Towards the Algorithmic Molecular Self-Assembly of Fractals by Cotranscriptional Folding*

Yusei Masuda, Shinnosuke Seki**, and Yuki Ubukata

Department of Computer and Network Engineering, The University of Electro-Communications, 1-5-1, Chofugaoka, Chofu, Tokyo, 1828585, Japan

Abstract. RNA cotranscriptional folding has been recently proven capable of self-assembling a rectangular tile *in vitro* (RNA origami). The oritatami system (OS) is a novel computational model of cotranscriptional folding. In this paper, we initiate the theoretical study on the algorithmic self-assembly of shapes by cotranscriptional folding using the oritatami system. We propose an OS that folds into an arbitrary finite portion of the Heighway dragon, which is a fractal also-known as the paperfolding sequence $P = \mathrm{RRLRRLLR}\cdots$. The i-th element of P can be obtained by feeding i in binary to a 4-state DFA with output (DFAO). We implement this DFAO and a bit-sequence bifurcator as modules of oritatami system. Combining them with a known binary counter yields the proposed system.

1 Introduction

An RNA sequence, over nucleotides of four kinds `A`, `C`, `G`, `U`, is synthesized (*transcribed*) from its template DNA sequence over `A`, `C`, `G`, `T` nucleotide by nucleotide by an RNA polymerase (RNAP) enzyme according to the one-to-one mapping `A` $\to$ `U`, `C` $\to$ `G`, `G` $\to$ `C`, and `T` $\to$ `A` (for details, see, e.g., [2]). The yield, called *transcript*, starts folding immediately after it emerges from RNAP. This is the *cotranscriptional folding* (see Fig. 1). Geary, Rothemund, and Andersen have recently demonstrated the capability of cotranscriptional folding to self-assemble an RNA molecule of an intended shape at nano-scale [8]. They actually proposed an architecture of a DNA sequence whose transcript folds cotranscriptionally into an RNA tile of specific rectangular shape highly likely *in vitro*.

Algorithms and computation are fundamental to molecular self-assembly as illustrated in an enormous success of their use in DNA tile self-assembly (see, e.g., [4, 14, 17] and references therein). The Sierpinski triangle fractal was algorithmically self-assembled even *in vitro* from coalescence of DNA tiles that compute XOR [15]. Cotranscriptional folding exhibits highly sophisticated computational

* This work is in part supported by JST Program to Disseminate Tenure Tracking System, MEXT, Japan, No. 6F36 and by JSPS KAKENHI Grant-in-Aid for Young Scientists (A) No. 16H05854 to S. S.

** Corresponding author (`s.seki@uec.ac.jp`)

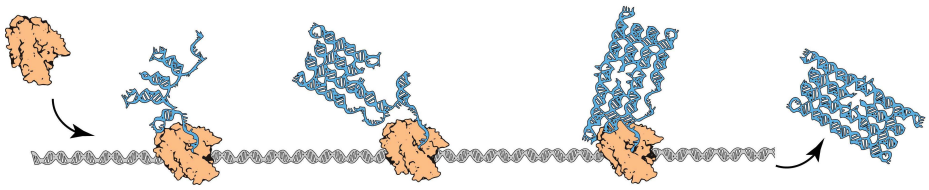

Fig. 1. RNA cotranscriptional folding. An RNA polymerase attaches to a template DNA sequence (gray spiral), scans it through, and synthesizes its RNA copy. The RNA sequence begins to fold upon itself immediately as it emerges from polymerase.

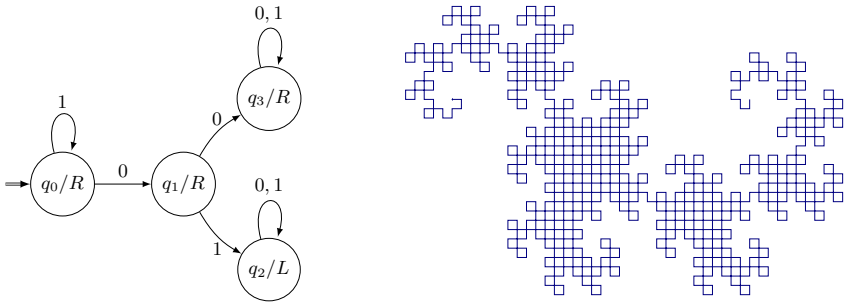

Fig. 2. (Left) DFAO to output the direction (L/R) of i-th turn of the Heighway dragon given $i \geq 0$ in binary from the LSB. (Right) The first $2^{10}-1$ turns of the dragon.

and algorithmic behaviors as well. Indeed, fluoride riboswitches in *Bacillus cereus* bacteria cotranscriptionally fold into a terminator stem or does not in order to regulate gene expression [16]. This is just one example but should be enough to signify the context-sensitivity of cotranscriptional folding and shapes thus self-assembled. Geary et al. have proved the capability of context-sensitivity to count in binary using a novel mathematical model of cotranscriptional folding called *oritatami system* (abbreviated as OS) [7].

