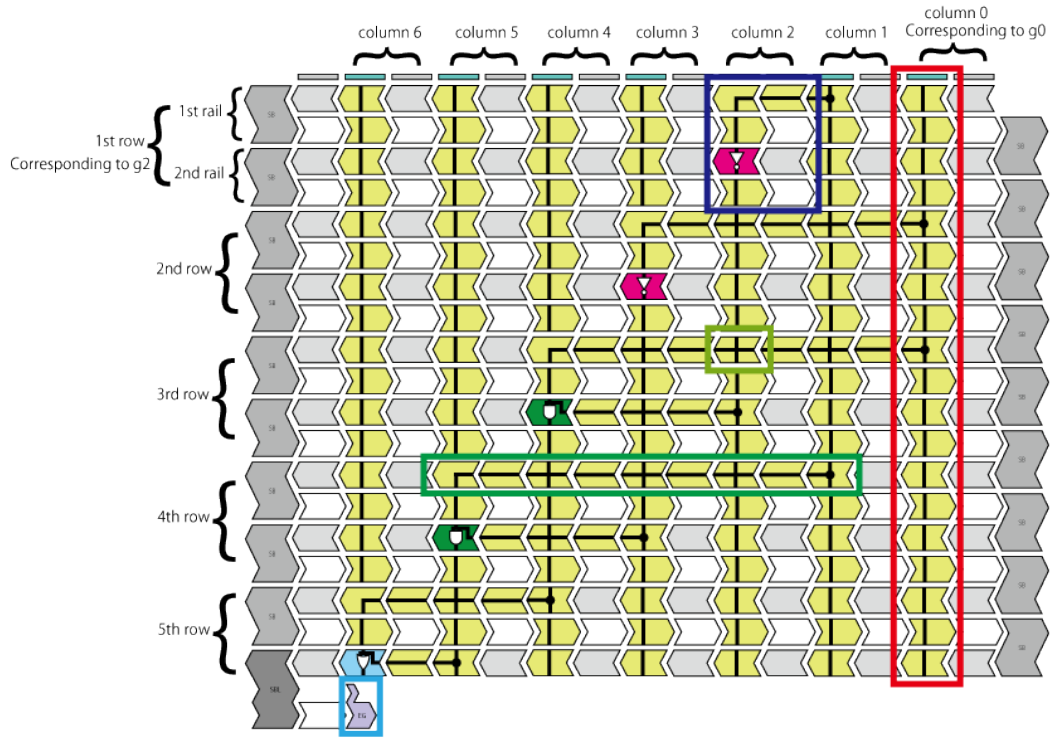


Figure 4 (The Common-rail architecture) This example illustrates an oritatami system that computes the XOR $g_0 \oplus g_1$ using AND, OR and NOT gates where g_0, g_1 are both inputs. The pink, green and light blue chevron shapes are NOT, AND and OR gates respectively. They are all arranged along the diagonal of the system. The inputs and the output of each gate are conveyed from top to bottom by long vertical wires called common rails. The red box is the common rail for g_0. From these common rails, values are branched as needed onto horizontal lateral rails to be delivered to each gate. The green box is the lateral rail that conveys the value of g_1 to g_5. At the intersections of common rails and lateral rails, XO modules (detailed later) indicated by the lime green box are placed to resolve the crossovers. The light blue box at the bottom-left is the terminal module to visualise the terminal output of the system. The blue box indicates the gate cell of g_2 which is located at the first row and the column 2.

Unlike the conformation or path, the shape provides neither the order of points along which M folds, nor where on $\mathbb{T}$ the seed is located. The first deficiency seems to make the shape reachability intrinsically sequential. Indeed, we shall prove, in the rest of this paper, that DSR(3), the class of DSR instances whose oritatami system is at delay 3, is **P**-complete.

