平成 28 年度 情報・通信工学科 コンピュータ・サイエンスコース 卒業研究

Paperfolding sequence を出力するオートマトンの Cotranscriptional folding による実装

情報・通信工学科　学籍番号:1311178　関新之助 研究室　増田 優生

概要

本論文では Oritatami System を用いた「ヘイウェイドラゴン (Heighway Dragon)」の折りたたみに挑戦する。Oritatami System とは、RNA が鋳型 DNA から転写と同時に折りたたまれる現象である「共転写性フォールディング (cotranscriptional folding)」の数理モデルである。Heighway Dragon は二進数で表せるフラクタルの一種であり、この二進数はオートマトンの出力によって得られることが知られている。本稿ではこのオートマトンを Oritatami System に実装する。

1　はじめに

ヌクレオチドやアミノ酸が生物学的な機能を発揮するためには、複雑な立体構造を持つ必要がある。この構造を分析し、理解することが、今日の生物学における課題のひとつといえるだろう。

DNA や RNA はヌクレオチドの鎖として情報を保持している。このヌクレオチドは 4 種類からなり、それぞれ (A, C, G, T)、(A, C, G, U) と表現される。これらには A-T (U) 間、C-G 間で水素結合を持つという特徴があることが知られている [1]。

RNA の鎖は、DNA の情報を読み取ることによってまず生成される。この際に機能するのが RNA ポリメラーゼという酵素である。この酵素が鋳型 DNA に付着し、DNA 鎖の 5' 末端から 3' 末端に向けて 4 つのヌクレオチドからなる情報を走査する。そして酵素が付着している DNA から解離するまで、DNA 鎖に相補的な RNA 鎖を作り出す。この現象を「転写」と呼ぶ。

なお、ヌクレオチドが転写される速度は、すでに転写された RNA が最適な構造に折りたたまれるよりも遅い [2]。つまり、転写と折りたたみは同時に進行しているのである。この現象は「共転写性フォールディング (cotranscriptional folding)」と呼ばれる [3]。

また、RNA が折りたたまれる際に生じる動力学は、RNA の最終構造に影響を及ぼすことがすでに証明されている [4]。[5] では、この現象を利用して実際に図 1 のような構造を人為的に作成することにも成功している。

この構造の特徴は、局所的な最適化が行われているという点にある。これまでの研究では、高分子の鎖が全て合成された状態から最小エネルギー構造を問う議論がなされてきた。それと比較すると、この共転写性フォールディングは新しい可能性を持った研究であるといえるだろう。

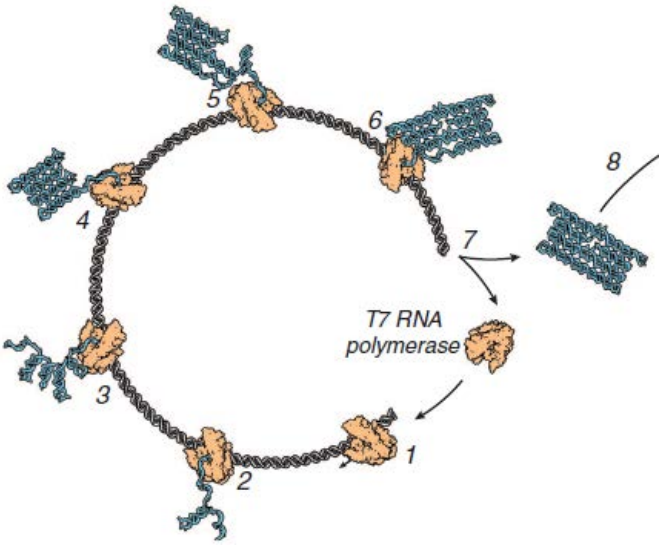
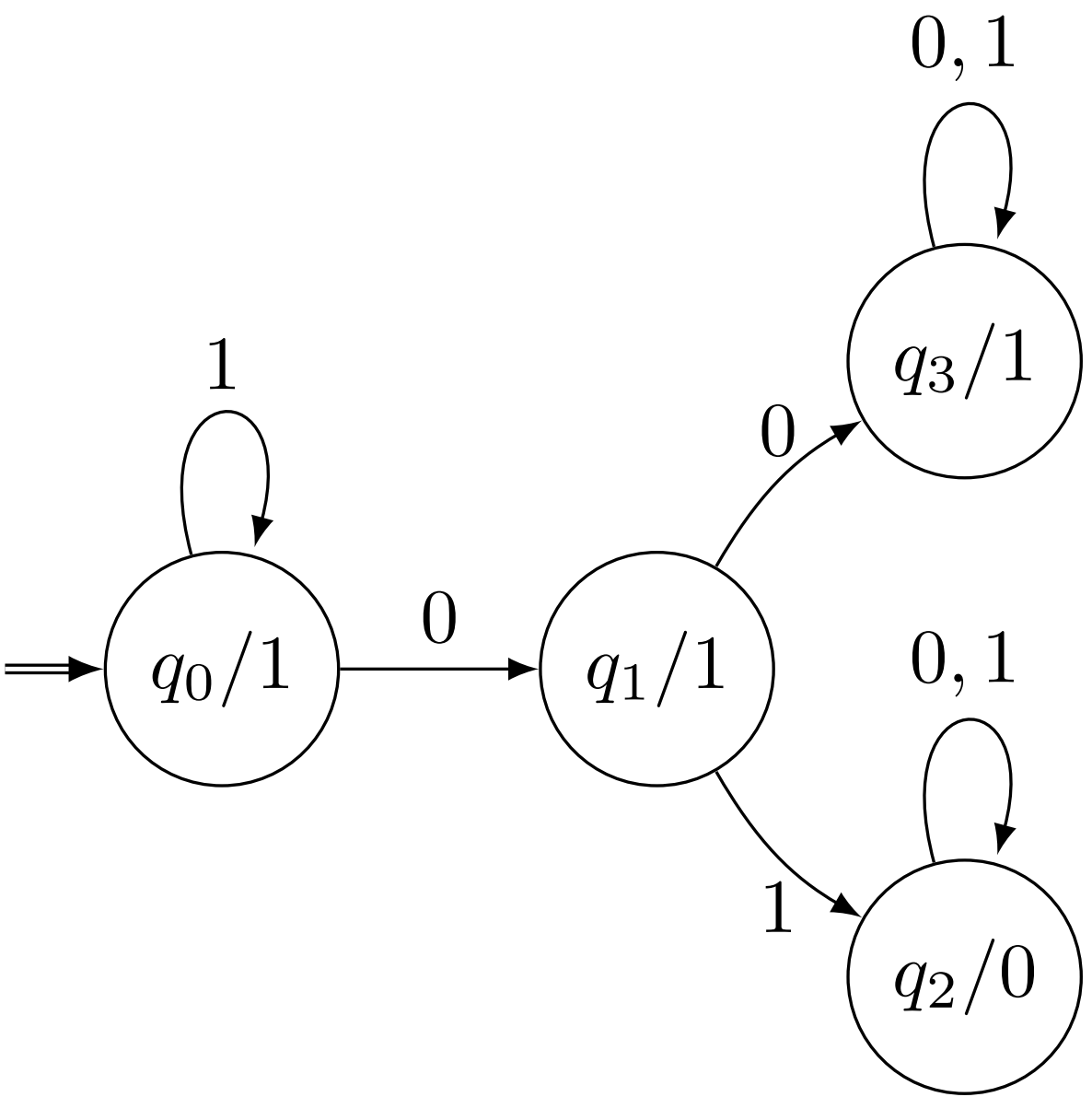


図 1: RNA 鎖が共転写フォールディングによって折りたたまれる様子 [5]

これを踏まえた上で、本論文では Geary らによって提案された Oritatami System を用いた [6]。この数理モデルでは、RNA の鎖を (5'→3' のように) 方向付けされた抽象高分子の鎖としてモデル化する。この抽象高分子のことを「ビーズ」と呼ぶ。Oritatami System は何らかの格子上で折りたたまれた有向パスと、パス上で隣接するビーズの間で形成された水素結合の集合の 2 つ組を RNA の構造としてモデル化する。完全に出来上