We shall initiate theoretical study on algorithmic self-assembly of shapes by cotranscriptional folding using oritatami system. Sierpinski triangle would allow our study to borrow rich insights from the DNA tile self-assembly. However, in order to cut directly to the heart of algorithmic self-assembly by cotranscriptional folding, shapes of choice should be traversable somehow algorithmically. One such way is to feed a turtle program (see [1]) with an *automatic sequence* as commands (drawing a line segment, rotation, etc.), whose i-th bit can be obtained by giving i in binary from the least significant bit (LSB) to one DFA with output (DFAO) [3]. Shapes thus describable include the Heighway dragon [3] and von Koch curve [11]. A DFAO for the Heighway dragon (Fig. 2) outputs the following sequence, given $i = 0, 1, 2, \ldots$ in binary:

$$P = \mathrm{RRLRRLLRRRLLRLLRRRLRRLLLRRLLRLL}\cdots .$$

(The notation P is after its appellative *paperfolding sequence* [3].) For instance, $i = 2$ is given in binary from the LSB as 01, with which the DFAO transitions as $q_0 \to q_1 \to q_2$ and hence $P[2] = L$. A turtle should interpret an L (resp. R) as "move forward by unit distance and turn left (resp. right) 90 degrees." Any portion of the dragon can be represented by a factor of P; for instance, Fig. 2 (Right) depicts the factor $P[0..1022]$, i.e., the first $2^{10} - 1$ turns of the dragon.

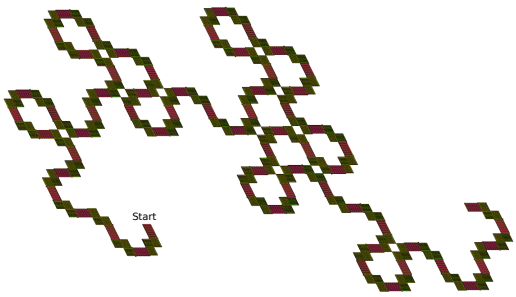

Fig. 3. The portion $P[0..62]$ of the Heighway dragon folded by the proposed oritatami system.

In this paper, we propose a generic design of oritatami system for the algorithmic cotranscriptional folding of an arbitrary finite portion of the Heighway dragon. Fig. 3 shows the portion $P[0..62]$ thus folded (the dragon is slanted but this is because the OS operates on the triangular grid). The transcript is a repetition of three modules: a catenation of binary counters, DFAO module, and turning module. The counter is a technical modification of the one proposed in [7]. By being fed with carry exactly once, the catenation increments the count i exactly by 1 while folding into a (red) line segment. At the end of the segment comes a DFAO module, which computes the turn direction $P[i]$, reinterprets it properly as A/O, and propagates the count i for the next turning module. A (green) L-shaped block is the turning module. It is a concatenation of three bit-sequence bifurcators, each of which folds into a rhombus, bifurcates i leftward as well as rightward, and guides further folding according to A/O.

The generic design proves the next theorem (for terminologies, see Sect. 2).

Theorem 1. *For any finite portion $P[i..j]$ of the Heighway dragon, there exist an integer c and a deterministic cyclic oritatami system of delay 3 and arity 3 that weakly folds into the c-rhombus scaling of $P[i..j]$.*

A JavaScript program to run this OS is available at `https://wolves13.github.io`.

2 Preliminaries

Let Σ be a set of types of abstract molecules, or *beads*, and Σ^* be the set of finite sequences of beads. A bead of type $a \in \Sigma$ is called an a-bead. Let $w = b_1 b_2 \cdots b_n \in \Sigma^*$ be a string of length n for some integer n and bead types $b_1, \ldots, b_n \in \Sigma$. The *length* of w is denoted by $|w|$, that is, $|w| = n$. For two indices i, j with $1 \leq i \leq j \leq n$, we let $w[i..j]$ refer to the subsequence $b_i b_{i+1} \cdots b_{j-1} b_j$; if $i = j$, then we simplify $w[i..i]$ as $w[i]$. For $k \geq 1$, $w[1..k]$ is called a *prefix* of w.

Oritatami systems fold their transcript, a sequence of beads, over the triangular grid as suggested in Fig. 4 cotranscriptionally based on hydrogen-bond-based

interactions (*h-interactions* for short) which their own rule set allow for between adjacent beads of particular types. Let $\mathbb{T} = (V, E)$ be the triangular grid graph. A directed path $P = p_1 p_2 \cdots p_n$ in $\mathbb{T}$ is a sequence of *pairwise-distinct* points $p_1, p_2, \ldots, p_n \in V$ such that $\{p_i, p_{i+1}\} \in E$ for all $1 \leq i < n$. Its i-th point is referred to as $P[i]$. A *rule set* $\mathcal{H} \subseteq \Sigma \times \Sigma$ is a symmetric relation over the set of pairs of bead types. A (finite) *conformation* C is a triple (P, w, H) of a directed path P in $\mathbb{T}$, $w \in \Sigma^*$ of the same length as P, and a set of h-interactions $H \subseteq \{\{i, j\} \mid 1 \leq i, i+2 \leq j, \{P[i], P[j]\} \in E\}$. This is to be interpreted as the sequence w being folded in such a manner that its i-th bead $w[i]$ is placed on the i-th point $P[i]$ along the path and there is an h-interaction between the i-th and j-th beads if and only if $(i, j) \in H$. The condition $i + 2 \leq j$ represents the topological restriction that two consecutive beads along the path cannot h-interact with each other. The conformation C is *$\mathcal{H}$-valid* if for all h-interaction $(i, j) \in H$, $(w[i], w[j]) \in \mathcal{H}$. For an integer $\alpha \geq 1$, C is *of arity* α if the maximum number of h-interactions per bead is α, that is, if for any $k \geq 1$, $|\{i \mid (i, k) \in H)\}| + |\{j \mid (k, j) \in H\}| \leq \alpha$ and the both sides become equal for some k. By $\mathcal{C}_{\leq \alpha}$, we denote the set of all conformations of arity at most α.