▶ **Theorem 16.** DSR(3) *is* **P***-complete.*

3 A hard-wired CVP solver in oritatami

In order to prove Theorem 16, we shall demonstrate a general architecture of a delay-3 oritatami CVP solver, which can be wired according to a Boolean circuit to be simulated using only logarithmic space. The architecture (see Figure 4) is based on the zigzag oritatami counter [10, 21], in which the transcript folds macroscopically in zigs (←) and zags (→) of height 3, while microscopically each of its modules, such as half-adders, folds into a glider (Figure 2) or into one of the other conformations of the same chevron-like shape depending on

the current count and a carry; independent of the underlying computation, the counter always folds into the same shape (from the *global* bird's-eye view, one can read the computation by measuring the number of zigzags between two consecutive overflows, where zigzags get wider for 1 extra bit). The novel, programmer-friendly geometry-driven architectures in oritatami for computation [11, 25] do not serve the proof of Theorem 16. Since the hardness of SR appears to stem from the "stealthiness" of intermediate values of computation, simulations of those systems might be efficiently parallelisable.

The primary challenges in the proposed construction include:

- designing new modules to simulate Boolean circuits that conceal interim calculations, unlike "too chatty" existing systems;
- implementing 3D-like wiring crossovers within the 2D oritatami model; and
- developing a reduction algorithm that determines the optimal arrangement of modules and wiring within logarithmic space.

While our gate modules are largely original, they are inspired by the half-adder that intrinsically computes AND operation [10]. As detailed later, this module computes through its folding path rather than its shape. In addition, we developed a crossover module to facilitate multi-level interchanges of wiring. Finally, we propose the Common-rail architecture, which decouples input propagation from logical computation. This enables selective routing of values from the input-carrying rail to individual gates and ensures that the entire reduction is doable in logarithmic space, as the arrangement of modules is done by simple computations in this architecture.

3.1 Common-rail architecture

While the reduction algorithm we shall show in Section 4 must place gate and wiring modules at appropriate positions in the system, determining an optimal layout is challenging due to complex inter-dependencies between modules. To overcome this, we have devised a redundant but systematic architecture called the *common-rail architecture* (Figure 4), where all gate modules are placed diagonally on the basal plate of the system. In this architecture, starting from the seed positioned at the very top, the transcript macroscopically folds into zigzags and proceeds downward, being guided by switch-back modules, which were rather called left and right-turn modules in the zigzag counter [10, 21] from a zig to the next zag and from a zag to the next zig. The transcript is a chain of modules, a functional unit, all of which but the ones for switchback fold into a chevron-like shape of width 6 and height 3 (this is defined as the set of points occupied by the glider depicted in Figure 2). At its end, a terminal module reports the binary output of the circuit by shape.

This architecture has two types of wiring: *common rails* and *lateral rails*. Common rails run vertically from the seed to the bottom of the system, and each of them is dedicated to holding and conveying either an input encoded in the seed or the output of a specific gate. They are implemented by vertical wire modules piled on top of each other. In order to avoid crosstalk, gliders are intercalated in-between. Let us bundle a pillar of vertical wire modules and that of "insulators" together and call it a *column*. The i-th gate g_i, which is an input, AND, OR, or NOT, is placed inside the column i. Lateral rails are horizontal wires branched from common rails to deliver their values through a zig (of height 3) leftward to gates. All the modules along a zig are based on the glider, which is capable of propagating 1-bit of information as the position of its first bead: top for false and bottom for true, and are designed so as to always start and end at the same height, inheriting this capability of propagating 1-bit. Therefore, one logical gate is collaboratively accommodated in two

continuous zigzags to wire common rails into at most two input pins; let us call these zigzags a *row*. As illustrated in Figure 4, the i-th row, which consists of the $(2i-1)$-th and $2i$-th zigzags, takes responsibility for the computation by the i-th logical gate. The first input to the gate goes through the upper lateral rail, turns downward, and is fed to the gate from above, while the second input (only for AND and OR) goes through the lower lateral rail and is fed to the gate from the side. Unlike common rails, a lateral rail is placed only when necessary and deployed solely between its source common rail and the destination gate, with no further branching. Let us define a *gate cell* as a structural unit positioned at the intersection of each row and column.

This common-rail architecture enables the reduction algorithm to appropriately place modules with simple conditional branches, ensuring that it can run in logarithmic space.