0, 1
1
$q_3/1$
0
$q_0/1$
0
$q_1/1$
0, 1
1
$q_2/0$

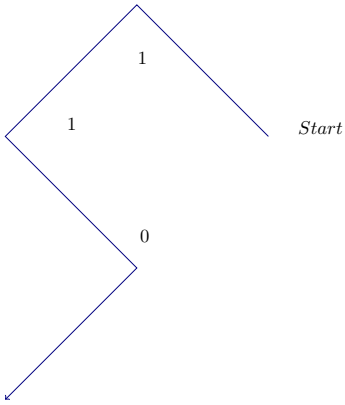


図 3: Paperfolding Sequence: Two Folds [7]

Heighway Dragon を設計するために、本論文ではこの出力付きオートマトンを Oritatami System に実装する。Heighway Dragon の概要図を図 5 に示した。オートマトンの入力はバイナリーカウンターの値となっている。オートマトンの出力した値により、方向転換を行うことで Heighway Dragon へと折りたたむ。バイナリーカウンターの設計には [8] を参考にした。ただし、[8] では異なるダイナミクスの Oritatami System を使用している。そのため、設計方法も異なっている。詳しい実装の方法は第 4 節において説明する。

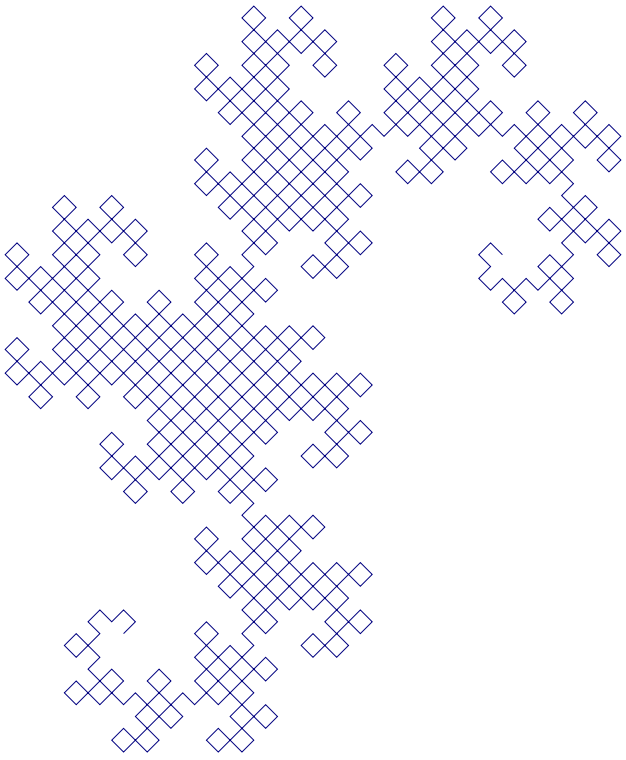


図 4: Heighway Dragon [7]

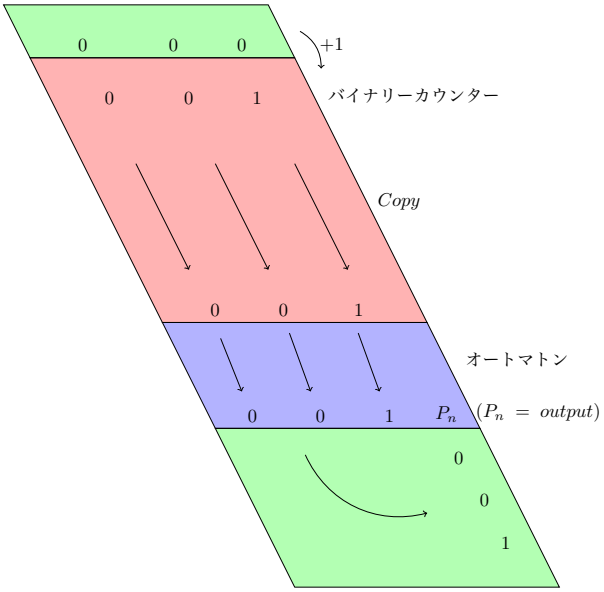


図 5: Oritatami System における Heighway dragon の概要図

2 Oritatami System

Oritatami System の導入としてまず定義に用いられる記号について説明をする (詳細は [8, 3] を参照)。

B は高分子の種類 (*bead types*) の有限集合と定義する。B の要素をビーズと呼ぶ。B^* は B に含まれる有限個のビーズの鎖を表す。$w \in B^*$ において、$w = a_1 \dots a_n$ ($|w| = n, a_1, \dots, a_n \in B$) と 2 つの添字 i, j ($i \leq j$) に対して $w[i, j]$ は部分文字列 $a_i a_{i+1} \dots a_j$ を表す。このとき $i = j$ の場合は単に $w[i]$ と書き、i 番目の文字を表す。ある整数 k に対して、$w[1, k]$ を w の接頭語 (prefix) といい Pref (w) で w のすべての接頭語の集合を表す。

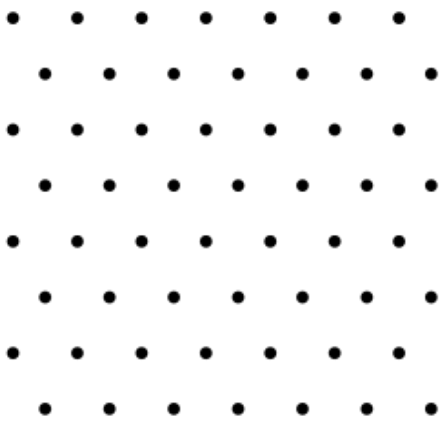


図 6: 三角格子

Oritatami は B^* に含まれるビーズの鎖をビーズの結合のルール $\mathcal{H}$ に従って三角格子上 (図 6) で折りたたむ。ここで $\mathcal{H}$ ($\mathcal{H} \subseteq B \times B$) は Rule Set 集合と呼ばれ、対称な二項関係を表している。

$\mathbb{T}$ は三角格子グラフとする。長さ n の文字列 $w \in B^*$ が与えられたとき、w でラベル付けされたパス (w-path) とは $P_w = (p_1, w[1]), (p_2, w[2]), \dots, (p_w, w[n])$ という列のことをいう。ただし $p_1, p_2, \dots, p_n$ は $\mathbb{T}$ 上の頂点である。*conformation* とは w-path と、水素結合の集合 H の組を指す ($H \subseteq \{i, j\} \mid 1 \leq i,\ i + 2 \leq j,\ (w[i], w[j] \in \mathcal{H})$)。$H$ に i と j の組が入っているとは、この *conformation* において i 番目と j 番目のビーズの間が水素結合していることを表す。また、w の prefix による部分的な *conformation* を *partial conformation* という。C_w は w による *partial conformation* の全ての集合を指し, $C_w = \{P_w, H\}$ と表す。

次に伸長という操作について説明する。ある *conformation* C_1 と C_2 において、C_2 が C_1 の $a \in B$ による伸長であるとは以下のように定義できる。

伸長 (elongation) の定義

$C_1 = (P_w, H)$ とするとき、C_2 は C_1 の a による伸長であるとは、ある頂点 p と $H' \subseteq \big\{\{i, |w|+1\} \mid i < |w|,\ p_i \text{ と } p \text{ が隣接},\ \{a_i, a\} \in \mathcal{H}\big\}$ において $C_2 = \{P_w \cdot (p, a),\ H \cup H'\}$ となる。このとき $C_1 \to_a C_2$ で表す。

高分子 1 つによる伸長は有限な文字列による文字列による伸長へと以下の要領で再起的に拡張することができる [3]。

伸長の再起的な拡張

1. 任意の構造 C に対し、$C \to^*_\lambda C$ が成立する。

2. ある文字列 $w \in B^*$ と高分子 $b \in B$ に対し、C_2 が文字列 wb による伸長、すなわち $C_1 \to^*_{wb} C_2$ であるとは、$C_1 \to^*_w C'$ かつ $C' \to_b C_2$

また、1 つの *conformation* は複数の *conformation* に伸長可能である。例えば図 7 では C が C_1, C_2 の 2 つに伸長可能であることを示した。ある有限な *conformation* C を有限な文字列 $w \in B^*$ で伸長可能な *conformation* の集合を $\mathcal{E}(C, w)$ とする。すなわち $\mathcal{E}(C, w) = \{C' \in C \mid C \to^*_w C'\}$ である。