Oritatami systems grow conformations by elongating them under their own rule set. Given a rule set $\mathcal{H}$ and an $\mathcal{H}$-valid conformation $C_1 = (P, w, H)$, we say that another conformation C_2 is an *elongation of C_1 by a bead $b \in \Sigma$*, written as $C_1 \xrightarrow{\mathcal{H}}_b C_2$, if $C_2 = (Pp, wb, H \cup H')$ for some point p not along the path P and set of h-interactions $H' \subseteq \big\{\{i, |w|+1\} \,\big|\, 1 \leq i < |w|, \{P[i], p\} \in E, (w[i], b) \in \mathcal{H}\big\}$, which can be empty. Note that C_2 is also $\mathcal{H}$-valid. This operation is recursively extended to the elongation by a finite sequence of beads as: for any conformation C, $C \xrightarrow{\mathcal{H}}{}^*_\lambda C$; and for a finite sequence of beads $w \in \Sigma^*$ and a bead $b \in \Sigma$, a conformation C_1 is elongated to a conformation C_2 by wb, written as $C_1 \xrightarrow{\mathcal{H}}{}^*_{wb} C_2$, if there is a conformation C' that satisfies $C_1 \xrightarrow{\mathcal{H}}{}^*_w C'$ and $C' \xrightarrow{\mathcal{H}}_b C_2$.

A finite *oritatami system* (OS) is a 5-tuple $\Xi = (\mathcal{H}, \alpha, \delta, \sigma, w)$, where $\mathcal{H}$ is a rule set, α is an arity, $\delta \geq 1$ is a parameter called the *delay*, σ is an initial $\mathcal{H}$-valid conformation of arity α called the *seed*, upon which its finite *transcript* $w \in \Sigma^*$ is to be folded by stabilizing beads of w one at a time so as to minimize energy collaboratively with the succeeding $\delta - 1$ nascent beads. The energy of a conformation $C = (P, w, H)$, denoted by $\Delta G(C)$, is defined to be $-|H|$; the more h-interactions a conformation has, the more stable it gets. The set $\mathcal{F}(\Xi)$ of conformations *foldable* by this system is recursively defined as: the seed σ is in $\mathcal{F}(\Xi)$; and provided that an elongation C_i of σ by the prefix $w[1..i]$ be foldable (i.e., $C_0 = \sigma$), its further elongation C_{i+1} by the next bead $w[i+1]$ is foldable if

$$C_{i+1} \in \operatorname*{arg\,min}_{\substack{C \in \mathcal{C}_{\leq \alpha}\, s.t. \\ C_i \xrightarrow{\mathcal{H}}_{w[i+1]} C}} \min \Big\{ \Delta G(C') \,\Big|\, C \xrightarrow{\mathcal{H}}{}^*_{w[i+2 \ldots i+k]} C', k \leq \delta, C' \in \mathcal{C}_{\leq \alpha} \Big\}. \quad (1)$$

We say that the bead $w[i+1]$ and the h-interactions it forms are *stabilized* according to C_{i+1}. Note that an arity-α OS cannot fold any conformation of arity larger than α. A conformation foldable by Ξ is *terminal* if none of its

elongations is foldable by Ξ. The OS Ξ is *deterministic* if for all $i \geq 0$, there exists at most one C_{i+1} that satisfies (1).

Let us provide an example of deterministic OS that folds into a motif of great use called the *glider*. Let $\Sigma = \{a, a', b, b', \bullet\}$. Consider a delay-3 OS whose seed is colored in red in Fig. 4, whose transcript w is a repetition of $a \bullet b'b \bullet a'$, and whose rule set is $\mathcal{H} = \{(a, a'), (b, b')\}$, which makes $\bullet$-beads inert.

By the fragment $w[1..3] = a \bullet b'$, the seed can be elongated in various ways, only three of which are shown in Fig. 4 (left). The only bead on the fragment capable of a new h-interaction is b' (with a b-bead according to $\mathcal{H}$), and for that, the fragment must be folded as bolded in Fig. 4 (left). The first bead $w[1] = a$ is hence stabilized to the east of the previous bead, and then the bead $w[4] = b$ is transcribed. The next two beads $w[2], w[3]$ are stabilized as illustrated one after another, but we can easily see that the bolded elongation dominates even their stabilization. It suffices to observe that neither $w[4]$ nor $w[5]$ can form any new h-interaction. When $w[4] = b$ is transcribed (after $w[1]$ is stabilized), b'-beads around are either too far or too close (recall that $w[4]$ cannot interact with $w[3] = b'$). In addition, $w[5]$ is inert. Thus, they cannot override the bolded "decision." It is easily induced inductively that gliders of arbitrary "flight distance" can be folded.

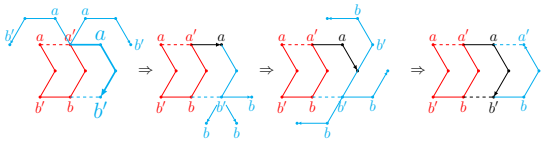

Fig. 4. Progression of a glider by distance 1.

Gliders also provide a medium to propagate 1-bit at arbitrary distance as the position of their last beads, which is determined by the height (top or bottom) of the first bead and a flying distance. For instance, the glider in Fig. 4 launches top and thus its last bead (the a') also comes top after traveling the distance 2. The OS we shall propose exploits this information-carrying capability.

A *shape* is a set of points on the triangular grid. For an integer $c \geq 1$, the *c-rhombus scaling* of a shape S is a shape obtained by expanding the distance between the points in S equally by c to obtain another shape S' and replacing each point $\boldsymbol{p} \in S'$ by the rhombus whose sides are of length c. Let $\diamondsuit_c(S)$ be its c-rhombus scaling of a shape S. We say that an OS Ξ *weakly folds* (or "self-assembles") $\diamondsuit_c(S)$ if every terminal assembly of Ξ puts its bead only at positions in $\diamondsuit_c(S)$ and it puts at least one bead in all rhombuses of $\diamondsuit_c(S)$.