3.2 Gadget construction for Boolean circuits

The basic design and specification of the modules were redesigned based on the half-adder for counting [10]. We have designed the following five categories of modules:

- gate modules (AND, OR, NOT)
- wire modules
- padding / absorber modules
- the switch-back module
- the terminal module

All but the terminal module fold into a 6-bead width and 3-bead height chevron shape.

Gate modules

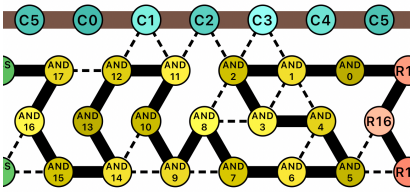
AND and OR gates (Figure 5 (a-d) and (e-h)) receive their first input from above, encoded as how the gate binds with the lateral rail above, while their second input is propagated from the previous module as the starting position of the folding process as: *top* representing false or *bottom* representing true. NOT gate (Figure 5 (i, j)) receives its sole input from above. Although the output specifications vary slightly between modules, they all share two common features: the output value is represented by the presence or absence of bonds between the gate and the common rail below, and the folding process terminates at the bottom position if the output is true, and otherwise at the top.

Wiring

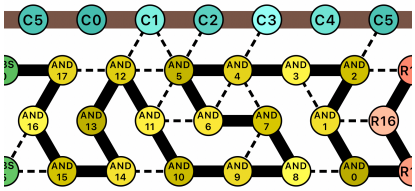
There are five types of wiring in this system:

- vertical wires for common rails
- horizontal wires for lateral rails
- crossovers
- 1-to-2 fan-out wires
- L-shaped wires

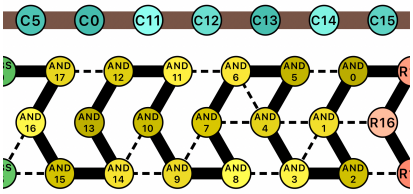
Vertical wires (Figure 6) constitute the common rails by being stacked on top of each other. They propagate 1-bit of information downward as absence or presence of bonds between them (no bond means true). Compare Figure 6 (a) with (b) and observe that when receiving false from above, it ends at the top (interpreted as false through lateral rails), or it ends at the bottom otherwise. This property enables this module to be also used to branch a lateral rail from a common rail. It suffices to transcribe a glider after this module; the 1-to-2 fan-out is implemented in this way. Non-bifurcation simple vertical wires can be



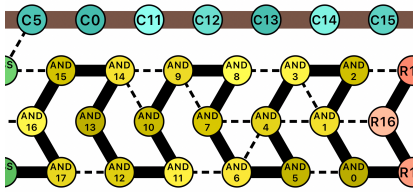
(a) AND00.



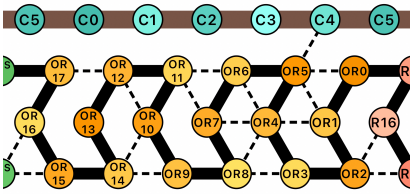
(b) AND01.



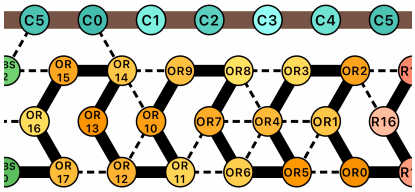
(c) AND10.



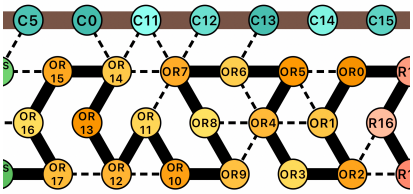
(d) AND11.



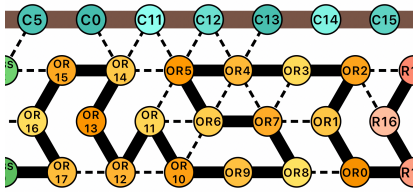
(e) OR00.



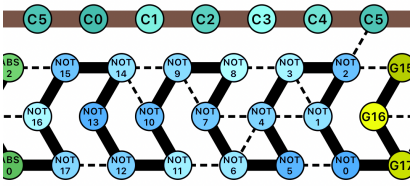
(f) OR01.



