

Nondeterministic Seedless Oritatami Systems and Hardness of Testing Their Equivalence

Yo-Sub Han¹, Hwee Kim^{1(✉)}, Makoto Ota², and Shinnosuke Seki²

¹ Department of Computer Science, Yonsei University, 50 Yonsei-Ro, Seodaemun-Gu, Seoul 120-749, Republic of Korea
{emmous,kimhwee}@yonsei.ac.kr

² University of Electro-Communications, 1-5-1, Chofugaoka, Chofu, Tokyo 182-8585, Japan
o1111032@edu.cc.uec.ac.jp, s.seki@uec.ac.jp

Abstract. The oritatami system (OS) is a model of computation by cotranscriptional folding, being inspired by the recent experimental success of RNA origami to self-assemble an RNA tile cotranscriptionally. The OSs implemented so far, including binary counter and Turing machine simulator, are deterministic, that is, uniquely fold into one conformation, while nondeterminism is intrinsic in biomolecular folding. We introduce nondeterminism to OS (NOS) and propose an NOS that chooses an assignment of Boolean values nondeterministically and evaluates a logical formula on the assignment. This NOS is seedless in the sense that it does not require any initial conformation to begin with like the RNA origami. The NOS allows to prove the co-NP hardness of deciding, given two NOSs, if there exists no conformation that one of them folds into but the other does not.

1 Introduction

In nature, an one-dimensional RNA sequence—a primary structure—folds itself autonomously and forms a more complex secondary structure. It has been a constant question to predict the secondary structure from a given primary structure, and based on experimental observations, researchers established various RNA secondary structure prediction models including RNAfold [12], Pknets [9], mFold [11] and KineFold [10]. Traditional models tend to rely on the energy optimization of the whole structure.

Recently, biochemists showed that the kinetics—the step-by-step dynamics of the reaction—plays an essential role in the geometric shape of the RNA foldings [2], since the folding caused by intermolecular reactions is faster than the RNA transcription rate [7]. By controlling cotranscriptional foldings, researchers succeeded in cotranscriptionally assembling a rectangular tile out of RNA, which is called RNA Origami [5] as depicted in Fig. 1. From this kinetic point of view, Geary et al. [4] proposed a new folding model called the oritatami system (OS). In general, OS defines a sequence of beads (which is the primary structure) and a set of rules for possible intermolecular reactions between beads. For each bead

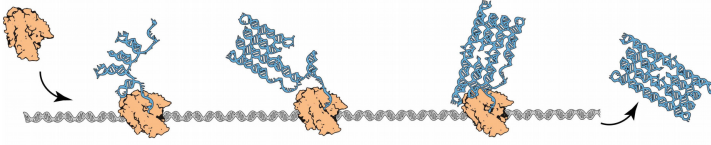


Fig. 1. RNA Origami [5]. The artwork is by Cody Geary.

in the sequence, the system takes a lookahead of a few upcoming beads and determines the best location of the bead that maximizes the number of possible interactions from the lookahead. Note that the lookahead represents the reaction rate of the cotranscriptional folding and the number of interactions represents the energy level. In OS, we call the secondary structure *the conformation*, and the resulting secondary structure *the terminal conformation*.

Geary et al. implemented an OS to count in binary [3] and an OS to simulate a cyclic tag system [4]. These OSs uniquely folds into one conformation, and in this sense, they are deterministic. In contrast, nondeterminism is intrinsic in biomolecular folding. Therefore, we define the nondeterministic OS (NOS) in this paper, and examine its power. It turns out that nondeterminism can be made use of for OSs to execute randomized algorithms. We propose an NOS that evaluates a Boolean formula in disjunctive normal form (DNF formula) on a random assignment. This NOS is in fact seedless like the RNA origami. More importantly, the NOS proves the co-NP hardness of the OSEQ problem, which asks, given two NOSs, if there exists no conformation that one folds into but the other does not (Theorem 2). The equivalence test is indispensable in the optimization of the design of a given OS. As we will see, the equivalence of two deterministic OSs is testable in linear time (Theorem 1).

2 Preliminaries

Let Σ be a set of bead types, and Σ^* be the set of finite strings of beads, i.e., strings over Σ , including the empty string λ . Let $w = a_1a_2 \cdots a_n$ be a string of length n for some integer n and bead types $a_1, \dots, a_n \in \Sigma$. The *length* of w is denoted by $|w|$. For two indices i, j with $1 \leq i \leq j \leq n$, we let $w[i, j]$ refer to the substring $a_i a_{i+1} \cdots a_{j-1} a_j$; if $i = j$, then we denote it by $w[i]$ instead. We use w^n to denote the string $\underbrace{ww \cdots w}_n$.

Oritatami systems operate on the hexagonal lattice. The *grid graph* of the lattice is the graph whose vertices correspond to the lattice points and connected if the corresponding lattice points are at unit distance hexagonally. For a point p and a bead type $a \in \Sigma$, we call the pair (p, a) an *annotated point*, or simply a *point* if being annotated is clear from context. Two annotated points $(p, a), (q, b)$ are *adjacent* if pq is an edge of the grid graph.

A *path* is a sequence $P = p_1 p_2 \cdots p_n$ of *pairwise-distinct* points p_1, p_2, \dots, p_n such that $p_i p_{i+1}$ is at unit distance for all $1 \leq i < n$. Given a string $w \in \Sigma^*$ of

bead types of length n , a *path annotated by w* , or simply *w -path*, is a sequence P_w of annotated points $(p_1, w[1]), \dots, (p_n, w[n])$, where $p_1 \cdots p_n$ is a path. Annotated points of the w -path are regarded as a bead, and hence, we call them beads and, in particular, we call the i -th point $(p_i, w[i])$ the i -th bead of the w -path.

Let $\mathcal{H} \subseteq \Sigma \times \Sigma$ be a symmetric relation, specifying between which types of beads can form a hydrogen-bond-based interaction (h-interaction for short). This relation \mathcal{H} is called the *ruleset*. It is convenient to assume a special *inert* bead type $\bullet \in \Sigma$ that never forms any h-interaction according to \mathcal{H} .

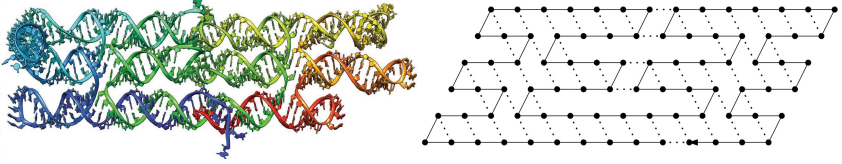


Fig. 2. (Left) An example of an RNA tile generated by RNA Origami. (Right) A conformation representing the RNA tile in OS. The directed solid line represents a path, dots represent beads, and dotted lines represent h-interactions. The idea and artwork were provided by Cody Geary.