3 Heighway dragon oritatami system

We propose a generic design of deterministic oritatami system that allows us to fold an arbitrary finite portion of the Heighway dragon. The folded dragon is actually slanted as illustrated in Fig. 3, which is more natural than the conventional (upright) one to be folded over the triangular grid. Independently of which portion to be folded, the design sets both delay and arity to 3 and employs 567 bead types and a fixed rule set $\mathcal{H}$ (some of the bead types could be saved but not

easily due to the NP-hardness of saving [9]). The design also challenges to make its transcript periodic; a periodic RNA transcript is likely to be transcribed out of a circular DNA sequence [5]. Without requiring the periodicity, one could simply design left and right-turn modules and concatenate their copies according to the sequence P. Such a "hardcoding" goes against the spirit of algorithmic self-assembly, and an OS that folds into the infinite Heighway dragon, if any, could not take this approach in order to be describable by a finite mean.

One period folds into successive two line segments of the dragon. Why does the design distinguish them? The answer lies in that the dragon is slanted. The slanted dragon involves two types of left turn as well as two types of right turn: acute and obtuse. Capability of one turning module to make all of the four possible turns would halve the period but require quite a number of conformations. Recall that what the module has to turn is not a straw but a thick wire through which the count i propagates. Such a module is too advanced for the current oritatami technology. Our approach makes it enough for a turning module to handle just two tasks: turning acutely and obtusely. Observe that after (slanted) vertical segments, the dragon turns left obtusely and right acutely, whereas after horizontal ones, it turns left acutely and right obtusely. Moreover, vertical and horizontal segments alternate on the dragon. Thus, each segment can be equipped with a proper interpreter *a priori* of the direction L/R into a signal A(cute)/O(btuse). Two types of interpreters are hence needed: the vertical one interprets L as O and R as A, while the horizontal one does conversely.

The first and second halves of a period differ only in the type of interpreter (AO or $\overline{\text{AO}}$) so that we just explain the first. Its transcript folds as abstracted in Fig. 5 into one line segment plus one turn of the dragon. The transcript consists of three subsequences (modules) responsible for the following tasks, respectively:

1. (Counter module) $i \leftarrow i + 1$ and propagate i, drawing a line segment;
2. (DFAO module) Compute $P[i]$ and interpret it properly as A or O;
3. (Turning module) Make a turn accordingly.

Before explaining them, one issue intrinsic to the folding by oritatami systems should be pointed. It rises when the dragon makes a turn where it has already turned before. By definition, OSs cannot put a bead anywhere occupied by another bead. Being scaled sufficiently large, a rhombus corresponding to a point affords two turning modules, which otherwise collide, as long as they fold into an L-shape (Figs. 3 and 5). A turning module has its three bifurcator submodules direct further folding obtusely one after another when it receives O from the previous DFAO module (colored in blue in Fig. 5) or acutely when it receives A (yellow), and folds into two L-shape conformations. Note that two turns which share a point are both acute or both obtuse.

Modules Having outlined the generic design, now we explain how the design implements an OS for a specific target portion $P[j_1..j_2]$, or more precisely, how the three modules of the OS and their components are implemented, interlocked with each other, and collaborate. Let $n = \min\{m \mid j_2 < 2^m\}$. The counter

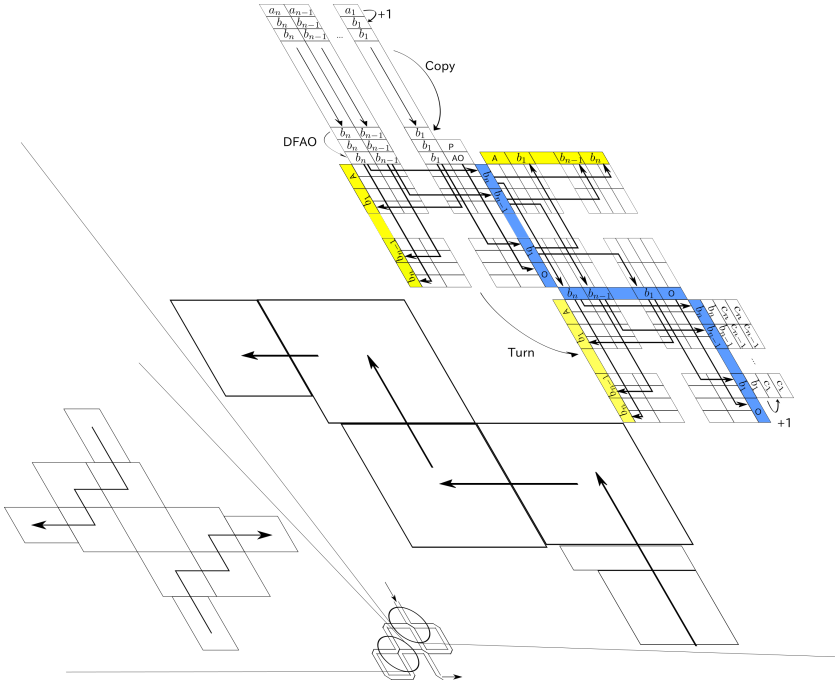

Fig. 5. Folding of one segment plus turn of the Heighway dragon, flow of information through it, and the two ways of collision avoidance between two turns.

and turning modules, or more precisely, their transcripts, are of length $O(n^2)$ while the DFAO module is of length $O(n)$. Thus, the period of the OS is of length proportional to n^2. They consist of components of constant size. Using s simulator developed for [10], we verified that all of the components fold correctly in all possible environments, which are abstracted in Figs. 5, 7, and 12.