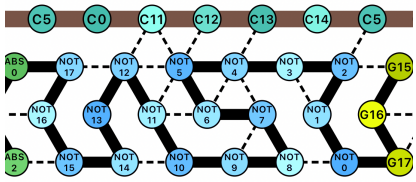
(g) OR10.



(h) OR11.

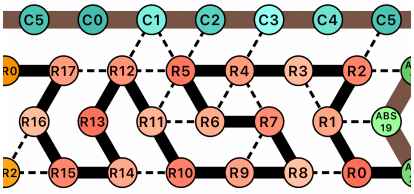


(i) NOT0.

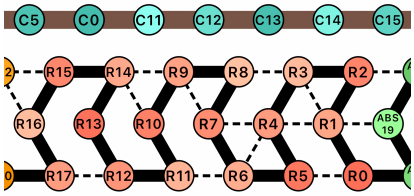


(j) NOT1.

Figure 5 All conformations of the AND, OR, and NOT gate modules. **(a-d)** All four conformations of the AND module on respective four inputs $(x, y) \in \{(0,0),(0,1),(1,0),(1,1)\}$, where x comes from above while y comes from right. ANDxy represents the conformation that computes $x \wedge y$. For instance, AND01 is the conformation of the computation $0 \wedge 1 (= 0)$. **(e-h)** All four conformations of the OR module on (x, y). **(i, j)** The two conformations of the NOT module on the respective inputs 0 and 1.



(a) R0.

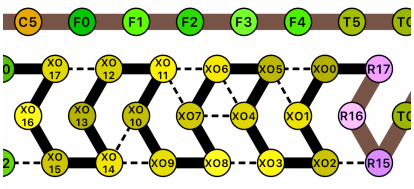


(b) R1.

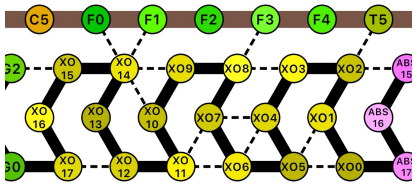
Figure 6 The two conformations of the vertical wire module on the respective two inputs 0 and 1. Note that the vertical wire module extends along the zig-zag path horizontally while it transmits a value vertically.

implemented merely by transcribing an auxiliary absorber module (explained shortly), which immediately cancels out a horizontal top/bottom signal by ending at the bottom no matter what. In fact, this module is the H0c and H1c components of the half-adder in [10]. Due to design constraints of intermodular binding rules, the system involves four "doppelgängers" (R, PA, PB and PC) of vertical wires, which are implemented identically but out of four pairwise-disjoint sets of bead types.

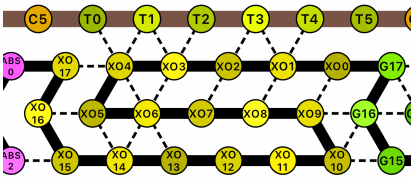
Horizontal wires are just a conventional glider (Figure 2), propagating 1-bit of information as the position of its first bead: top for false and bottom for true.



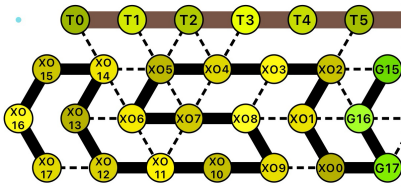
(a) XO00.



(b) XO01.



(c) XO10.

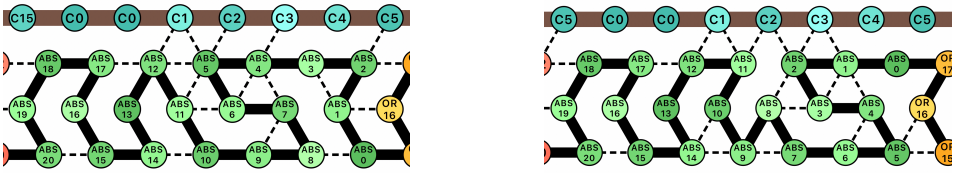


(d) XO11.

Figure 7 The four conformations of the crossover (XO) module on the respective four inputs $(x, y) \in \{(0,0),(0,1),(1,0),(1,1)\}$.