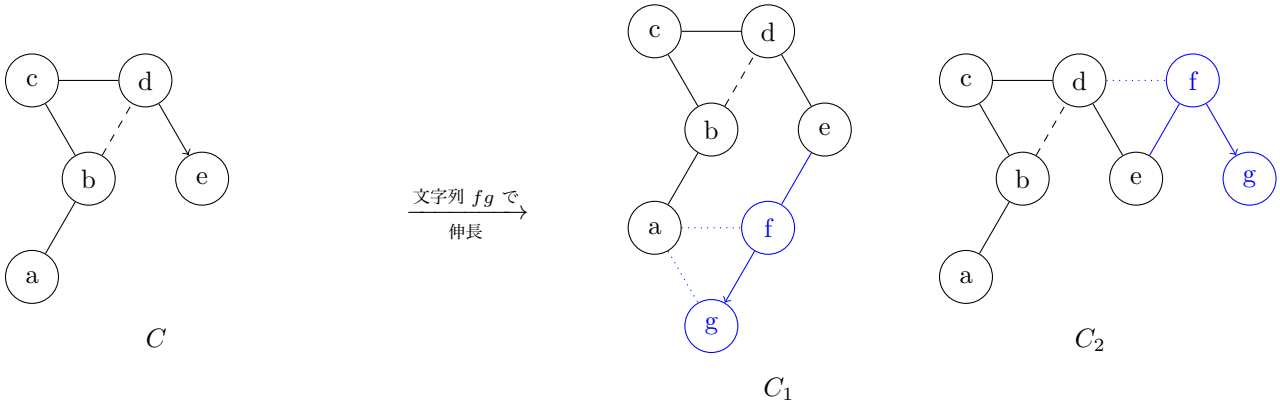


図 7: ある *conformation* C が文字列 $fg \in \Sigma^*$ によって伸長されてできる *conformation* C_1, C_2。
黒い実線は固定されたビーズの鎖 (path)、破線は水素結合を表し、青の実線は転写されたばかりのビーズの鎖、点線はその水素結合を表す。

Oritatami System は $O = (\mathcal{H}, \alpha, \delta, \sigma, w)$ の 5 つ組で定義される。文字列 w、RuleSet 集合 $\mathcal{H}$ はすでに定義されている。α は *arity* と呼ばれ、 一つの分子が他の分子と形成できる水素結合の最大数を表す。形式的には

$$\text{全ての自然数 } k \geq 1 \text{ について}, |\{\{i, j\} \in H | i = k \ \ or \ \ j = k\}| \leq \alpha$$

が成り立つ。また arity が最大であるとは、arity の値が採用した grid の degree に等しいことを意味する (三角格子 grid であれば 6)。δ ($\delta \geq 1$) は *delay* と呼ばれる正整数を指す。σ は *seed* と呼ばれる初期構造で、システムが動作し始める前から存在する *conformation* を指す。

前節でも述べたが Oritatami System は共転写性フォールディングの数理モデルであり、1 次構造 w を共転写的に安定した構造、つまり *conformation* へと折りたたむ。ここでいう *conformation* の安定性はエネルギー関数で定められている。この関数は *conformation* のエネルギーの値を実数で表している。

$$\Delta G(P, H) = -|H|$$

この関数より Oritatami System はエネルギーの低く安定した、つまり水素結合の数の多い *conformation* へと折りたたむ。Oritatami System の動作は次のように定義される。

Oritatami System の動作

ある高分子 b、長さ $\delta-1$ の文字列 $x \in B^*$、*conformation* C_1 の b による伸長 C_2 に対し、O が $bx \in w$ を転写している際に C_1 を C_2 へと折りたたむのは、

$$C_2 \in \mathrm{argmin}_{C \in \mathcal{E}\,(C_1,b)} \min\{\Delta G\,(C') | C' \in \mathcal{E}\,(C, x_p),\ x_p \in \mathrm{Pref}\,(x)\}$$

となるときであり、$C_1 \overset{O}{\hookrightarrow}_{bx} C_2$ とかく。

このとき重要なのは、長さ δ の文字列 bx が結ぶ水素結合の数を探索して安定した *conformation* を探しているが、一回の動作では初めの文字である b のみが固定されるということである。以下に例を示した。

例 2.1　Rule Set 集合 $\mathcal{H} = \{\{a,d\},\{a,e\},\{b,c\},\{c,d\}\}$, arity $\alpha = 2$, delay $\delta = 2$ とする。図 8 の右側は C の fg による伸長可能な *conformation* C_1, C_2 の 2 つを表している。この時 $C_1, C_2 \in \mathcal{E}\,(C, f)$ となる。水素結合の数が大きいのは明らかに C_1 であり、始めの文字 f を固定してできる *conformation* が、図の矢印の先に示した C' となる。C' の時点ではまだ f のみが固定されている。

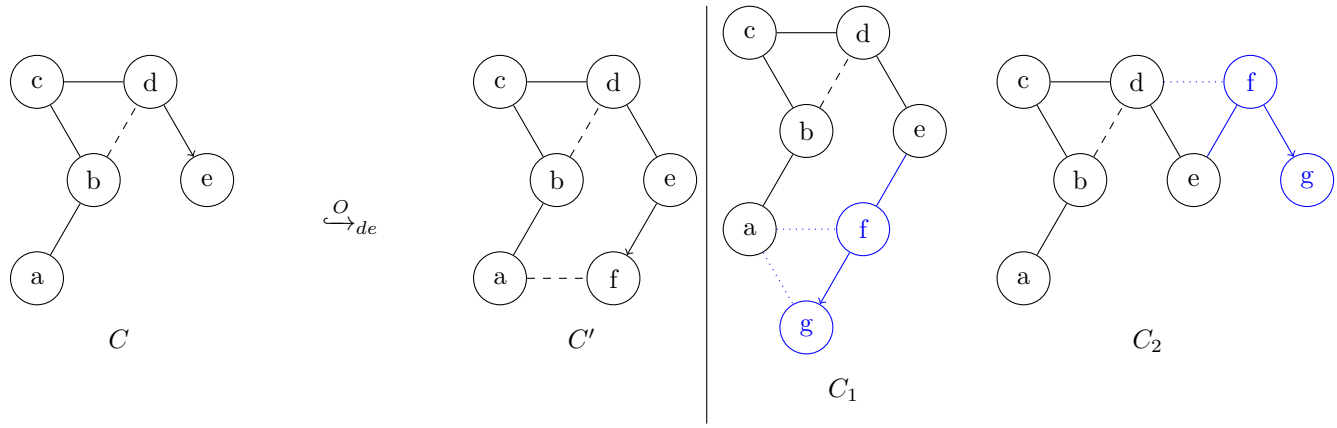


図 8: $C \overset{O}{\hookrightarrow}_{de} C'$ を図で表している。

O によってつくられる *conformation* の集合 $\mathcal{F}(\mathcal{O})$ は次のように再起的に定義される。

$\mathcal{F}$ の定義[3]

1. seed σ は $\mathcal{F}(\mathcal{O})$ に含まれている。
2. ある *conformation* $C_i \in \mathcal{E}(\sigma, w[1,i])$ に対し、
   $C_i \in \mathcal{F}(O)$ かつ $C_i \overset{O}{\hookrightarrow}_{w[i+1,i+\delta]} C_{i+1}$ ならば $C_{i+1} \in \mathcal{F}(O)$ である。

このとき、ある有限な *conformation* $C_i \in \mathcal{E}(\sigma, w[1,i])$ が終端構造であるとは以下が成り立つときである。

1. 1 次構造 w が有限であり, $i = |w|$ である。
2. $i < |w|$ かつ、 $C_i \overset{O}{\hookrightarrow}_{w[i+1,i+\delta]} C_{i+1}$ であるような *conformation* C_{i+1} が存在しない。

また、O によってつくられる終端構造以外のどんな *conformation* $C_i \in \mathcal{F}(O) \cap \mathcal{E}(\sigma, w[1,i])$ に対して、$C_i \overset{O}{\rightarrow}_{w[+1,i+\delta]} C_{i+1}$ となるような C_{i+1} が唯 1 つ存在するとき、O は決定的 (deterministic) であるという。