Counter module The existing binary counter [7] was modified so as to operate in the dynamics (1), which is more prevailing [9, 10, 13] though less tractable.

A counter module for n-bit sequences is a catenation of the following components in the order: right-turner, n half-adders, left-turner, and n formatters. It folds into one zigzag. Suppose each bit of an n-bit input $a_n a_{n-1} \cdots a_1$ is given as a specific sequence of 4-bead types as suggested in Fig. 6 and the input is provided as the catenation of such bit sequences for $a_n, a_{n-1}, \ldots, a_1$. The right-

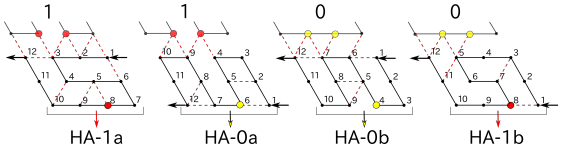

Fig. 6. All conformations of the half-adder. The first and third are diverted to implement the body-lpx2 component of the turning module.

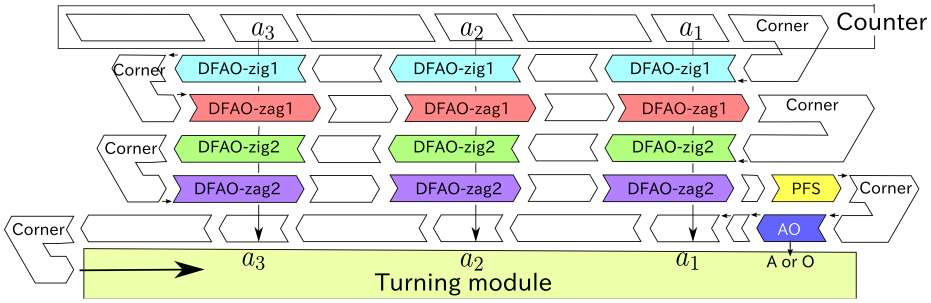

Fig. 7. Component-level abstraction of the folding of DFAO module.

turner folds so as for the first bead of the first half-adder to come at a distance of either 1 or 3 southeastward from the rightmost bead of the LSB a_1. Thus, the first half-adder encounters only the four surrounding environments (contexts) shown in Fig. 6, in which the half-adder, or actually its transcript `1-2-...-12`, folds deterministically as illustrated, or more precisely, the generic rule set $\mathcal{H}$ is designed so as for the transcript to favor the illustrated conformation the most in each context. Interpret the first bead `1` (resp. last bead `12`) being far from the input as carry-in (resp. carry-out) and close as no carry-in (resp. no carry-out). Then we can see carry be out only when $a_1 = 1$ and carry is in (the second context in Fig. 6). Being concatenated, half-adders ripple carry from one to the other. The half-adder can expose four sequences below: `HA-1a`, `-0a`, `-0b`, and `-1b` as output. They are pairwise distinct enough for a formatter, which is supposed to come below the output in the next zag, to favor one of its two conformations if the output is `HA-1a/1b` or the other `HA-0` otherwise, which expose 1 and 0 downward, respectively. Thus, a counter module increments an input by 1 if its first half-adder is fed with carry (i.e., the right-turner locates its `1` away from a_1), or just propagates it otherwise. Concatenating counter modules and feeding carry-in only to the first one yields a line segment through which the current count i is incremented by 1 and propagated through.

DFAO module As abstracted in Fig. 7, a DFAO module receives the current count i from the previous counter module, computes $P[i]$, interprets it properly either A or O, and outputs it together with the count i.

Observe in Fig. 2 that what the DFAO for the Heighway dragon does exactly in order to compute $P[i]$ is to search for the first 0 read from the LSB and check whether it is followed by 0 ($P[i] = \mathrm{R}$) or by 1 ($P[i] = \mathrm{L}$). While folding into zigzag twice and one more zig (see Fig. 7), the DFAO module does the search and check in the first and second zigzags, respectively. It employs the following six components: DFAO-zig1, -zag1, -zig2, -zag2, PFS, and AO (or $\overline{\mathrm{AO}}$). The last zig is equipped with AO if this module is for a vertical segment or with $\overline{\mathrm{AO}}$ otherwise; as shown in Fig. 8, AO interprets the output L of PFS as obtuse and R as acute, while $\overline{\mathrm{AO}}$ interprets them the other way around.

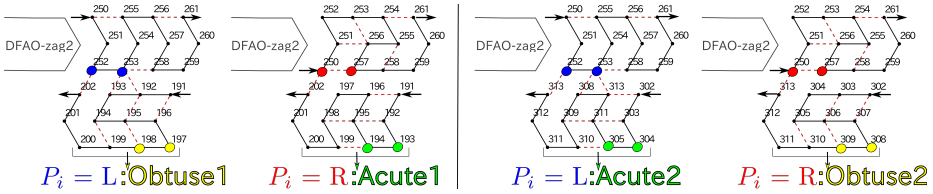

Fig. 8. The two conformations of PFS above and the corresponding two conformations of (left) AO and (right) of $\overline{\text{AO}}$.

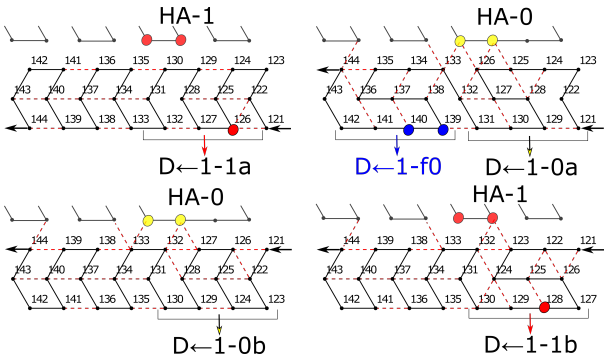

Fig. 9. The four conformations of DFAO-zig1: (top) Dzig1-1 and Dzig1-f0; (bottom) Dzig1-20 and Dzig1-21.