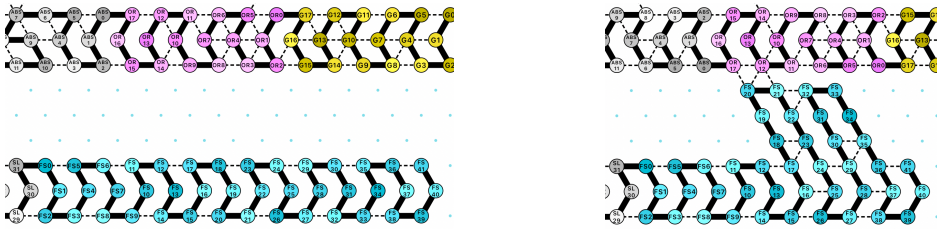
Crossover (XO) modules (Figure 7) are placed at the intersections of common and lateral rails to prevent signal collisions. Instead of physically crossing the wires in 3D space, this module achieves a "pseudo-crossover" by folding into one of the four possible conformations depending on the four possible inputs. Adjacent modules then interpret these motifs to retrieve the encoded values for each rail.

Finally, the L-shaped wiring module redirects a signal propagated horizontally downward at the end of a lateral rail in order to be fed to a gate below. While the structure of this module is identical to that of a horizontal wire, the value is interpreted and propagated to the gate by the vertical wire positioned directly beneath it.



(a) ABSb. **(b)** ABSt.

Figure 8 The two conformations of the absorber module, both of which end at the bottom, no matter whether they have started (a) at the bottom or (b) at the top.



(a) The terminal module indicates a false output. **(b)** The terminal module indicates a true output.

Figure 9 The terminal module and its output conformations. The module translates the final Boolean result into a geometric shape: (a) a horizontal extension represents false, while (b) an upward protrusion represents true. This distinct geometric change at the system's tail enables the encoding of the computation result as a Shape Reachability instance.

Auxiliary modules

The system involves modules that do not directly take part in logical computations but rather ensure the correct geometric growth, enable gates and wiring modules to function properly, and simplify their design. Gliders are used as a padding module to fill the vacant area, which can be found, for instance, to the left of the gates in each row. Absorbers (Figure 8) end at the bottom no matter whether it has started at the top or at the bottom. They are placed at the end of a lateral rail and cancel out the 1-bit of information having propagated therein. They also prevent interference between the computations of gates or wiring modules in neighbouring columns. The switch-back module is positioned at the end of each row to reverse the macroscopic folding direction (from zig to zag, and vice versa).

Terminal module

This is the only module that takes (two) different shapes. As shown in Figure 9, at the end of the system, it protrudes upward if and only if the terminal output of the Boolean circuit is true. As the computation has been fully concealed so far, without this module, the system would yield the unique shape no matter what the seed is, collapsing the reduction.

4 Logspace reduction from CVP to SR

Algorithm 1 receives a Boolean circuit $G = (g_0, g_1, \ldots, g_n)$ as input and generates a seed (C_0 in the pseudocode) in Phase 1 and a transcript (T) in Phase 2 both sequentially bit-by-bit. Storing only counters up to n, this algorithm can be executed in $O(\log n)$ space.

Algorithm 1 Logspace construction of seed and transcript from a gate sequence.