A *conformation* C is a pair of a w -path $P_w = (p_1, w[1])(p_2, w[2]) \cdots$ and a set H of h-interactions, where $H \subseteq \{\{i, j\} \mid 1 \leq i, i+2 \leq j\}$ and $\{i, j\} \in H$ implies that the i -th and j -th beads of the w -path form an h-interaction between them. An example conformation is found in Fig. 2 (Right). The condition $i+2 \leq j$ represents the topological restriction that two beads $(p_i, w[i]), (p_{i+1}, w[i+1])$, adjacent to each other on the w -path, cannot form an h-interaction between them. We say C is *finite* if its path is finite. From now on, when a conformation is illustrated, any unlabeled bead is assumed to be labeled with \bullet , that is, be inert. For an integer $\alpha \geq 1$, C is of *arity* α if none of its beads interact with more than α beads. On the hexagonal lattice where every point is adjacent to six points, $\alpha > 6$ is merely meaningless, but on another lattice larger α 's may. Let \mathcal{C}_α be the set of all conformations of arity- α .

A rule (a, b) in the ruleset \mathcal{H} is *used* in the conformation C if there exists $\{i, j\} \in H$ such that $w[i] = a$ and $w[j] = b$ or $w[i] = b$ and $w[j] = a$. A conformation C is *valid* (with respect to \mathcal{H}) if for all $\{i, j\} \in H$, $(w[i], w[j]) \in \mathcal{H}$. In a context with one fixed ruleset, only valid conformations with respect to the ruleset are considered, and we may not specify with respect to what ruleset they are valid.

Given a ruleset \mathcal{H} and a valid finite conformation $C_1 = (P_w, H)$ with respect to \mathcal{H} , we say that another conformation C_2 is an *elongation* of C_1 by a bead $a \in \Sigma$ if $C_2 = (P_w \cdot (p, a), H \cup H')$ for some lattice point p and (possibly empty) set of h-interactions $H' \subseteq \{\{i, |w|+1\} \mid 1 \leq i \leq |w|, (w[i], a) \in \mathcal{H}\}$. Note that C_2 is also valid. For a conformation C and a finite string $w \in \Sigma^*$, by $\mathcal{E}(C, w)$, we denote the set of all elongations of C by w , that is, $\mathcal{E}(C, w) = \{C' \in \mathcal{C} \mid C \xrightarrow{*}_w C'\}$. For an arity α , let $\mathcal{E}_\alpha(C, w) = \mathcal{E}(C, w) \cap \mathcal{C}_\alpha$.

2.1 Oritatami System

An *oritatami system* (OS) is a 5-tuple $\Xi = (\mathcal{H}, \alpha, d, \sigma, w)$, where \mathcal{H} is a ruleset, α is an *arity*, $d \geq 1$ is a positive integer called the *delay*, σ is an initial valid conformation of arity α called the *seed*, and w is a possibly-infinite string of beads called a *primary structure*.

The delay d , arity- α oritatami system Ξ cotranscriptionally folds its primary structure in the following way. For a string $x \in \Sigma^*$, a conformation C_1 , and an elongation C_2 of C_1 by $x[1]$, we say that Ξ (*cotranscriptionally*) *folds x upon C_1 into C_2* if

$$C_2 \in \operatorname{argmin}_{C' \in \mathcal{E}_\alpha(C_1, x[1])} \min \{ \Delta G(C') \mid C' \in \mathcal{E}_\alpha(C, x[2, k]), 2 \leq k \leq d \}, \quad (1)$$

where $\Delta G(C')$ is an energy function that assigns C' with the negation of the number of h-interactions within C' as energy. Informally speaking, C_2 is a conformation obtained by elongating C_1 by the bead $x[1]$ so as for the beads $x[1], x[2], \dots, x[d]$ to create as many h-interactions as possible. Then we write $C_1 \xrightarrow{\Xi}_x C_2$, and the superscript Ξ is omitted whenever Ξ is clear from context. Through the folding, the first bead of x is *stabilized*. In figures, we conventionally color x —the fragment to be stabilized—in cyan.

Example 1 (Glider). Let us explain how the OS cotranscriptionally folds a motif called the *glider*. Gliders offer a directional linear conformation and have been heavily exploited in the existing studies on OS [3]. Consider a delay-3 OS whose seed is the black conformation in Fig. 3 (a), primary structure is $w = b \bullet ac \bullet bd \bullet c \cdots$, and the ruleset is $\mathcal{H} = \{(a, a), (b, b), (c, c), (d, d)\}$.

By the fragment $w[1, 3] = b \bullet a$, the seed can be elongated in many ways; three of them are shown in Fig. 3 (a). The only bead on the fragment that may form a new h-interaction is a (b is also capable according to \mathcal{H} but no other b is around). In order for the a to get next to the other a , on the seed, the b on the fragment must be located to the east of the last bead of the seed; thus, the b is stabilized there, as shown in Fig. 3 (b). The stabilization transcribes the next bead $w[4] = c$. The sole other c around is on the seed but is too far for the c just transcribed to get adjacent to. Thus, the only way for the fragment $w[2, 4] = \bullet ac$ to form an h-interaction is to put the two a 's next to each other as before, and for that, the \bullet must be located to the southeast of the preceding b as shown in Fig. 3 (c). The next bead to be transcribed, $w[5]$, is inert, and hence, cannot override the previous decision to put the two a 's next to each other. The first six beads have been thus stabilized as shown in Fig. 3 (d), and the glider has thus moved forward by distance 2.

It is easily induced inductively that gliders of arbitrary “flight distance” d can be folded by a delay-3 OS; such long-distance gliders have been used in [3]. Moreover, as suggested in Fig. 3, a constant number of bead types are enough for that (in this example, a, b, c, d, \bullet).