DFAO-zig1s detect the first 0 collaboratively in two phases. See Fig. 9 for its possible contexts and corresponding conformations. Phase1 is to copy all the 1's prior to the first 0 and Phase2 is to copy all the bits after the first 0. These phases are distinguished by the relative position where a DFAO-zig1 starts folding to the input (far in Phase 1 while close in Phase 2). In Phase1, DFAO-zig1s certainly take the conformation Dzig1-1 (the top left one in Fig. 9). At the first 0, a DFAO-zig1 folds into Dzig1-f0 instead, ending at the top in order to transition to Phase 2. Each of the succeeding DFAO-zig1s takes Dzig1-20 or -21 to copy all the remaining bits. Note that there is a cushion between two DFAO-zig1s called *spacer*. Spacers are used to prevent undesirable interference among components. They are implemented as a glider (see Sect. 2), hence capable of propagating 1bit on which phase the system is in. In the first zag, DFAO-zag1s just propagate 0's, 1's, and the first 0 using conformations in Fig. 15.

In the second zig, DFAO-zig2s check whether the first 0 is followed (being read from LSB) by 0 or 1, in a similar manner to the search for the first 0. They usually take one of the two conformations Dzig2-1 and Dzig2-0 to copy 1's and 0's, which start and end at the bottom (see Fig. 10). At the encounter to the first 0, a DFAO-zig2 folds into a special conformation Dzig2-f0 and ends at the top. The next DFAO-zig2, if any, starts folding at the top so that it takes the special conformation Dzig2-0f0 if the first 0 is followed by 0 or Dzig2-1f0 if it is followed by 1. Recall that reading 1 here is equivalent to transitioning to q_2 in the DFAO, that is, $P[i] = \text{L}$. Dzig2-1f0 exposes a marker q_2 downward. These conformations end at the bottom so that the remaining bits are copied by the ordinary conformations. The second zag starts at the bottom and copy 0's and 1's by the two conformations Dzag2-L0 and -L1 of DFAO-zag2 (top left

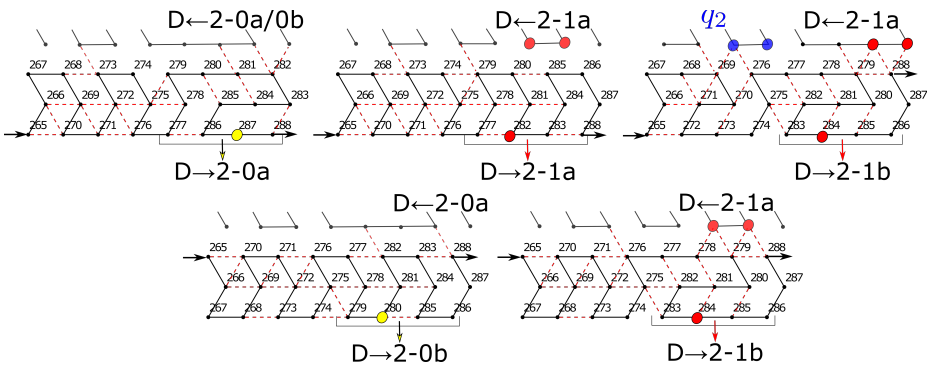

Fig. 10. The five conformations of DFAO-zag2: (top) Dzag2-L0, Dzag2-L1, Dzag2-T1, (bottom) Dzag2-R0, and Dzag2-R1. The first and second halves are diverted to implement the body-rgy and body-gx components of the turning module, respectively.

and center in Fig. 11) until a DFAO-zag2 encounters the 1 marked by q_2, if any. At the encounter, the DFAO-zag2 takes the special conformation Dzag2-T1 and changes the ending position to the top, letting the remaining DFAO-zag2s rather take Dzag2-R0 and -R1 for copying, which end at the top. As such, the second zag can feed $P[i]$ to PFS as the position of its first bead.

Turning module The last module is for turn. It consists of two submodules: bit-sequence bifurcator and steering arm (shaded in light blue in Fig. 12).

The bifurcator bifurcates the bits of the current count i and the interpreted signal (A or O) as shown in Fig. 5 while folding into zigzags. For that, it employs components to handle the following four types of tasks:

1. propagate 1-bit vertically: body-rpx (Fig. 16 (left)), body-lpx1 (Fig. 16 (right)), and body-lpx2 (Fig. 6);
2. let 1-bit cross another 1-bit: body-gx (Fig. 11);
3. fork 1-bit vertically and horizontally: body-rgy (Fig. 11) and body-lgy (Fig. 17);
4. undergo transition between a zig and a zag and exposes 1-bit outside: turn-rgp (Fig. 13) and turn-lgp (Fig. 18).

Components to handle the first two types of tasks have already been implemented (see, e.g., [10]) so that we shall explain the others.

The component body-rgy is implemented by reusing the first half of DFAO-zag2 (Fig. 11). Starting from the bottom, it can take two conformations which end at different heights and expose sequences of bead types sufficiently pairwise distinct downward. Hence, we can divert it to

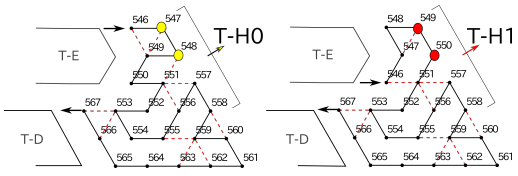

Fig. 12. The two conformations of turn-rgp.