3 Heighway Dragon

Heighway Dragon はフラクタルであり、Paperfolding Sequence という数列から作られることで知られている。Paperfolding Sequence とは、一枚の紙を半分に折りたたみ、その折りたたまれた紙をさらに半分に同じ方向に折りたたむ、という操作を複数回行い、最後にその紙を開くことで作ることができる開かれた紙の “山 (ここでは 1)” と “谷 (0)” の列である。例えば一回半分に折りたたみ、その紙を 90 度開くと図**??**の図形を得ることができる。さらに二回折りたたみ 90 度に開くと、図**??**のような図形を得ることができる。10 回折りたたみ 90 度に折りたたむと図 9 のようになり、有名な Heighway Dragon の図形を得ることができる。また、この数列は Automatic Sequences という数列の 1 つであり、オートマトンから算出することができる ([7] を参照)。

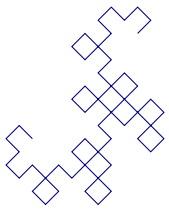


図 9: Paperfolding Sequence: Ten Folds [7]

3.1 Automatic Sequences

Automatic Sequences の導出として、出力付きのオートマトン、DFAO (Deterministic Finite Automata with Output) について説明する。DFA は入力の文字列に対して受理か拒否のみを出力するが、DFAO では各状態 (state) に出力の文字が割り振られている。そして、DFAO は入力に対して最後に移動した状態に割り振られた文字を出力する。また、整数 k ($k \geq 2$) において入力の文字 $\Sigma = \Sigma_k \coloneqq \{0, 1, \ldots, k-1\}$ のとき DFAO を k-DFAO と呼ぶ。例えば図 10 の場合は 2-DFAO である。この DFAO に “01110” という文字列を入力したとき、最終的に移動する状態は q_2 となり、この DFAO は “0” を出力する。

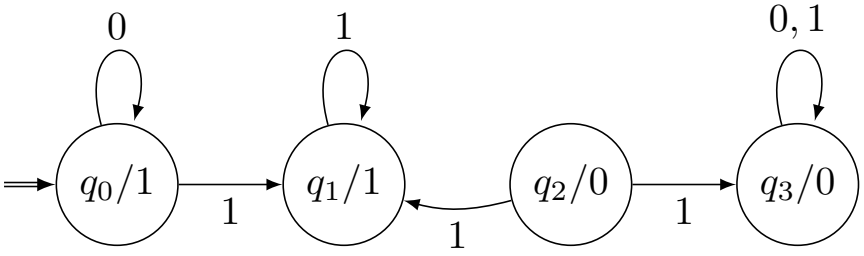


図 10: Example of DFAO[7]

Automatic Sequences は DFAO から算出できる文字列である。文字列 $(a_n)_{n\geq 0}$ が automatic sequence である時、a_n は文字列 n を k ($k \geq 2$) 進数表現で一文字ずつ DFAO に入力し、最後に移動した状態で得られる文字をマッピングすることで得られる。この時 a_n を k-automatic sequences と呼ぶ。定義としては以下のようになる。

k-automatic sequence

有限な文字の集合 Δ 上の文字列 $(a_n)_{n\geq 0}$ が k-automatic であるとき、$a_n = \tau(\delta(q_0, w))$ となる k-DFAO $M = (Q, \Sigma_k, \delta, q_0, \Delta, \tau)$ が存在する。
このとき、
Q は有限な状態の集合
$\delta : Q \times \Sigma \to Q$ 状態遷移
$q_0 \in Q$ は初期状態
$\tau : Q \to \Delta$ 出力関数
$w : [w] = [n]_k$ DFAO に入力する文字列で、n の k 進数表現となる

とする。

例えば図 10 の DFAO について考える。$n = 0123456\ldots$ のとき、出力 a_n は以下のようになる

$$n = 0\ 1\ 2\ 3\ 4\ 5\ 6\ldots$$
$$(a_n)_{n\geq 0} = 1\ 1\ 0\ 1\ 1\ 0\ 0\ldots$$

まず、最初の “0” を入力すると、図 11 の DFAO では q_0 で終わり出力は “1” となる。従って a_0 の最初の文字は 1 となる。次に “1” を入力すると、状態遷移は q_1 で終わり出力は “1” となる。従って a_n の次の文字は 1 となる。次の数字は “2” だが、この DFAO は 2-DFAO なので入力は “10” となる。状態遷移は q_2 で終わるので、出力は “0” となる。従って a_2 の文字は 0 となる。以下はこのくりかえしである。

3.2 Paperfoldig Sequence

Paperfolding Sequence の DFAO は図 11 のようになる。

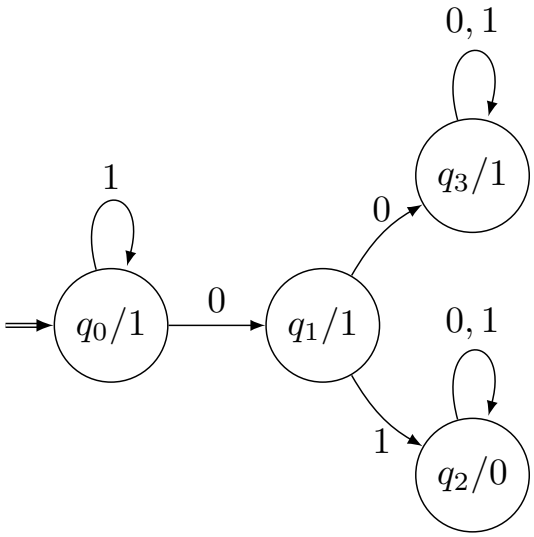


図 11: Paper folding sequence の 2-DFAO[7]

この DFAO は入力した文字列を LSB から読み込んで出力する。例えば “1010” という数字を入力した場合、まず最初 “0”を読み込んで q_1 に移動し、次に “1” を読み込んで q_2 に移動する。次の “0” と “1” を読み込んで

も q_2 のままなので、出力は “0” となる。出力の文字列を P_n とすると、以下の通りになる。

$$\begin{aligned} n &= 0\,1\,2\,3\,4\,5\,6\,7\,8\ldots \\ P_n &= 1\,1\,0\,1\,1\,0\,0\,1\,1\ldots \end{aligned}$$

上記で説明したように、P_n の文字列は Heighway Dragon の折り目が山 (ここでは 1) か谷 (0) かを表している。この P_n を利用して Heighway Dragon を直線で書き表すにはまず直線を一本描き、P_n の最初の文字が “1” なので右 (ここでは “1” を右とする) に直角に直線を描く。すると、図**??**の一回折りたたんだ Heighway Dragon となる。次に P_n の 2 番目の文字が “0” なので、直角に左 (ここでは “0” を左とする) の直線を描く。これを後二回繰り返すと、図**??**の二回折りたたんだ Heighway Dragon が出きあがる。この曲がる方向の文字列を $D_n(D_n = \{R, L\})$ とすると、図 11 の DFAO からは n を入力して以下の文字列を得ることができる。

$$\begin{aligned} n &= 0\,1\,2\,3\,4\,5\,6\,7\,8\ldots \\ P_n &= 1\,1\,0\,1\,1\,0\,0\,1\,1\ldots \\ D_n &= R\,R\,L\,R\,R\,L\,L\,R\,R\ldots \end{aligned}$$

4 Heighway Dragon の Oritatami System での実装

4.1 概要

Heighway Dragon を実装する上で、太田らが [9] で開発したシミュレーターを使用した。また、δ と α は $\delta = 3, \alpha = 3$ とした。前節で述べたように、入力 n による DFAO の出力 P_n の方向に従って直線を書くことで Heighway Dragon を描くことができる。そこで、Oritatami System 上では大まかに分けて次の 3 つの操作を再起的に繰り返すことで Heighway Dragon を描く。

Oritatami System 上での Heighway Dragon の大まかな動作

1. $n \rightarrow n + 1$
2. DFAO に n を入力し, P_n を出力する。
3. P_n に従い方向転換する。

この操作を行うために、「カウンターパート」、「伝達パート」、「DFAO パート」、そして「方向転換パート」の 3 つのパートを作成する。図 12 に大まかな構造を示した。また、本研究は生方との共同研究で行っており、私は上記で述べた操作での “1” と “2” の部分を担当した。