Gliders also provide a medium to propagate 1-bit information at arbitrary distance. The height (up or down) of the first bead determines whether the

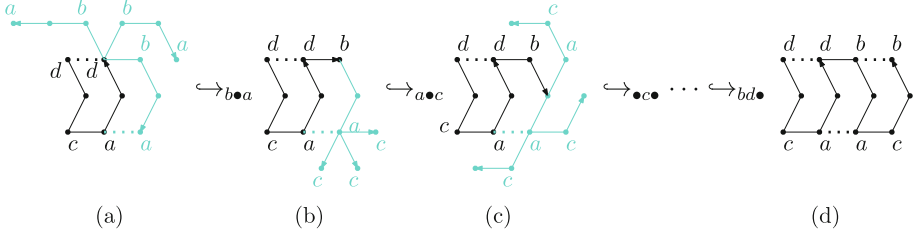


Fig. 3. A glider folded by a delay-3 OS. (a) Three ways to elongate the current conformation by the fragment $b \bullet a$ among many. (b) Three most stable elongations by the fragment $\bullet ac$. (c) Three ways to elongate the current conformation by the fragment $ac \bullet$ among many. (d) The stabilization of $b \bullet ac \bullet b$.

last bead is stabilized up or down after the glider traverses the distance d . For instance, the glider in Fig. 3 launches up and thus its last bead (the second b) also comes up after traveling the distance $d = 2$; being launched down implies being terminated down. This capability has been exploited for an OS to simulate a cyclic tag system for Turing universality [3] and the NOS that we shall propose in Sect. 3 also uses it. \square

Note that cotranscriptional folding, as formulated in (1), considers not only elongations of C by $x[2, d]$ but also those by prefixes of $x[2, d]$, that is, $x[2], x[2, 3], \dots, x[2, d-1]$. This is necessary to fully fold the primary structure till the end or when there is not enough space around the last bead of C to elongate C by the whole $x[2, d]$ (See Fig. 4). Otherwise, under the current energy function, the optimization just ignores those “halfway” elongations because more beads never rise energy. Under other “more realistic” energy functions, halfway elongations would play a more active role in the folding.

The set $\mathcal{F}(\Xi)$ of all conformations foldable by Ξ is now defined recursively as follows: $\sigma \in \mathcal{F}(\Xi)$, and if $C_i \in \mathcal{E}_\alpha(\sigma, w[1, i])$ is in $\mathcal{F}(\Xi)$ and $C_i \xrightarrow{\Xi}_{w[i+1, i+d]} C_{i+1}$, then $C_{i+1} \in \mathcal{F}(\Xi)$. A finite conformation $C_i \in \mathcal{E}_\alpha(\sigma, w[1, i])$ foldable by Ξ is *terminal* if one of the following conditions holds:

1. the primary structure w is finite and $i = |w|$;
2. either w is infinite or $i < |w|$, and there exists no conformation C_{i+1} such that $C_i \xrightarrow{\Xi}_{w[i+1, i+\delta]} C_{i+1}$.

Note that not only the conformation in Fig. 4 (d) but also the conformation in Fig. 4 (f) is terminal by the second condition of the terminal conformation. By $\mathcal{F}_\square(\Xi)$, we denote the set of all terminal conformations foldable by Ξ .

The OS Ξ is *deterministic* if any foldable conformation $C_i \in \mathcal{E}_\alpha(\sigma, w[1, i])$ is either terminal or admits a unique conformation C_{i+1} such that $C_i \xrightarrow{\Xi}_{w[i+1, i+\delta]} C_{i+1}$, that is, every bead is stabilized uniquely. For example, the system in Fig. 4 is nondeterministic. Note that nondeterministic systems fold into multiple terminal conformations as suggested in Fig. 4. On the other hand, deterministic

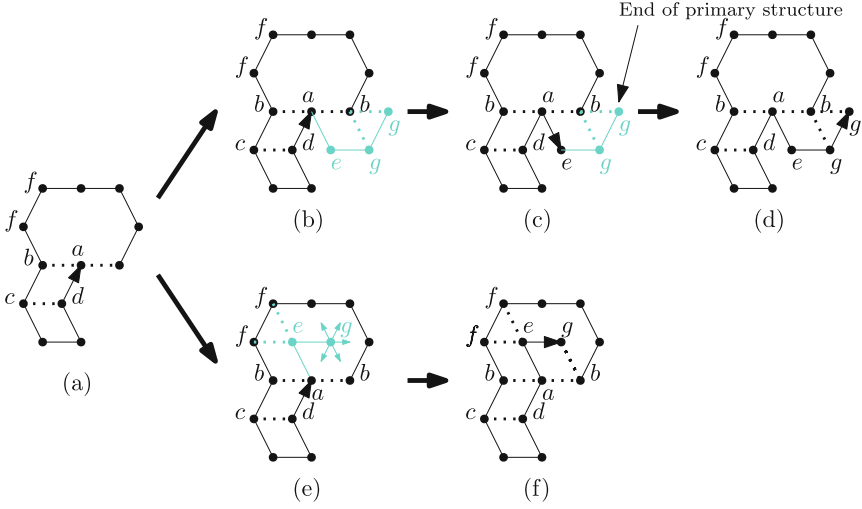


Fig. 4. The two cases in which cotranscriptional folding considers “halfway” elongations. (b) and (e) show two most stable elongations of the current conformation (a); the one in (e) is a halfway elongation. (b) to (d) show the case when folding is almost over, and (e) and (f) show the case when there is not enough space. The conformations in (d) and (f) are both terminal, though the one in (f) is “shorter.”

systems fold into exactly one terminal conformation by definition. Thus, an oritatami system is deterministic if and only if the system folds into one terminal conformation. The property of being deterministic is decidable in linear time. Indeed, it suffices to run an OS and check whether it encounters nondeterminism or not.

Example 2 (Assignor). Let us exhibit here how nondeterminism is used in the OS that we shall propose in Sect. 3, or more particularly, in its module called the *assignor*. The OS is of delay 3, with a ruleset including $(10^{\text{eb}}, 3^{\text{a}}), (10^{\text{eb}}, 9^{\text{a}}), (12^{\text{eb}}, 3^{\text{eb}}), (12^{\text{eb}}, 9^{\text{a}}), (12^{\text{eb}}, 4^{\text{a}})$. The OS folds the assignor uniquely as shown in Fig. 5 (a), up to its fourth last bead. The last three beads of the assignor are $10^{\text{eb}}, 11^{\text{eb}}$ and 12^{eb} .

The fragment $10^{\text{eb}}\text{--}11^{\text{eb}}\text{--}12^{\text{eb}}$ can be fold in two ways equally stably with three h-interactions as shown in Fig. 5 (b-1) and (b-2), and no more no matter how the fragment is folded. Hence, the bead 10^{eb} is stabilized in two ways as shown in Fig. 5 (c-1) and (c-2). The remaining beads 11^{eb} and 12^{eb} are stabilized uniquely one after another as shown in Fig. 5 (d-1) and (d-2). The assignor nondeterministically stabilizes the last bead 12^{eb} up or down. In our NOS, this random assignment of 1-bit information is propagated by gliders in the way mentioned in Example 1. \square