Input: A sequence of gates G of length n
Output: Seed C_0 and transcript T

```
1  C_0 ← ∅, T ← ∅, n_input ← 0;
   /* Phase 1:  Construct seed configuration                              */
2  for i ← 0 to n − 1 do
3      g ← G_i;
4      if g.type = INPUT then
5          n_input ← n_input + 1;
6          if g.value = true then
7              C_0.addFirst(C_t);
8          else
9              C_0.addFirst(C_f);
10     else
11         C_0.addFirst(C_none);
12 C_0.add(c_start);
   /* Phase 2:  Construct transcript                                      */
13 for y ← n_input to n − 1 do
14     g ← G_y, l ← left_index(g);
       /* First rail:  zig path                                           */
15     for x ← 0 to n − 1 do
16         case x = l do
17             T.add(R), T.add(G) ;          /* Branch from common to lateral */
18         case x = y do
19             T.add(G), T.add(ABS) ;                    /* Input to gate */
20         case x < l do
21             T.add(PA), T.add(ABS) ;                   /* Common rail */
22         case x > y do
23             T.add(G), T.add(ABS) ;                    /* Glider fill */
24         otherwise do
25             T.add(XO), T.add(G) ;                     /* Lateral rail */
26     T.add(SW);
       /* First rail:  zag path                                           */
27     for x ← 0 to n − 1 do
28         T.add(ABZ), T.add(PB);
29     T.add(SB);
       /* Second rail:  zig path                                          */
30     r ← right_index(g);
31     for x ← 0 to n − 1 do
32         if x = y then
33             T.add(g.type), T.add(ABS);
34         else if x = j then
35             T.add(PC), T.add(ABS);
36         else if g.type ≠ NOT ∨ x < r then
37             T.add(R), T.add(ABS);
38         else if x > y then
39             T.add(G), T.add(ABS);
40         else
41             T.add(XO), T.add(G);
42     if y = n − 1 then
43         T.add(SBL);
44         return (C_0, T);
45     else
46         T.add(SB);
       /* Second rail:  zag path                                          */
47     for x ← 0 to n − 1 do
48         T.add(ABZ), T.add(PC);
49     T.add(SB);
```

The circuit G is given as a comma-separated sequence of gate definitions enclosed in parentheses. While the gate indices are not explicitly labelled to these brackets, the $(i+1)$-th element in the sequence uniquely defines gate g_i. The encoding rules are as follows:

- If g_i is an input gate, it is represented as `(true)` or `(false)`.
- If $g_i = g_k \wedge g_j$, `(AND, k, j)`.
- If $g_i = g_k \vee g_j$, `(OR, k, j)`.
- If $g_i = \neg g_k$, `(NOT, k)`.

In Phase 2, a transcript is generated sequentially by catenating the module specified by the corresponding row and column indices according to the common-rail architecture (Figure 4). For the i-th row, the length of the lateral rails in its first and second zigs are determined by checking from which common rail(s) the gate g_i should be fed with its inputs, while the zags can be actually even hardcoded. The transcript is completed by appending a terminal module at the end.

An executable Java program of this algorithm is shared on `https://github.com/kabegamikamio/Gates2Oritatami`.

5 Concluding remarks

In this paper, we have formulated the conformation, path, and shape reachability problems in the oritatami model, and proved upper and lower bounds on their computational complexity under the assumption that the delay δ be bounded by a constant. In order to prove the **P**-completeness of the deterministic shape reachability problem, we developed a suite of logic gates (AND, OR, NOT) and supplementary modules such as wires and crossovers and demonstrated how to combine them into a delay-3 deterministic oritatami CVP solver in a logarithmic space. With a mechanism to guess Boolean values nondeterministically (see [18]), wiring the proposed logic gates in the proposed Common-rail architecture should enable us to design a delay-3 nondeterministic oritatami `UNSAT` solver, which would test a hardcoded Boolean formula according to a guessed assignment. If such a solver could be assembled from a given `UNSAT` instance in a logarithmic space, then `PR`(3) would be proved to be **coNP**-complete; recall that `PR`(3) $\in$ **coNP** (Proposition 12).

The length of a transcript is actually bounded by that of its DNA template, and there is a technical limit on the synthesizable DNA templates. Geary and Andersen have unveiled the idea of transcribing a periodic RNA transcript from a circular DNA template [13]. Most computational oritatami systems adopt their idea and utilize a periodic transcript [11, 15, 21, 25]. Once the length of the period of a transcript is bounded by a constant, a circuit cannot be hardcoded entirely in a transcript anymore, but needs to be also encoded on a seed and retrieved co-transcriptionally.

We have shown how to provide a polynomial-size no-certificate for the path and shape reachability problems (without the promise of determinism), proving that they are in **coNP**. Aside from the computational complexity, it should be interesting to imagine a path or a shape that is intrinsically nondeterministic, that is, cannot be reached by any deterministic oritatami system, but there exists a nondeterministic one all of whose terminal conformations follow the path/are of the shape (see [4] for an analog of this research direction in the abstract tile assembly model).