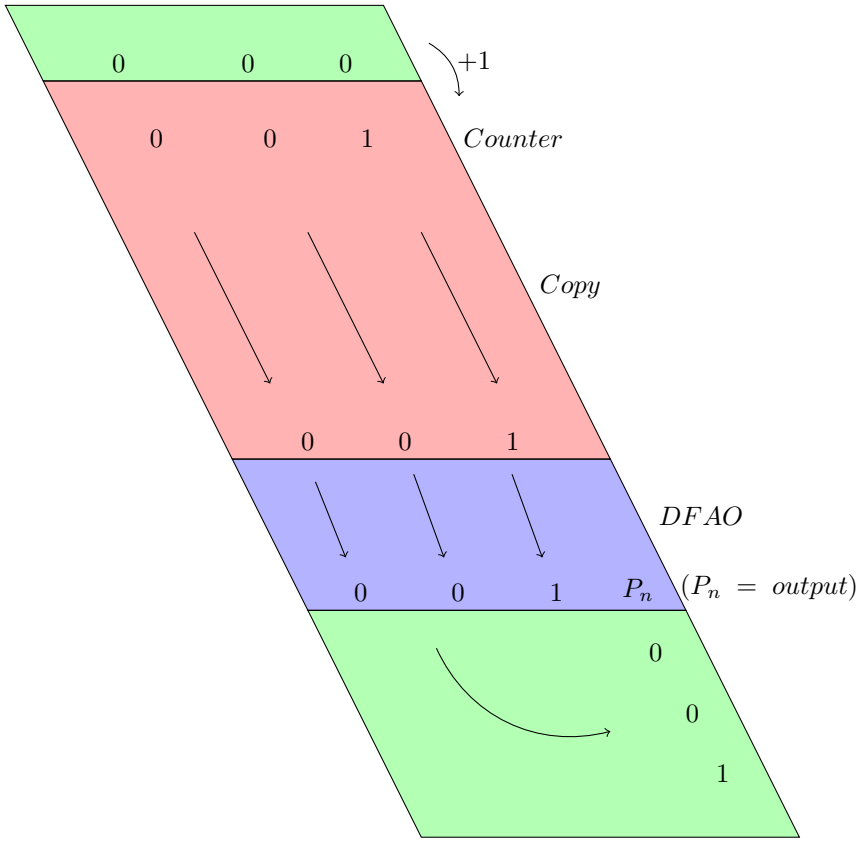


図 12: Heighway dragon の概要図

Heighway Dragon で使用している構造は主に図 13 のような「平行四辺形型」と「グライダー型」の 2 つを

使用している。

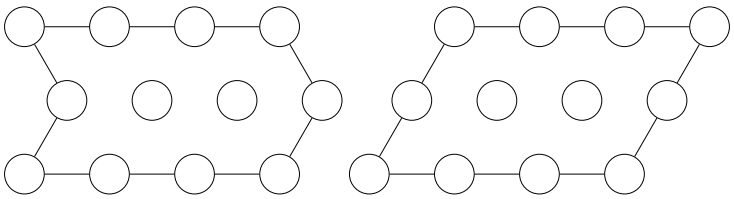


図 13: グライダー型と平行四辺形型

4.1.1 情報の伝達

図 12 の矢印は情報の伝達を表している。主に Heighway Dragon では、情報伝達の方向は上から下へとなっている。今回情報伝達の方法は [8] で作成された半加算器を参考にした。上の列から下の列への伝達は「折りたたまれた構造の底に配置されるビードタイプ」を利用している。下の列の構造は「折りたたみのスタートの位置」と「上に配置されているビードタイプ」の 2 つによって複数の異なる構造へと折りたたまれる。

図 16 に例を示した。黒色のビーズは折りたたまれた構造を、黄色の直線は水素結合を表している。ビーズの中の数字はビードタイプを表している。赤色のビーズが上に配置されたビードタイプである。ex1, ex2 のルールセットは $R = \{(1,6),(2,5),(4,9),(2,300),(3,100),(3,300),(6,200),(7,100)\}$ とする。これらの構造は “300” というビーズを配置することで折りたたみ方を変えることができる。$\delta = 3$ なので、最初に “1”, “2”, “3” の 3 個のビーズが探索を始める。この時上に 300 のビーズが配置されていた場合、2 と 3 は水素結合を持つので上に引き寄せられる。これは 3 が 100 と水素結合した場合、結合は 1 本のみだが、2 と 3 が 300 と水素結合した場合は 2 本持つためである。300 が配置されていない場合、3 個のビーズが結合を持つのは 3 と 100 のみなので、ex1 の構造へと折りたたまれる。この結果 ex1 と ex2 の底に配置されるビードタイプはそれぞれ (3-4-9-10) と (7-8-9-10) となる。(3-4) と (7-8) は共通のビードタイプではない。従って、次にこの構造の下の列で折りたたまれる構造は (3-4) と (7-8) の違いによって複数の構造へと折りたたむことができる。

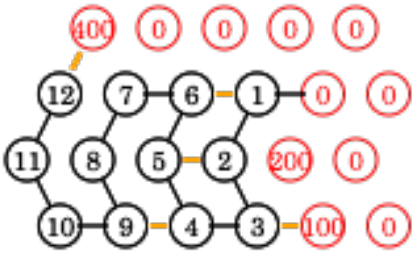


図 14: ex1

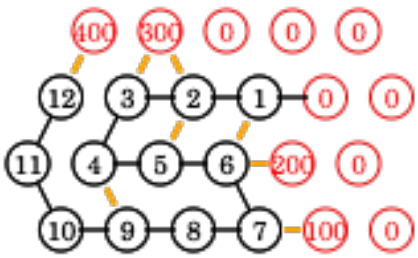


図 15: ex2

図 16: 情報伝達の例

以下より Heighway Dragon における各パートを大まかに説明する。

4.1.2 カウンターパート

カウンターパートは [8] を参考にして作成した。このパートはジグパス (zig pass・←) とザグパス (zag pass・→) にわかれている。左右のターン部分以外は幅と高さが 4×3(ビーズ 4 の幅とビーズ 3 つの高さの意)

の平行四辺形型と、14×3 の平行四辺形型の構造の 2 つによって構成されている。

ジグパス ($\leftarrow$)

ジグパスでは次の 2 つのパート (以下より「モジュール」と呼ぶ) によって構成されている。各モジュール内は同じルールセットとビードタイプで構成されている。

モジュール A　半加算器 (4×3)
スペーサー (14×3)

モジュール A は A0, A1, A2, A3 という 4 種類の構造からなる。スペーサーはモジュール A の場所がお互いに近すぎると誤作動が起きてしまうので、距離を開けるために配置している。スペーサーはこのモジュールに限らず全てのもので言えることだが、スタートの位置と終了の位置が同じになっている。例えば、スタートの位置が上だった場合終了の位置は必ず上となる。ジグパスの動作の例を図 17 に示す。青色の部分がモジュール A を表しており、黒い部分はスペーサーと左右のターン部分である。

モジュール A ではスタートの位置と上に存在する構造の 2 つの入力によって 4 種類の構造を構成する。この時、上に存在する構造は入力の値である “0”, もしくは “1” を表している。折りたたみのスタートの位置が下の場合、$carry = 1$ となりこの半加算器はキャリーを持っていることを表している。スタートの位置が上の場合は、$carry = 0$ となってこの半加算器はキャリーを持っていないことを表す。これらの入力に対し、折りたたまれた構造によって下の列に “0”, もしくは “1” を出力する。

例えばスタートの位置が下で、上の構造が 0 を表していた場合、このモジュール A は $carry = 1$ なので、出力は 1 を表す A1 となる。最初スタートの位置は下で $carry = 1$ だが、$carry = 0$ となると構造の終了位置は上となる。そのため以降の構造のスタートの位置は上となる。また、図 12 に示した通りカウンターは Heighway Dragon の最初に出現するパートとなっている。上に存在する構造は seed、もしくは前の直線部分の方向転換パートである。