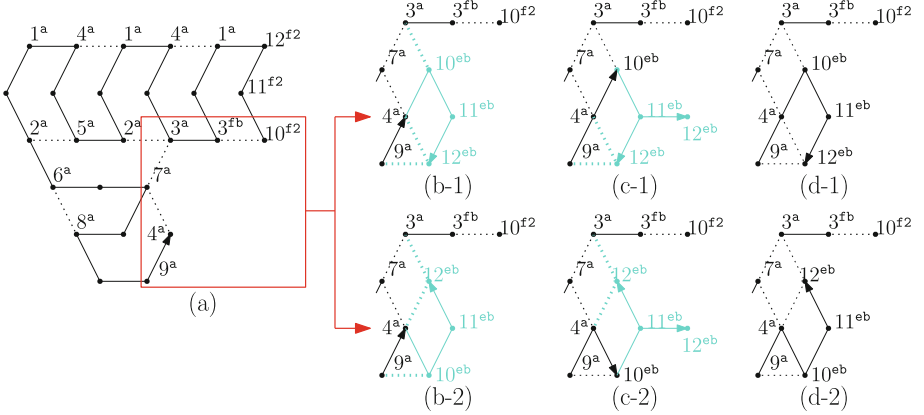


Fig. 5. An assignor folded by a delay 3 OS. While stabilizing the bead 10^{eb} , two elongations equally give three interactions and the bead nondeterministically stabilizes at two different points. (a) The conformation up to the fourth last bead. (b) Two ways to elongate the current conformation by the fragment 10^{eb} - 11^{eb} - 12^{eb} . (c) Ways to elongate the current conformation by the fragment 11^{eb} - 12^{eb} . (d) Two final conformations.

We say that an OS is *seedless* if its seed is (λ, \emptyset) . A seedless OS can start folding at any point of the lattice. If a conformation C is foldable, then any of its congruence, that is, a conformation obtained by applying a combination of translation, rotation, and reflection to C is also foldable. Therefore, fixing the first bead to the origin of the lattice and the second bead to one specific neighbor of the origin does not cause any loss of generality. In this sense, we can regard an OS with a seed of at most 2 beads seedless. Furthermore, a seed of 3 beads can make an OS seedless. Imagine an OS whose seed consists of 2 beads. if an elongation of the seed by the first bead is foldable, then its reflection across the seed, which is just a line segment, is also foldable. Hence, if the first bead is stabilized uniquely, then it can be rather considered as a part of the seed. That is the case for the OS we shall propose in Sect. 3. In this sense, the OS we propose is seedless.

We formally define the equivalence of two oritatami systems.

Definition 1. Two oritatami systems Ξ_1 and Ξ_2 are equivalent if $\mathcal{F}_{\square}(\Xi_1) = \mathcal{F}_{\square}(\Xi_2)$, namely, if there is no terminal conformation that one folds into but the other does not.

Then, we define oritatami system equivalence test.

Definition 2. Given two oritatami systems Ξ_1 and Ξ_2 , oritatami system equivalence test (OSEQ) is to determine whether or not they are equivalent.

Since it takes linear time to simulate a deterministic OS, we establish the following theorem.

Theorem 1. *For two deterministic oritatami systems, the OSEQ problem can be solved in linear time.*

On the other hand, the problem for NOSs turns out to be hard, and we shall prove the following theorem in Sect. 3.

Theorem 2. *The OSEQ problem is co-NP complete, even if the two input NOSs differ only in ruleset, and their rulesets $\mathcal{H}_1, \mathcal{H}_2$ satisfy $\mathcal{H}_1 = \mathcal{H}_2 \cup \{(a, b)\}$ for some rule (a, b) .*

3 Seedless NOS as a DNF Verifier

We propose a seedless NOS that evaluates a given DNF formula, i.e., a Boolean formula in the disjunctive normal form, and then make use of it to prove the coNP-hardness of deciding if two given NOSs are equivalent even under a severe constraint.

A DNF formula ϕ is written as $\bigvee_{1 \leq i \leq n} C_i$ for some clauses C_1, \dots, C_n that is a logical AND (\wedge) of some of the Boolean variables v_1, \dots, v_m and their negations. The *DNF tautology problem* asks if a given DNF formula is evaluated to TRUE on all possible assignments. The problem is coNP-hard, since it can be polynomially reduced from the dual problem of the satisfiability problem, which is NP-complete [6].

Algorithm 1. Evaluate a DNF formula with m variables and n clauses formula on a randomly chosen assignment

```

1 for  $k = 1$  to  $n$  do  $c[k] \leftarrow *$ ;
2 for  $i = 1$  to  $m$  do
3   Randomly assign TRUE or FALSE into  $v_i$ ;
4   for  $k = 1$  to  $n$  do
5     if The  $k$ -th clause involves  $v_i$  and  $v_i = \text{FALSE}$  then  $c[k] \leftarrow \text{U}$ ;
6     else if The  $k$ -th clause involves  $\neg v_i$  and  $v_i = \text{TRUE}$  then  $c[k] \leftarrow \text{U}$ ;
7 for  $k = 1$  to  $n$  do
8   if  $c[k] = *$  then return Satisfied;
9 return Unsatisfied

```

Algorithm 1 evaluates the DNF formula ϕ on a random assignment. For the ease of explanation, we assume m is even (otherwise we just assume one more imaginary variable that occurs nowhere). The seedless NOS Ξ_τ evaluates ϕ using this algorithm. Both its delay and arity are set to 3 (in fact, the arity can be set to any value larger). We conventionally use the term *context* to denote beads and interactions around the current bead that we consider during the stabilization.

Its primary structure is of the form $w_s w_1 w_2 \cdots w_m w_v$. We call the factors $w_s, w_1, \dots, w_m, w_v$ *modules*. The NOS is to fold the primary structure in