All these research directions should also be pursued in a more realistic model of RNA co-transcriptional folding.

References

1 Tatsuya Akutsu. Dynamic programming algorithms for RNA secondary structure prediction with pseudoknots. *Discrete Applied Mathematics*, 104(1-3):45–62, 2000. doi:10.1016/S0166-218X(00)00186-4.

2 Sanjeev Arora and Boaz Barak. *Computational Complexity. A Modern Approach*. Cambridge University Press, 2009.

3 Allan Borodin. On relating time and space to size and depth. *SIAM Journal on Computing*, 6(4):733–744, 1977. doi:10.1137/0206054.

4 Nathaniel Bryans, Ehsan Chiniforooshan, David Doty, Lila Kari, and Shinnosuke Seki. The power of nondeterminism in self-assembly. *Theory of Computing*, 9:1–29, 2013. doi:10.4086/TOC.2013.V009A001.

5 Anne Condon, Monir Hajiaghayi, and Chris Thachuk. Predicting minimum free energy structures of multi-stranded nucleic acid complexes is APX-hard. In *Proceedings of the 27th International Conference on DNA Computing and Molecular Programming*, volume 205 of *LIPIcs*, pages 9:1–9:21. Schloss Dagstuhl, 2021. doi:10.4230/LIPIcs.DNA.27.9.

6 Gwendal Ducloz, Ahmed Shalaby, and Damien Woods. Algorithmic hardness of the partition function for nucleic acid strands. In *Proceedings of the 31st International Conference on DNA Computing and Molecular Programming (DNA31)*, volume 347 of *LIPIcs*, pages 1:1–1:23. Schloss Dagstuhl, 2025. doi:10.4230/LIPIcs.DNA.31.1.

7 Antti Elonen, Ashwin Karthick Natarajan, Ibuki Kawamata, Lukas Oesinghaus, Abdulmelik Mohammed, Jani Seitsonen, Yuki Suzuki, Friedrich C. Simmel, Anton Kuzyk, and Pekka Orponen. Algorithm design of 3D wireframe RNA polyhedra. *ACS Nano*, 16:16608–16616, 2022.

8 Antti Elonen and Pekka Orponen. Designing 3D RNA origami nanostructures with a minimum number of kissing loops. In *Proceedings of the 31st International Conference on DNA Computing and Molecular Programming (DNA30)*, volume 314 of *LIPIcs*, pages 4:1–4:12. Schloss Dagstuhl, 2024. doi:10.4230/LIPIcs.DNA.30.4.

9 Antti Elonen, Leon Wimbes, Abdulmelik Mohammed, and Pekka Orponen. Dnaforge: A design tool for nucleic acid wireframe nanostructures. *Nucleic Acids Research*, 52(W1):W13–18, 2024.

10 Szilárd Zsolt Fazekas, Hwee Kim, Ryuichi Matsuoka, Shinnosuke Seki, and Hinano Takeuchi. On algorithmic self-assembly of squares by co-transcriptional folding. In *Proceedings of the 33rd International Symposium on Algorithms and Computation (ISAAC 2022)*, volume 248 of *LIPIcs*, pages 37:1–37:15. Schloss Dagstuhl, 2022. doi:10.4230/LIPIcs.ISAAC.2022.37.

11 Szilárd Zsolt Fazekas, Naoya Iwano, Yu Kihara, Ryuichi Matsuoka, Shinnosuke Seki, and Hinano Takeuchi. Towards composable computations by RNA co-transcriptional folding: A proof-of-concept demonstration of nested loops in oritatami. *Theoretical Computer Science*, 999:114550, 2024. doi:10.1016/J.TCS.2024.114550.

12 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W H Freeman & Co, 1979.

13 Cody Geary and Ebbe S. Andersen. Design principles for single-stranded RNA origami structures. In *Proceedings of the 20th International Conference on DNA Computing and Molecular Programming (DNA20)*, volume 8727 of *LNCS*, pages 1–19, 2014. doi:10.1007/978-3-319-11295-4_1.