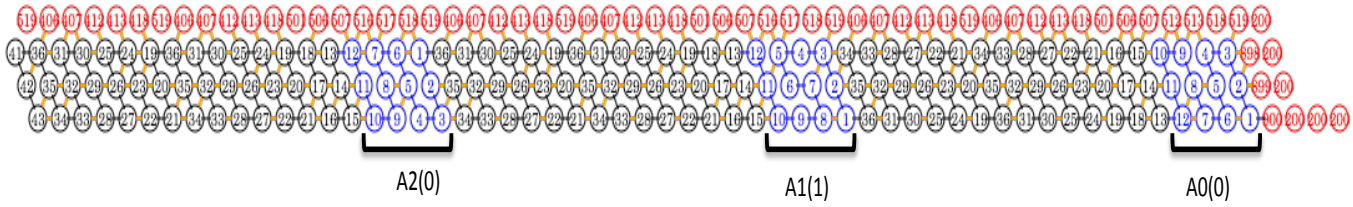


図 17: カウンターパートにおけるジグパスの例。青色がモジュール A を指しており、黒色のビーズはスペーサーを指している。() 内の数字は出力を指す。赤色の Seed が入力となる。この例の場合 $n = 1$、つまり 001 となっている。$n \rightarrow n+1$ より数字は 010 となるので、モジュール A は左から 0, 1, 0 を表す構造に折りたたまれている。今回スタートの位置は Seed の左下に位置するビードタイプ 900 のビーズとなっている。最初 carry=1 なのでスタートは下となっているが、 2 番目の構造で 0+1=1 となることから、以降は carry=0 となる。従って、最初に折りたたまれる一番左の構造の終了位置は上となり、次からのスタートと終了の位置も上となる。

ザグパス (→)

ザグパスでは次の 2 つのモジュールによって構成されている。

モジュール B　伝達 (4×3)
スペーサー (14×3)

モジュール B は 2 種類の構造 B0, B1 によって構成されている。スタートの位置全て上で統一されており、入力は上に存在するモジュール A の構造が表す 0 と 1 となる。折りたたまれた構造は入力の値を下の列に出力している。つまり、モジュール B の役割は上の列の値を下の列に伝達することである。モジュール B が上に存在するモジュール A を読み込みながら折りたたまる様子を図 18 に示す。

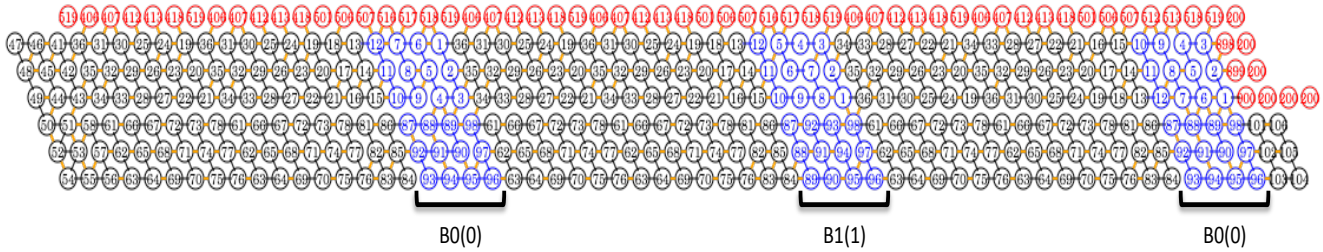


図 18: モジュール B の動作例。2 段目の青色の構造がモジュール B となっている。() 内の数字は出力を指す。モジュール B は上に存在するモジュール A の底に現れるビードタイプによって、2 種類の構造に折りたたまれる。今回の場合、上の段の構造は 010 を表しており、モジュール B も右から 0, 1, 0 (B0, B1, B0) を表す構造に折りたたまれている。

4.1.3　伝達パート

モジュール C,D　伝達 (4×3)
スペーサー (14×3)

まず伝達パートを作る理由を説明する。図 12 を見るとわかるように、Heighway Dragon 上でカウンターパートや DFAO パートは 1 直線の一部である。そして Heighway Dragon を作るために、直線は幅に比べてある程度の高さを持つ必要がある。カウンターパートと DFAO パートのみでは幅の方が高さに比べて大きくなってしまうので、高さを大きくするため伝達パートが必要となった。このパートは直線の高さを増やしつつ、上の情報 (0 か 1 の値) を底に伝達する、というパートである。これは幅に対して十分な高さを得るまで再起的に繰り返して折りたたまれる。

モジュール C, D は 2 つとも役割は同じである。この 2 つは上に配置されている 0 と 1 を表すビードタイプを読み込み、2 種類の構造へと折りたたまれる。

図 19 は伝達パートの動作例である。紫色の部分が情報を保持しており、カウンターパートと同じくジグパスとザグパスに分かれている。

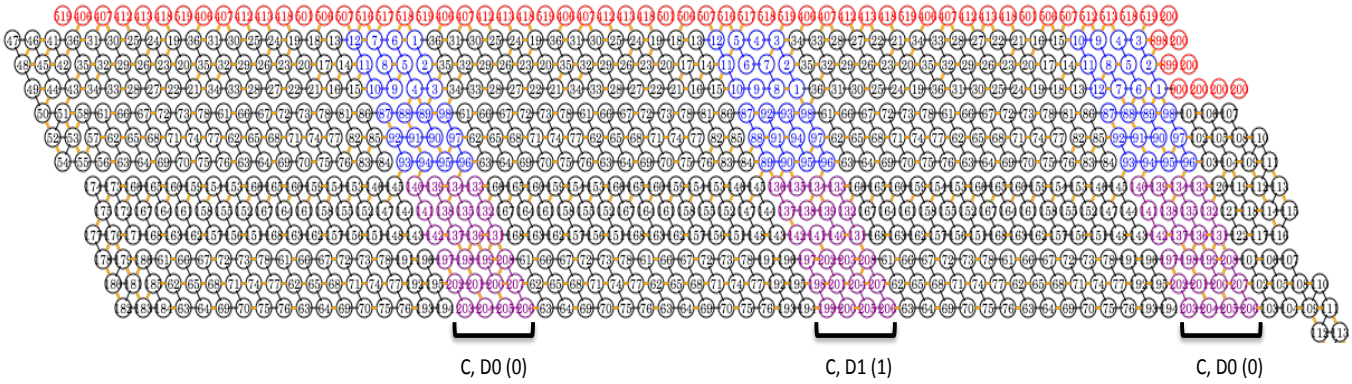


図 19: モジュール C, D における動作例。紫色の部分が伝達パートを指す。() 内の数字は出力を指す。1 段めのジグパスが上に存在するモジュール B を読み込みながらモジュール C に折りたたまれる。2 段めのザグパスがジグパスであるモジュール C を読み込みながらモジュール D に折りたたまれる。

4.1.4 DFAO パート

Paperfolding sequence の DFAO は、状態 q_2 か q_3 に一度進むとそれ以降、どのような文字が入力されても他の状態に遷移しない、という構成になっている。また、出力は状態 q_2 で状態遷移を終えた時のみ “1” を出力し、それ以外の場合出力は “0” となる。従って本論文では、この DFAO を状態 q_2 に遷移するか否かについて注目した。状態 q_2 に遷移する数字は、LSB から探索して最初に出会った 0 の次に 1 が続くものである。DFAO は初めて 0 が入力されると、状態 q_1 に遷移する。状態 q_1 に状態遷移した後、次に出会う数字によって状態 q_2 か q_3 に遷移する。0 に出会った場合は状態 q_3 に遷移し、1 に出会った場合は状態 q_2 に遷移する。このことから、DFAO を以下のように 2 つに分けて実装した。また、DFAO パートは左右のターン部分以外は幅と高さが 8×3 のグライダー型と、10×3 のグライダー型の 2 つによって構成されている。

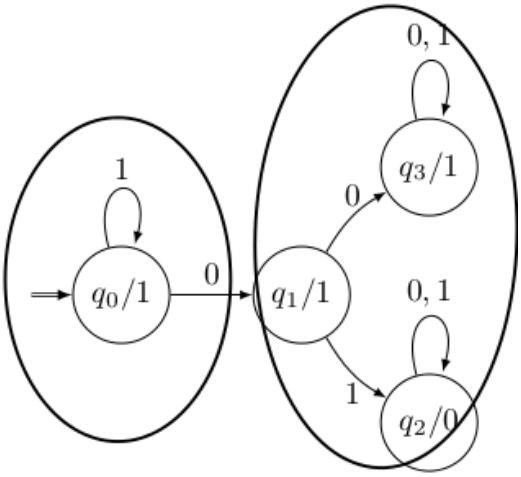


図 20: 2つに分けた DFAO

4.1.5 DFAO パート前半部分

ジグパス (←)

ジグパスのモジュールは次のようになっている。

モジュール E “最初の 0”探索 (8×3)

スペーサー (10×3)