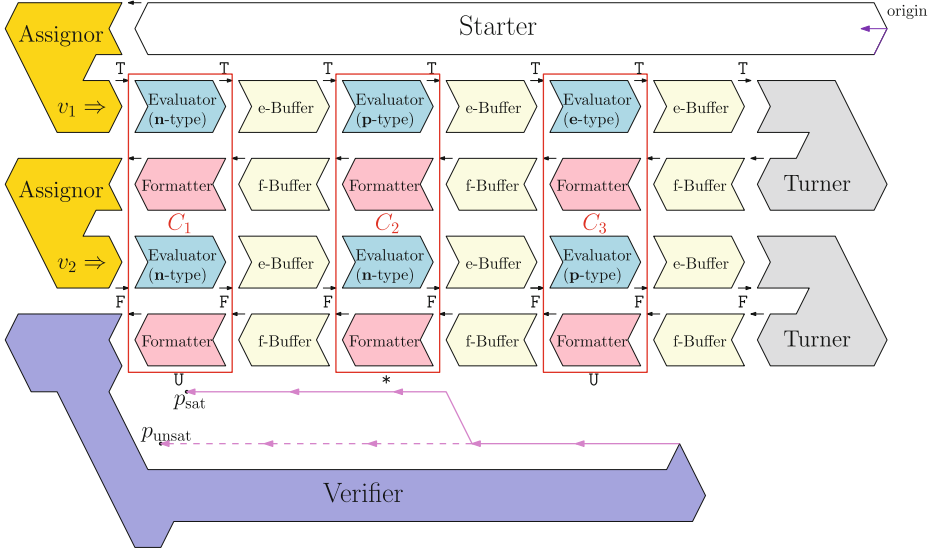


Fig. 6. An overview of the NOS that evaluates a given DNF formula $\phi = (\neg v_1 \wedge \neg v_2) \vee (v_1 \wedge \neg v_2) \vee (v_2)$, with two variables v_1, v_2 and three clauses C_1, C_2, C_3 , on the assignment $(v_1, v_2) = (T, F)$ that it chooses nondeterministically. The folding starts from the purple arrow in the starter. In the last module called the verifier, the conformation folds to p_{sat} since the formula is satisfied. Pink dashed lines represent an alternate conformation that stops at p_{unsat} when the formula is not satisfied. (Color figure online)

a zigzag manner as outlined in Fig. 6. The first module w_s named the *starter* folds into the glider as shown in Fig. 7 and offers a linear scaffold of width $O(n)$, which serves as a “seed” for the succeeding modules. This module admits no other conformation, and it is almost straightforward to design a module that folds uniquely by hardcoding the target conformation into the primary structure and ruleset. In fact, all the rules used in the glider in Fig. 7 are sufficient (and necessary) for the primary structure of w_s to fold into the glider. Note that we use superscripts to indicate sets of bead types used for different modules, i.e., $\mathbf{f}2$ is used for formatters and \mathbf{s} is used for starters. Being thus folded, the starter exposes below n copies of the sequence of bead types $10^{\mathbf{f}2}\text{-}9^{\mathbf{f}2}\text{-}8^{\mathbf{f}2}\text{-}7^{\mathbf{f}2}$ at a fixed interval (every 8 points), which shall be translated by succeeding modules as the corresponding clause being satisfiable (denoted by $*$). This corresponds to the initialization of the array c in line 1 of Algorithm 1.

The next module w_1 consists of submodules. The first submodule is the assignor explained in Example 2. It nondeterministically stabilizes its last bead up or down, and the OS interprets up as TRUE being assigned to v_1 and down as false being assigned to v_1 (See Fig. 6, where TRUE is assigned to v_1 , for instance.). The assignor is succeeded by submodules called evaluators and buffers, which occur alternately. The buffer is just a glider. There are n evaluators in w_1 ; one for each of the n clauses. The k -th evaluator, for C_k , takes the value (T/F) assigned

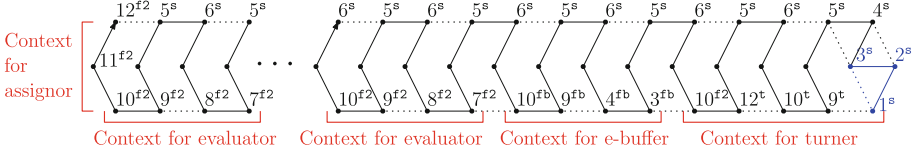


Fig. 7. The linear scaffold conformation into which the starter w_s deterministically folds. Three blue beads indicate the seed. (Color figure online)

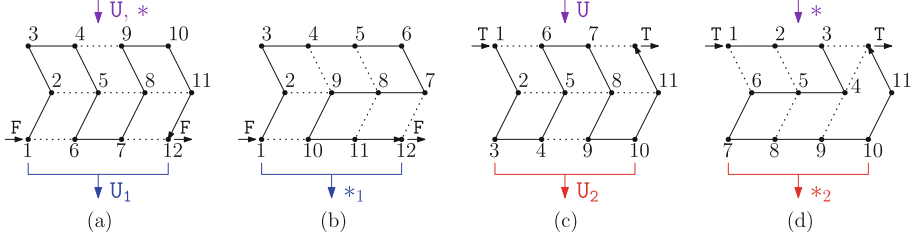


Fig. 8. The possible four conformations of evaluators and formatters. We denote each bead in the conformation by its order, from 1 to 12. In order to propagate the Boolean value, all of the conformations start and end at the same height. Arrows at top and bottom denote respectively the previous and updated evaluations of the corresponding clause, though they are in different formats. The purple arrows from top indicate which conformations the p-evaluator takes, depending on whether F or T is input from the left; hence, the p-evaluator never takes the conformation (b).

to v_1 from the left and the evaluation from the top as inputs, and outputs the value of v_1 to the right faithfully and the updated evaluation of whether the clause C_k is still satisfiable or not according to the value of v_1 to the bottom. Therefore, four distinct conformations are sufficient for evaluators, and we chose those in Fig. 8. There are three possible ways to update, depending on if C_k includes v_1 , or its negation, or none of them. Hence, the OS employs three types of evaluators: **p**, **n**, and **e**. For example, as shown in Fig. 8, the **p**-evaluator folds into the conformation (a) no matter how the clause has been evaluated so far if $v_1 = F$, while it folds into (c) or (d) depending on the evaluation if $v_1 = T$. Hence, the **p**-evaluator never folds into (b). The clauses C_1 , C_2 and C_3 of the formula evaluated in Fig. 6 include $\neg v_1$, v_1 , and none of them, respectively, and hence the first three evaluators are of type **n**, **p**, and **e**, respectively. Note that evaluators output each evaluation in two distinct formats (U_1, U_2 for unsatisfied, $*_1, *_2$ for satisfiable). They will be reformatted by a submodule called the *formatter* in the succeeding zag, as 10-9-8-7 for $*$ and 10-9-4-3 for U . Analogously, for any $2 \leq i \leq m$, the module w_i first assigns T or F randomly to v_i and update the evaluation of clauses provided by the previous zag (of w_{i-1}). As such, the folding of w_i corresponds to the i -th iteration of the for-loop at line 2–6 of Algorithm 1.