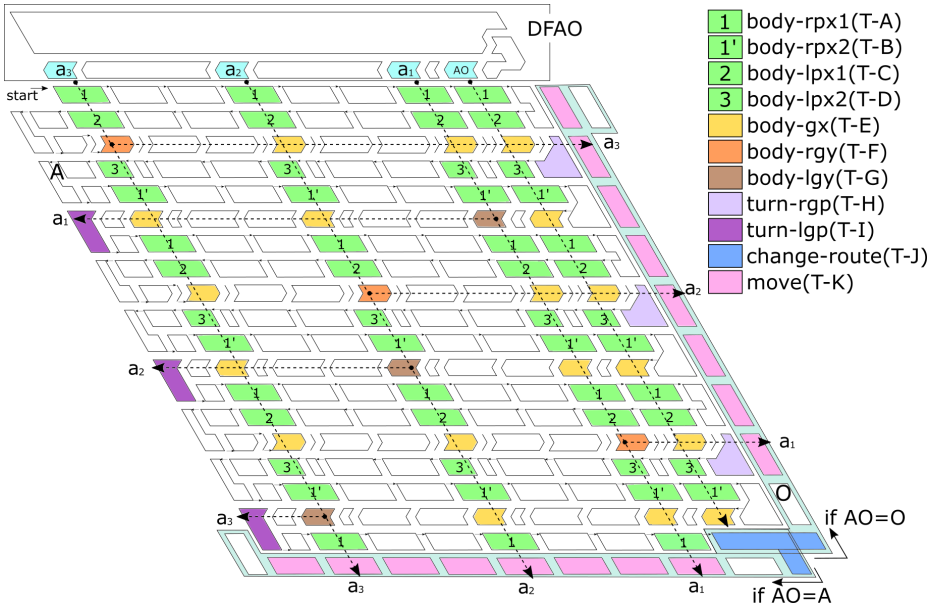

Fig. 11. Component-level abstraction of folding of turning module. All the white components in the middle are spacers, some of which are implemented in the shape of parallelogram instead of glider.

fork 1bit input rightward and downward. The 1bit thus forked transfers till the end of a zag and is converted by the turn-rgp into a sequence of bead types (see Fig. 13). The body-lgy and turn-lgp are their zig counterparts (Figs. 17 and 18).

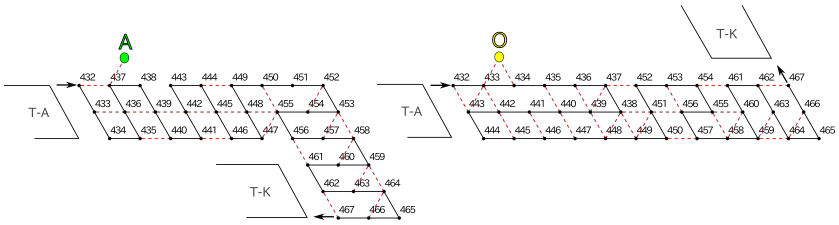

Fig. 13. The two conformations of change-route.

The bifurcator also propagates the 1-bit A/O, output by the DFAO module, to tell the steering arm which way it should take. Specifically, the signal has the component, change-route, of the steering arm take one of the two conformations in Fig. 14, guiding the rest of the arm towards the specified direction. The remaining arm is a catenation of move components (Fig. 19), which is capable of letting the bifurcated bit sequence through. Note that the turning module need

not bifurcate AO. Indeed, the second and third turning modules are supposed to turn in the same manner as the first one. It hence suffices to append A and O to the bifurcated bit sequences on the acute side and obtuse side, respectively.

Remark 1. As suggested in Fig. 12, the bifurcator actually outputs an input even downward. That is, it actually serves as a trifurcator. This provides a more space-efficient way to replicate a bit sequence many-folds.

Acknowledgements We would like to thank Hwee Kim and Aleck Johnsen for their helpful advices and discussions.

References

1. Abelson, H., diSessa, A.: Turtle Geometry The Computer as a Medium for Exploring Mathematics. MIT Press Series in Artificial Intelligence, The MIT Press (1981)
2. Alberts, B., Johnson, A., Lewis, J., Morgan, D., Raff, M., Roberts, K., Walter, P.: Molecular Biology of the Cell. Garland Science, 6th edn. (2014)
3. Allouche, J.P., Shallit, J.: Automatic Sequences: Theory, Applications, Generalizations. Cambridge University Press (2003)
4. Doty, D.: Theory of algorithmic self-assembly. Communications of the ACM 55(12), 78–88 (2012)
5. Geary, C., Andersen, E.S.: Design principles for single-stranded RNA origami structures. In: Proc. DNA20. vol. LNCS 8727, pp. 1–19. Springer (2014)
6. Geary, C., Meunier, P.E., Schabanel, N., Seki, S.: Efficient universal computation by greedy molecular folding (2015), coRR abs/1508.00510
7. Geary, C., Meunier, P.E., Schabanel, N., Seki, S.: Programming biomolecules that fold greedily during transcription. In: Proc. 41st International Symposium on Mathematical Foundations of Computer Science (MFCS2016). pp. 43:1–43:14. Leibniz International Proceedings in Informatics (LIPIcs) 58 (2016)
8. Geary, C., Rothemund, P.W.K., Andersen, E.S.: A single-stranded architecture for cotranscriptional folding of RNA nanostructures. Science 345(6198), 799–804 (2014)
9. Han, Y.S., Kim, H.: Ruleset optimization on isomorphic oritatami systems. In: Proc. 23rd International Conference on DNA Computing and Molecular Programming (DNA23). LNCS, Springer (2017), in press
10. Han, Y.S., Kim, H., Ota, M., Seki, S.: Nondeterministic seedless oritatami systems and hardness of testing their equivalence. In: Proc. 22nd International Conference on DNA Computing and Molecular Programming (DNA22). LNCS, vol. 9818, pp. 19–34. Springer (2016)
11. Ma, J., Holdener, J.: When Thue-Morse meets Koch. Fractals 13, 191–206 (2005)
12. Masuda, Y., Seki, S., Ubukata, Y.: The simulator of heighway dragon
13. Ota, M., Seki, S.: Ruleset design problems for oritatami systems. Theoretical Computer Science 671, 26–35 (2017)
14. Patitz, M.J.: Self-assembly of fractals. In: Encyclopedia of Algorithms, pp. 1918–1922. Springer (2016)
15. Rothemund, P.W.K., Papadakis, N., Winfree, E.: Algorithmic self-assembly of DNA Sierpinski triangle. PLoS Biology 2(12), e424 (2004)