モジュール E の役割は DFAO の前半部分、状態 q_1 への状態遷移である。使用する構造は 4 種類あり、E0, E1, E2, E3 としている。入力は上に存在する構造とスタートの位置であり、上に存在する構造は 0 もしくは 1 の値を表している。スタートの位置が下の場合、状態 q_1 に遷移する最初の 0 を探索するサーチモードとなる。最初の 0 を見つけると、折りたたみの終了の位置は上となり、以降のスタートの位置は上となる。スタートの位置が上の場合、上に存在する構造が表している入力の値を下の列に出力している。これらの入力に対し、モジュール E の出力の値は 0, 最初の 0, 1 である。モジュール E 内では最初の 0 を出力する構造のみ、後半ビーズ 12 個分の折りたたみ方が異なっている。この折りたたみによって底に現れるビードタイプが最初の 0 を意味しており、前半ビーズ 12 個分の折りたたみ方によって “0”か “1”を示している。

モジュール E の動作例を図 21 に示した。

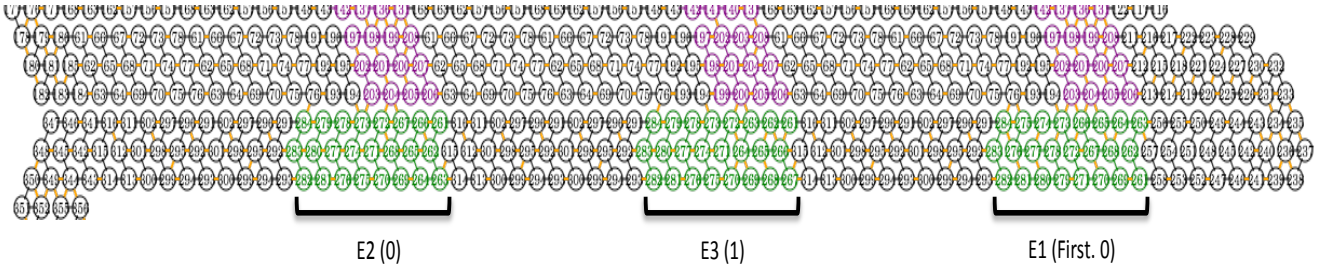


図 21: モジュール E の動作例。緑色の部分がモジュール E を指している。() 内の数字は出力を指す。最初、モジュール E はサーチモードとなっているため、スタートの位置は下になっている。今回の入力は 010 であり、最初の 0 である 1 つ目の構造の終了の位置が上となっている。

ザグパス ($\rightarrow$)

ザグパスのモジュールは次のようになっている。

モジュール F　伝達 (8×3)
スペーサー (10×3)

モジュール F は 0, 最初の 0, 1 の 3 種類の情報を下の列に伝達している。各構造は F0, F1, F2 となっている。モジュール F における折りたたみのスタートの位置は上で統一している。入力は上に存在する構造が表す 0, 最初の 0, 1 である。

図 22 にモジュール F がモジュール E を読み込みながら折りたたまる様子を示した。モジュール F 内では F0 と F1 の後半ビーズ 12 個分の折りたたみは同じ折りたたみ方となっている。前半ビーズ 12 個分の折りたたみ方で最初の 0 とそれ以外の 0 を区別し、後半ビーズ 12 個分の折りたたみ方で 0 か 1 を区別している。

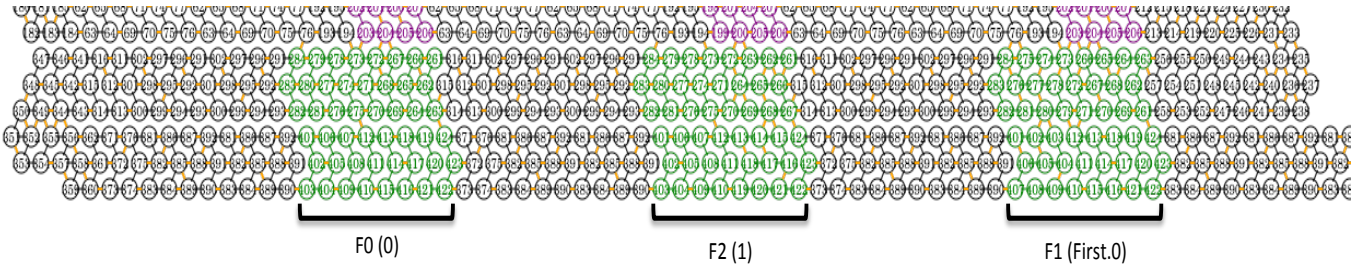


図 22: モジュール F の動作例。緑色のパートの 2 段目がモジュール F を指す。() 内の数字は出力を指す。今回の入力の文字は左から 0、1、最初の 0 となっているため、一番左の構造のみ前半部分が他 2 つと異なる折りたたみ方となっている。

4.1.6 DFAO パート後半部分

後半部分は前半部分で見つけた最初の 0 の後に 1 が来るか否かを探索している。

ジグパス (←)

ジグパスのモジュールは次のようになっている。

モジュール G　最初の 0(F1) の後の 1(F2) 探索 (8×3)
スペーサー (10×3)

モジュール G では次のような動作を行っている。

モジュール G での動作

1. スタートの位置は下で、最初の 0 と出会うと折りたたみの終了の位置が上になる。(サーチモード)
2. 最初の 0 の後に 1 が来るか否かを探索し、終了の位置が下に戻る。(DFAO の出力モード)
3. 折りたたみのスタートと終了の位置は下のまま、入力の値を下の列に伝達する。 (伝達モード)

上記の動作を行うために、モジュール G は 5 種類の構造 (G0, G1, G2, G3, G4) としている。入力は上に存在する構造とスタートの位置であり、上の構造は 0, 最初の 0, 1 を表している。動作 2 の時、最初の 0 の後に出会う 1 を “Key.1” とする。モジュール G は 0, 1, Key.1 を下の列に出力する。

図 23 にモジュール G の動作例を示した。

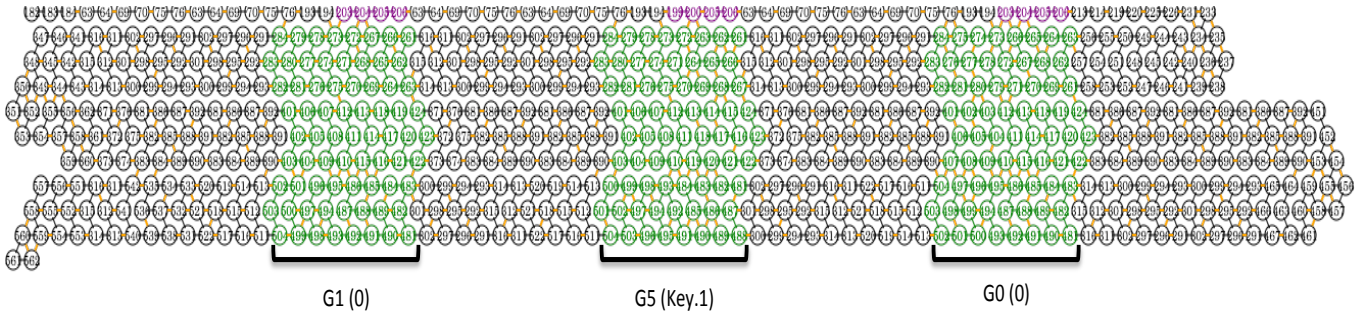


図 23: モジュール G の動作例。緑色のパートの 3 段目を指す。() 内の数字は下の列に出力する値を指す。最初のスタートの位置は下、つまりサーチモードとなっている。この時、入力の文字は右から最初の 0, 1, 0 となっている。最初の 0 と出会ったので、一番左の構造の折りたたみの終了位置は上となっている (上記の 1)。次の構造は出力を表している。数字は 1 なので、DFAO の状態 q_2 の出力を行いつつ、終了の位置が下となっている (上記の 2)。次の構造は入力の値を下の列に伝達している (上記の 3)。

DFAO 出力は、5 種類の内 Key.1 を表す G4 という構造が現れた時のみ 1 となる。モジュール G もモジュール F と同じく、前半ビーズ 12 個の折りたたみ方で 0 か 1 の情報を下に伝達している。後半ビーズ 12 個の折りたたみ方で DFAO の状態 q_2 に遷移したか否かを伝達している。

ザグパス (→)