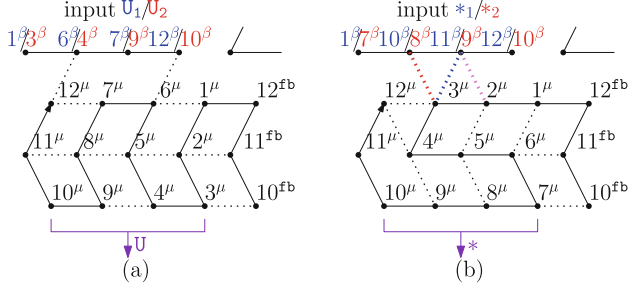


Fig. 9. (a) The glider-like conformation that a formatter takes on the input U_1/U_2 . (b) The alternative conformation of a formatter on the input $*_1/*_2$. Blue and red interactions are for the corresponding colored bead only. (Color figure online)

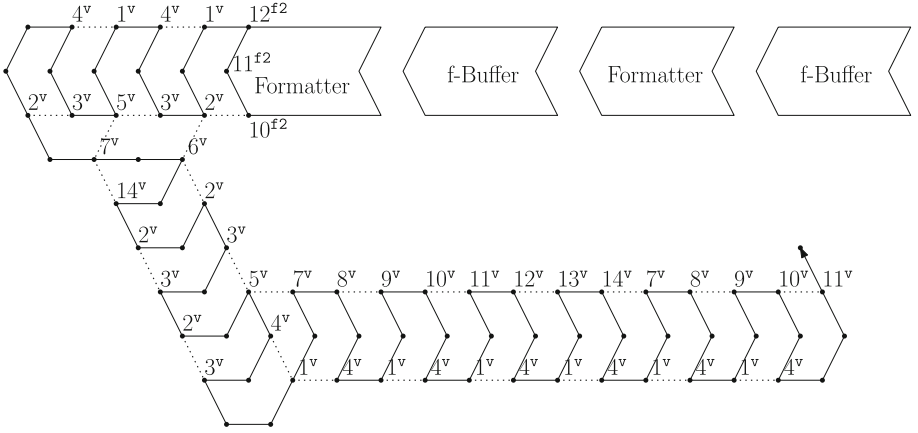


Fig. 10. The first part of the verifier uniquely folds thus and provides a scaffold on which the rest of the verifier serves its role to verify the clauses.

The final module w_v verifies if there is a clause still evaluated to be satisfiable, or equivalently, if the last zag (of w_m) exposes below the sequence $10^{f2}-9^{f2}-8^{f2}-7^{f2}$, corresponding to the termination process at line 7–9 (Note that we use superscripts to denote different sets of beads for different types of submodules.). The module is named the *verifier* after this role. The verifier first folds into the conformation shown in Fig. 10. Like the starter, this conformation is also hardcoded; all the rules used are sufficient for the conformation to be folded as shown in Fig. 10. The rest of the verifier is to thread its way from right to left through the recess between the last zag and the floor just made by the first part of the verifier, as shown in Fig. 6. More precisely, it is stretched straight along the floor and once it “detects” a satisfiable clause, or the encoding of $*$, i.e., $10^{f2}-9^{f2}-8^{f2}-7^{f2}$, it is pulled up and starts being stretched straight along the zag above. The detection is done by the segment $15^v-16^v-17^v$, which we name the *probe*. The probe forms two

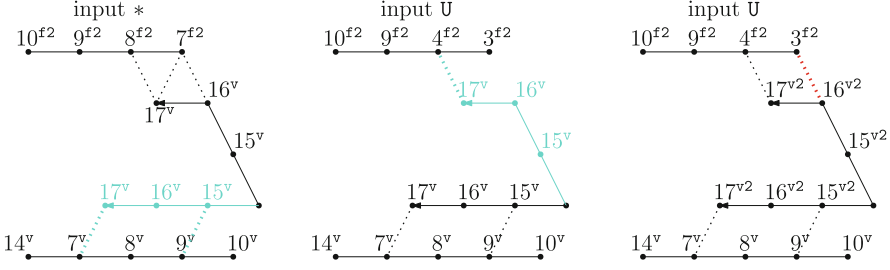


Fig. 11. Detection of satisfiable clauses by the probe segment $15^v-16^v-17^v$. (Left) The sequence $8^{f2}-7^{f2}$, a part of the encoding of $*$, pulls the probe strongly by 3 h-interactions, one more than the number of h-interactions between the probe and the floor. (Middle) The sequence $4^{f2}-3^{f2}$, a part of the encoding of U , cannot pull the probe as strongly as the floor does. (Right) The rule $(3^{f2}, 16^{v2})$ added to Ξ_{τ_1} allows the probe for C_1 to be pulled upward as well as leftward equally strongly when C_1 is evaluated to be unsatisfied.

h-interactions with the floor, but more h-interactions with the encoding of $*$ due to the rules $(17^v, 8^{f2})$, $(17^v, 7^{f2})$, and $(16^v, 7^{f2})$ (see Fig. 11 (Left)). In contrast, the probe can form only 1 h-interaction with the encoding of U , as shown in Fig. 11 (Middle) due to lack of rules. As a result, the last bead of the probe is stabilized close to the zag above (at the point p_{sat} in Fig. 6) if and only if ϕ is satisfied by the chosen assignment. The last probe is of distinct bead types as $15^{v2}-16^{v2}-17^{v2}$ for the sake of proving hardness of OSEQ later.

It now suffices to explain the module w_i for the i -th zigzag ($1 \leq i \leq m$) into detail. Its primary structure is made up as $w_a u_{i,1} \triangleright u_{i,2} \triangleright \cdots \triangleright u_{i,n} \triangleright w_t \triangleleft f_{i,n} \triangleleft \cdots \triangleleft f_{i,2} \triangleleft f_{i,1}$, where w_a is the assignor, w_t is a submodule called the *turner*, and $u_{i,k}$ and $f_{i,k}$ are the evaluator and formatter for the k -th clause, respectively, and triangles (\triangleright and \triangleleft) indicate sequences of 12 beads called *e-buffers* and *f-buffers* respectively. Buffers keep a sufficient distance between the consecutive submodules horizontally so as for them not to interact with each other. As shown in Fig. 6, buffers in a consecutive zig and zag may get adjacent to each other vertically. Should they involve a common bead type, an inter-buffer interaction could prevent them from folding into a glider. Therefore, e-buffers and f-buffers use pairwise-distinct sets of bead types. The turner w_t is a hardcoded submodule. Its two possible conformations, shown in Fig. 12, let multiple possible paths arisen from the nondeterministic value assignment to v_i converge into one path (at this point, the system is allowed to “forget” the value).