16. Watters, K.E., Strobel, E.J., Yu, A.M., Lis, J.T., Lucks, J.B.: Cotranscriptional folding of a riboswitch at nucleotide resolution. Nature Structural & Molecular Biology 23(12), 1124–1133 (2016)
17. Winfree, E.: Algorithmic Self-Assembly of DNA. Ph.D. thesis, California Institute of Technology (June 1998)

Appendix

A Figures omitted

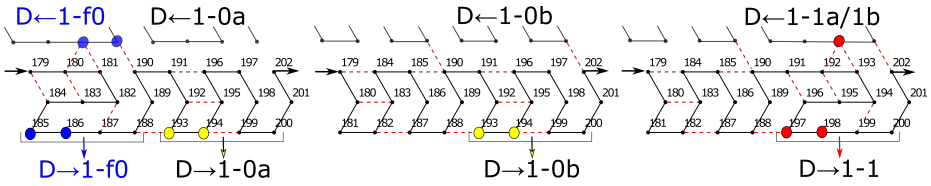

Fig. 14. The three conformations of DFAO-zag1: Dzag1-f0, Dzag1-0, and Dzag1-1.

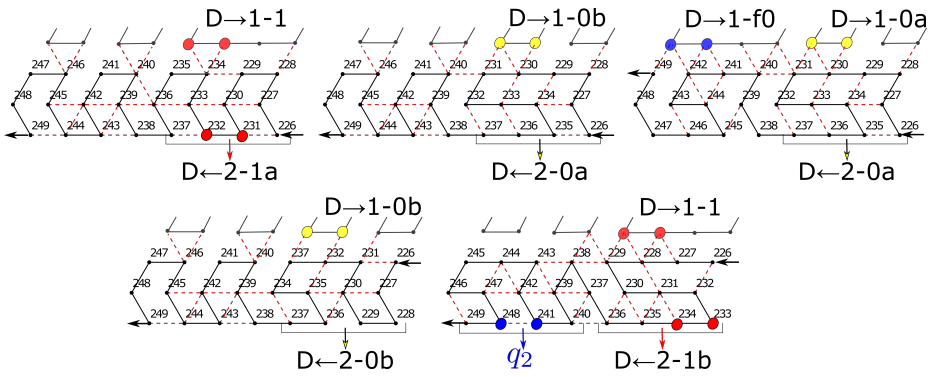

Fig. 15. The five conformations of DFAO-zig2: (top) Dzig2-1, Dzig2-0, Dzig2-f0, (bottom) Dzig2-0f0, and Dzig2-1f0.

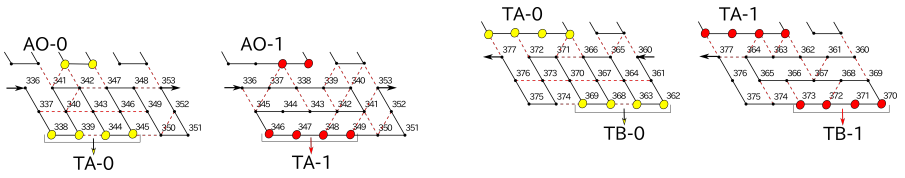

Fig. 16. (Left) The two conformations of body-rpx component. (Right) The two conformations of body-lpx1 component.

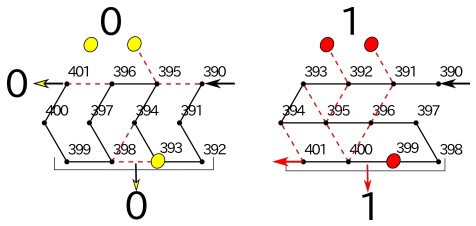

Fig. 17. The two conformations of body-lgy component.

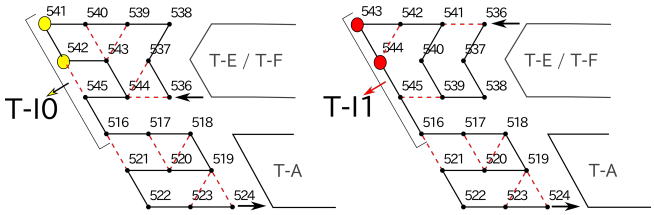

Fig. 18. The two conformations of turn-lgp component.

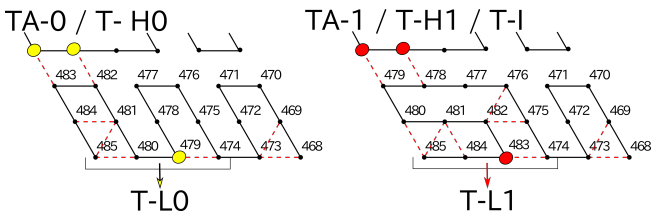

Fig. 19. The two conformations of move component.