14 Cody Geary, Guido Grossi, Ewan K. S. McRae, Paul W. K. Rothemund, and Ebbe S. Andersen. RNA origami design tools enable cotranscriptional folding of kilobase-sized nanoscaffolds. *Nature Chemistry*, 13:549–558, 2021.

15 Cody Geary, Pierre-Étienne Meunier, Nicolas Schabanel, and Shinnosuke Seki. Programming biomolecules that fold greedily during transcription. In *Proceedings of the 41st International Symposium on Mathematical Foundations of Computer Science (MFCS 2016)*, volume 58 of *LIPIcs*, pages 43:1–43:14, 2016. doi:10.4230/LIPIcs.MFCS.2016.43.

16 Cody Geary, Paul W. K. Rothemund, and Ebbe S. Andersen. A single-stranded architecture for cotranscriptional folding of RNA structures. *Science*, 345(6198):799–804, 2014.

17 Raymond Greenlaw, H. James Hoover, and Walter L. Ruzzo. *Limits to parallel computation: P-completeness theory*. Oxford University Press, 1995.

18 Yo-Sub Han, Hwee Kim, Makoto Ota, and Shinnosuke Seki. Nondeterministic seedless oritatami systems and hardness of testing their equivalence. *Natural Computing*, 17(1):67–79, 2018. doi:10.1007/S11047-017-9661-Y.

19 Richard E. Ladner. The circuit value problem is log space complete for p. *SIGACT News*, 7(1):18–20, 1975. doi:10.1145/990518.990519.

20 Mo Li, Mengxi Zheng, Siyu Wu, Cheng Tian, Di Liu, Yossi Weizmann, Wen Jiang, Guansong Wang, and Chengde Mao. In vivo production of RNA nanostructures via programmed folding of single-stranded RNAs. *Nature Communications*, 9:2196, 2018.

21 Kohei Maruyama and Shinnosuke Seki. Counting infinitely by oritatami co-transcriptional folding. *Natural Computing*, 20(2):329–340, 2021. doi:10.1007/S11047-021-09842-6.

22 Abdulmelik Mohammed, Pekka Orponen, and Sachith Pai. Algorithmic design of cotranscriptional folding of 2D RNA origami structures. In *Proceedings of the 17th International Conference on Unconventional Computation and Natural Computation (UCNC2018)*, volume 10867 of *LNCS*, pages 159–172. Springer, 2018. doi:10.1007/978-3-319-92435-9_12.

23 Ruth Nussinov, George Pieczenik, Jerrold R. Griggs, and Daniel J. Kleitman. Algorithms for loop matchings. *SIAM Journal on Applied Mathematics*, 35(1):68–82, 1978.

24 Pekka Orponen, Shinnosuke Seki, and Antti Elonen. Secondary structure design for cotranscriptional 3D RNA origami wireframes. In *Proceedings of the 31st International Conference on DNA Computing and Molecular Programming (DNA31)*, volume 347 of *LIPIcs*, pages 6:1–6:18. Schloss Dagstuhl, 2025. doi:10.4230/LIPIcs.DNA.31.6.

25 Daria Pchelina, Nicolas Schabanel, Shinnosuke Seki, and Guillaume Theyssier. Oritatami systems assemble shapes no less complex than tile assembly model (atam). In *Proceedings of the 39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022)*, volume 219 of *LIPIcs*, pages 51:1–51:23. Schloss Dagstuhl, 2022. doi:10.4230/LIPIcs.STACS.2022.51.

26 Erik Poppleton, Niklas Urbanek, Taniya Chakraborty, Alesasndra Griffo, Luca Monari, and Kerstin Göpfrich. RNA origami: Design, simulation and application. *RNA Biology*, 20(1):510–524, 2023.

27 Ahmed Shalaby and Damien Woods. An efficient algorithm to compute the minimum free energy of interacting nucleic acid strands. In *Proceedings of the 52nd International Colloquium on Automata, Languages, and Programming (ICALP2025)*, volume 334 of *LIPIcs*, pages 130:1–130:20. Schloss Dagstuhl, 2025. doi:10.4230/LIPIcs.ICALP.2025.130.