Evaluator and formatter. The k -th evaluator $u_{i,k}$ and the k -th formatter $f_{i,k}$ in the i -th zig and zag cooperatively update the evaluation of whether the k -th clause is still satisfiable or already unsatisfied. The role of formatters is auxiliary: as we already explained, the output of evaluators ($*/U$) is encoded (as a sequence of beads exposed below) redundantly, and formatters reformat them and ensure that evaluators in the next zig suffice to be capable of reading one sequence of

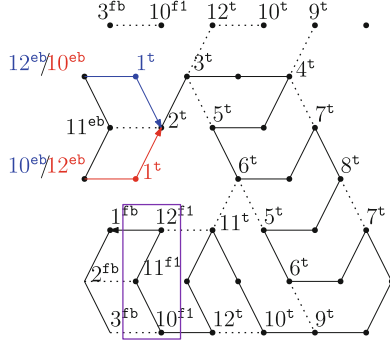


Fig. 12. The two possible conformations of turners, which converge multiple possible paths due to nondeterministic value assignment into one path. Purple box shows the context for the next f-buffer. (Color figure online)

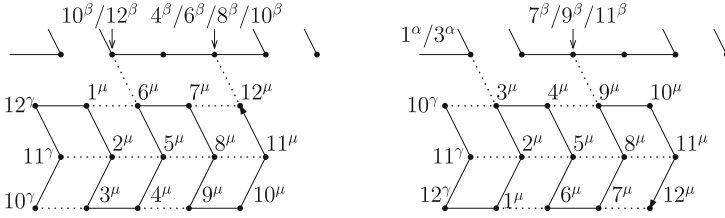


Fig. 13. The basic module is a modification of the glider and folds its primary structure $1^\mu 2^\mu \dots 12^\mu$ into these two conformations deterministically depending on whether the first bead 1^μ is up or down.

beads for $*$ and one for \mathbf{U} . The glider-based foundation, which we will explain shortly, is modified in such a way that the evaluator and formatter inherit the information transfer capability, which enables the n evaluators for the i -th zig, that is, $u_{i,1}, u_{i,2}, \dots, u_{i,n}$, to transfer the value randomly assigned to v_i one after another.

The basic module is to fold its primary structure $1^\mu 2^\mu \dots 11^\mu 12^\mu$ (we use Greek letters to represent a set of different bead types) into one of the two gliders shown in Fig. 13 deterministically depending on the two possible contexts (in another context, it could admit another conformation, but in the proposed NOS, the such context is never encountered). It is implemented using the following ruleset R : $R = \{(2^\mu, 11^\gamma), (3^\mu, 1^\alpha), (3^\mu, 3^\alpha), (3^\mu, 10^\gamma), (5^\mu, 2^\mu), (6^\mu, 11^\gamma), (6^\mu, 1^\mu), (6^\mu, 10^\beta), (6^\mu, 12^\beta), (7^\mu, 10^\gamma), (8^\mu, 5^\mu), (9^\mu, 2^\mu), (9^\mu, 4^\mu), (9^\mu, 7^\beta), (9^\mu, 9^\beta), (9^\mu, 11^\beta), (10^\mu, 1^\mu), (11^\mu, 8^\mu), (12^\mu, 4^\beta), (12^\mu, 6^\beta), (12^\mu, 8^\beta), (12^\mu, 10^\beta), (12^\mu, 3^\mu), (12^\mu, 4^\mu), (12^\mu, 7^\mu)\}$ where $\{(\alpha, \beta, \gamma, \mu)\} = \{(\mathbf{fb}, \mathbf{f2}, \mathbf{eb}, \{\mathbf{p1}, \mathbf{n1}, \mathbf{e1}\}), (\mathbf{eb}, \{\mathbf{p1}, \mathbf{n1}, \mathbf{e1}\}, \mathbf{fb}, \mathbf{f1}), (\mathbf{fb}, \mathbf{f1}, \mathbf{eb}, \{\mathbf{p2}, \mathbf{n2}, \mathbf{e2}\}), (\mathbf{eb}, \{\mathbf{p2}, \mathbf{n2}, \mathbf{e2}\}, \mathbf{fb}, \mathbf{f2})\}$.

The evaluator and formatter are derived by “equipping” the basic module with the capability of “reading” the output of the module above; formally speaking, we add some rules to the basic ruleset R that attract some factor of the primary structure (called *input reader*) towards the output so that the resulting module favors another conformation over the glider. Here, one design criterion should be noted: we designed the NOS in such a manner that a module (evaluator/formatter) taking a glider represents the evaluation U . In Fig. 8, the two non-glider conformations are illustrated. Note that these conformations properly propagate the value (F/T) randomly assigned to v_i by the assignor.

Let us focus attention to the evaluator $u_{i,k}$, which evaluates the k -th clause C_k according to the value randomly assigned to v_i and the evaluation made so far by the previous evaluators $u_{1,k}, u_{2,k}, \dots, u_{i-1,k}$. There are three possibilities to be taken into account depending on whether C_k contains the positive literal v_i , its negation $\neg v_i$, or none of them. That is, three types of evaluators (**p**, **n**, and **e**) are needed. The **p**-type evaluator is supposed to fold into the glider (U) no matter what the previous evaluation is if $v_i = F$, but be capable of taking both the glider and a non-glider conformation so as to propagate the previous evaluation as it is when $v_i = T$, corresponding to line 5. The **n**-type evaluator is supposed to behave analogously but the roles of **F** and **T** are flipped, corresponding to line 6. The **e**-type evaluator should propagate the previous evaluation as it is no matter which value is assigned to v_i .

The evaluator $u_{i,k}$ is sandwiched from above and below by two formatters. Since the evaluator interacts with both of them, a bead type common in these formatters would cause misfolding. Therefore, the NOS implements evaluators in consecutive zigs using two pairwise-distinct sets of bead types, even if they are of the identical type. This results in, for instance, two sets of bead types $\{p1, p2\}$ for the type-**p** evaluators. Similarly, the NOS uses two distinct sets of bead types for each type of evaluators and the formatter, which we distinguish with numbers.

We propose the following two sets R_T, R_F of extra rules, which enable the module to read the output when $v_i = T$ and when $v_i = F$, respectively: $R_T = \{(2^\mu, 9^\beta), (3^\mu, 8^\beta), (4^\mu, 7^\beta), (5^\mu, 10^{\text{fb}})\}$ where $\{(\beta, \mu)\} = \{(f2, p1), (f2, e1), (f1, p2), (f1, e2)\}$, and $R_F = \{(5^\mu, 8^\beta), (6^\mu, 7^\beta), (7^\mu, 10^{\text{fb}})\}$ where $\{(\beta, \mu)\} = \{(f2, n1), (f2, e1), (f1, n2), (f1, e2)\}$.

Rules in R_T convert the module with sets of bead types $\{p1, p2\}$ into the type-**p** evaluator, while rules in R_F convert the module with $\{n1, n2\}$ into the type-**n** evaluator. Note that these extra rulesets do not interfere with each other, and adding both of them converts the module with $\{e1, e2\}$ into the type-**e** evaluator. Figure 14 shows foldings with newly added rules.

The outputs of an evaluator are redundant: $*$ is encoded as $1^\mu 10^\mu 11^\mu 12^\mu$ or $7^\mu 8^\mu 9^\mu 10^\mu$ whereas U is encoded as $1^\mu 6^\mu 7^\mu 12^\mu$ or $3^\mu 4^\mu 9^\mu 10^\mu$. The redundant output is reformatted by formatters in the i -th zag so as for an input to the evaluators in the next $(i+1)$ -th zig to be encoded in a unique format. Unlike the evaluator, formatters do not have to propagate 1-bit information horizontally. The conformation of the turner fixes the first bead of the first formatter $f_{i,n}$ in the i -th zag up. The following ruleset R_{format} converts the module with sets

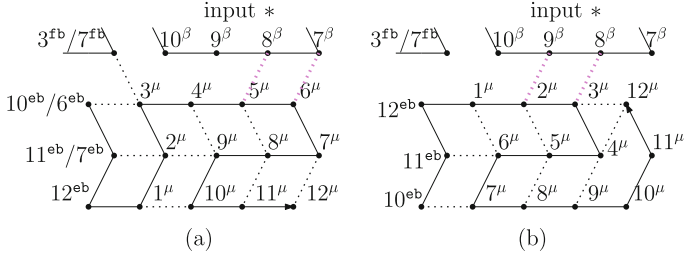


Fig. 14. (a) Folding of **n**-evaluators and **e**-evaluators when $v_i = F$ and the input is $*$. Newly added rules are colored in pink. (b) Folding of **p**-evaluators and **e**-evaluators when $v_i = T$ and the input is $*$. (Color figure online)

of bead types $\{f1, f2\}$ into the formatter, which takes the conformation (c) in Fig. 8 if the output of the evaluator above is **U** or the conformation (d) if the output is $*$: $R_{\text{format}} = \{(2^\mu, 9^\beta), (2^\mu, 11^\beta), (3^\mu, 8^\beta), (3^\mu, 11^\beta), (4^\mu, 8^\beta), (4^\mu, 1^\beta)\}$ where $\{(\beta, \mu)\} = \{(p1, f1), (n1, f1), (e1, f1), (p2, f2), (n2, f2), (e2, f2)\}$. Figure 9 shows foldings with newly added rules.

We establish the following theorem for Ξ_τ .

Theorem 3. *Let Ξ_τ be the seedless NOS generated from a DNF formula ϕ . Then, ϕ is tautology if and only if there is no conformation of Ξ_τ that reaches the point p_{unsat} .*

Once we establish a robust design of Ξ_τ , we now prove the hardness of OS equivalence test by variations of Ξ_τ . Note that the size of the ruleset in Ξ_τ is constant, and it takes $O(nm)$ time to construct the primary structure of Ξ_τ from a DNF formula with n clauses and m variables. Thus, we can construct Ξ_τ that represents the given formula in $O(nm)$ time. The OSEQ problem admits as a polynomial no-certificate a conformation that one of the two given OSs can fold but the other cannot. It hence belongs to coNP. The coNP-hardness of OSEQ is proved by reduction from DNF tautology. We derive Ξ_{τ_1} from Ξ_τ by adding one additional rule $(16^{v2}, 3^{f1})$, which makes the verification of Ξ_{τ_1} nondeterministic when ϕ is not tautology as illustrated in Fig. 11. Thus, ϕ is tautology if and only if Ξ_τ and Ξ_{τ_1} are equivalent. This proves that OSEQ is coNP-complete even if two OSs are seedless and identical except for their rulesets $\mathcal{H}_1, \mathcal{H}_2$ such that $\mathcal{H}_1 \subseteq \mathcal{H}_2$ and $|\mathcal{H}_2| - |\mathcal{H}_1| = 1$.

4 Conclusions

We have designed a seedless NOS that solves the DNF tautology problem, and demonstrated the hardness of testing the equivalence of two OSs using the designed NOS. Since this is the first attempt to exploit nondeterminism and seedlessness in the design of OS, our future work includes applying nondeterminism and seedlessness to solve other problems. It is an open problem whether

we can design an equivalent seedless OS from a given OS or not. Also note that we map an instance of DNF formulas to an NOS that is unique to that input. This notion is called semi-uniformity [8] compared to circuit uniformity [1], where we provide a computing device solely according to the length of the input. Introducing circuit uniformity to the design of OS is another open problem.

Acknowledgments. We would like to thank the anonymous reviewers for the careful reading of the paper and many valuable suggestions.

Kim was supported by NRF Grant funded by the Korean Government (NRF-2013-Global Ph.D. Fellowship Program). The work of S. S. was supported in part by JST Program to Disseminate Tenure Tracking System, MEXT, Japan, No. 6F36, and by JSPS Grant-in-Aid for Research Activity Start-up No. 15H06212 and for Young Scientists (A) No. 16H05854.

References

1. Borodin, A.: On relating time and space to size and depth. *SIAM J. Comput.* **6**(4), 733–744 (1977)
2. Frieda, K.L., Block, S.M.: Direct observations of cotranscriptional folding in an adenine riboswitch. *Science* **338**(6105), 397–400 (2012)
3. Geary, C., Meunier, P., Schabanel, N., Seki, S.: Efficient universal computation by greedy molecular folding (2015). CoRR, abs/1508.00510
4. Geary, C., Meunier, P., Schabanel, N., Seki, S.: Programming biomolecules that fold greedily during transcription. In: *Proceedings of the 41st International Symposium on Mathematical Foundations of Computer Science* (2016, to appear)
5. Geary, C., Rothmund, P.W.K., Andersen, E.S.: A single-stranded architecture for cotranscriptional folding of RNA nanostructures. *Science* **345**, 799–804 (2014)
6. Kleinberg, J., Tardos, É.: *Algorithm Design*. Addison-Wesley, Reading (2011)
7. Lai, D., Proctor, J.R., Meyer, I.M.: On the importance of cotranscriptional RNA structure formation. *RNA* **19**, 1461–1473 (2013)
8. Pérez-Jiménez, M.J., Romero-Jiménez, A., Sancho-Caparrini, F.: Complexity classes in models of cellular computing with membranes. *Nat. Comput.* **2**(3), 265–285 (2003)
9. Rivas, E., Eddy, S.R.: A dynamic programming algorithm for RNA structure prediction including pseudoknots. *J. Mol. Biol.* **285**(5), 2053–2068 (1999)
10. Xayaphoummine, A., Bucher, T., Isambert, H.: Kinefold web server for RNA/DNA folding path and structure prediction including pseudoknots and knots. *Nucleic Acids Res.* **33**, W605–W610 (2005)
11. Zuker, M.: Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Res.* **31**(13), 3406–3415 (2003)
12. Zuker, M., Stiegler, P.: Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Res.* **9**(1), 133–148 (1981)

DNA Computing and Molecular Programming
22nd International Conference, DNA 22, Munich,
Germany, September 4-8, 2016. Proceedings
Rondelez, Y.; Woods, D. (Eds.)
2016, XXV, 183 p. 59 illus., Softcover
ISBN: 978-3-319-43993-8