ザグパスのモジュールは次のようになっている。

モジュール H　伝達と DFAO の出力 (8 × 3)
モジュール I　DFAO の出力 (4 × 3)
スペーサー (10 × 3)

モジュール H では情報の伝達と DFAO の出力を行うために、5 種類の構造 (H0, H1, H2, H3, H4) としている。入力は上に存在する構造とスタートの位置であり、上の構造は 0, 1, Key.1 を表している。他のモジュールと同じく底に配置されるビードタイプによって 0 か 1 を表している。この時、構造の終了の位置が DFAO の出力を意味する。最後に折りたたまれた構造の終了の位置が、その時の DFAO の出力となる。

モジュール H はスタートの位置と上に存在するモジュール G のビードタイプによって折りたたみ方を変える。モジュール H の最初のスタートの位置は下となっている。G4 と出会った時のみ、 折りたたみの終了の位置が上となり、次からの折りたたみのスタートの位置と終了の位置は上のままとなる。逆に G4 と出会わなければスタートと終了の位置は下のままである。つまり、最後の構造の終了の位置が上の場合は DFAO の出力が 0 となり、下の場合は出力が 1 となる。また、このモジュールの構造は後半ビーズ 12 個の折りたたみ方で 0 と 1 を表しており、次のパートはこの後半部分から情報を読み込む。全ての情報を読み込むとモジュール I に入る。

図 24 にモジュール H の動作例を示した。

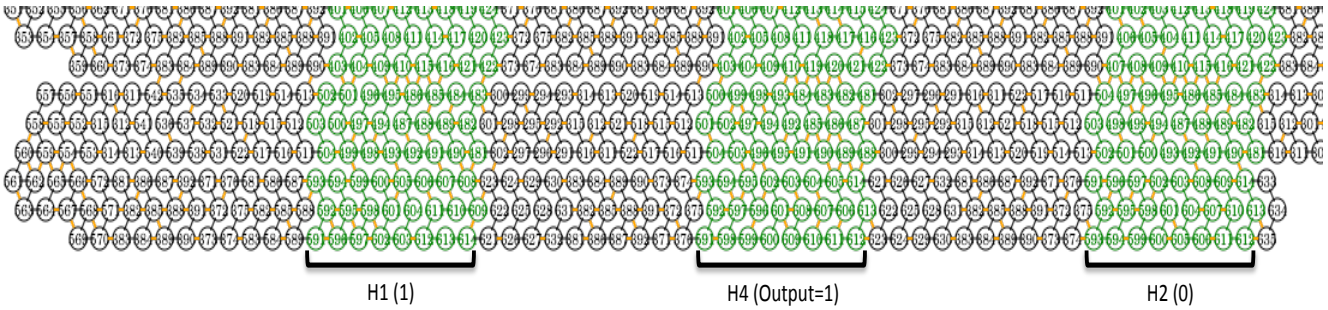


図 24: モジュール H における動作例。緑色のパートの 4 段目を指す。() 内の数字は下の列に出力する値を指す。最初のスタートの位置は下となっており、上に存在するモジュール G の構造のビーズを読みながら折りたたまれる。今回、上の構造は右から 0, Key.1, 0 を表している。従って、DFAO の出力は 0 となる。Key.1 を表す左から 2 番目の構造において、出力の値が 1 から 0、つまり終了の位置が下から上へと変わる。

モジュール I の役割は DFAO の出力を下の列に伝達することである。モジュール I は 2 種類の構造 (I0, I1) としている。入力はスタートの位置である。この時、2 つの折りたたみの終了の位置は上で統一されている。

図 25 にモジュール I の動作例を示した。

図 25: モジュール I における動作例。一番下の列のピンク色の構造がモジュール I を指す。今回の DFAO の出力は 0 となっているので、折りたたみのスタートの位置と終了の位置は上となっている。

4.1.7 方向転換パートへの伝達

次のパートは方向転換パートとなっている。n の値を DFAO の出力に基づいて左か右に折り曲げるパートである。しかし、モジュール H, I のままでは n の値は 4 種類の構造で表されているので、少し複雑である。そこで、次のパートでの動作をスームズに行うために、2 種類の構造へ整理する。なおこのパートはジグパス (←) のみで行っている。

モジュール J　DFAO の出力 (4×3)
モジュール K　n の伝達 (4×3)
スペーサー (14×3)

モジュール J とモジュール K では DFAO の出力とを 2 種類の構造で区別している。各構造はそれぞれ、(J0, J1) と (K0, K1) としている。この 2 つのモジュールは同じ折りたたみ方の構造を使用しているが、使用したルールセットとビードタイプは異なる。入力は上の構造が表す 0, 1 となっている。これらのモジュールにおける折りたたみのスタートの位置は上で統一している。

動作例を図 26 に示した。

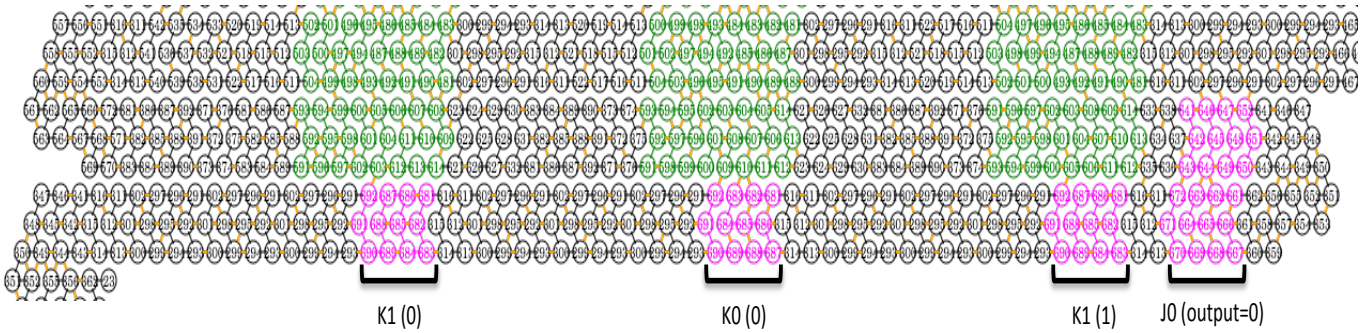


図 26: モジュール J,K における動作例。ピンク色のパートの 2 段目を指す。() 内の数字は下の列に出力する値を指す。一番左の構造はモジュール J を表している。次の 3 つの構造がモジュール K である伝達パートとなっている。

4.2 各パートの折りたたみについて

はじめに各パートで共通する 2 つの構造の折りたたみについて説明する。前節で述べたように今回使用した構造は大きく分けてグライダー型と平行四辺形型の 2 つである。図 27 のように情報伝達の方向がグライダー型と平行四辺形型では異なる (↓, ↘)。

グライダー型は比較的単純なルールセットで構成することが可能である。さらに同じルールセットで左右どちらの方向にも折りたたむことが可能となっている。平行四辺形型の場合、ジグパスとザグパスにおいて同じ形の *conformation* に折りたたむためには異なるルールセットが必要となる。

平行四辺形型を使う理由は、Heighway Dragon の直線を平行四辺形型に描くことが方向転換パートの設計上適していたからである。ただし、平行四辺形型の構造はグライダー型に比べて複雑なルールセットを必要とすることが多い。今回 DFAO パートで平行四辺形型ではなくグライダー型を利用したのは、平行四辺形型では折りたたむことが困難だったためである。

図 27: グライダー型と平行四辺形型の違い。緑色のビーズが折りたたみのスタートの位置だった場合、グライダー型は赤色、平行四辺形型は青色のビーズが下の入力となる (今回使用した構造の場合)。グライダー型は真下に現れるのに対して、平行四辺形型は斜め左下に現れる。

4.2.1 Seed

Seed はカウンターパートの上に入力として存在する。Heighway Dragon 上ではカウンターパートの上には方向転換パートが存在するが、最初のみ Seed からスタートする。そのため Seed は方向転換パートの最後のパスで “0” と “1” を表すビードタイプを置く。Seed0, Seed1 のビードタイプは 501~516 で, スペーサーの部分のビードタイプは 401~412 を使用している。例えば Seed0 と Seed1 の 1 個目のビードタイプは 501 で、スペーサーの部分の 5 個目のビードタイプは 405 となる。これは他の構造でも同じとする。Seed では底に現れるビーズのみ配置している。図 28 に Seed の構造を示した。

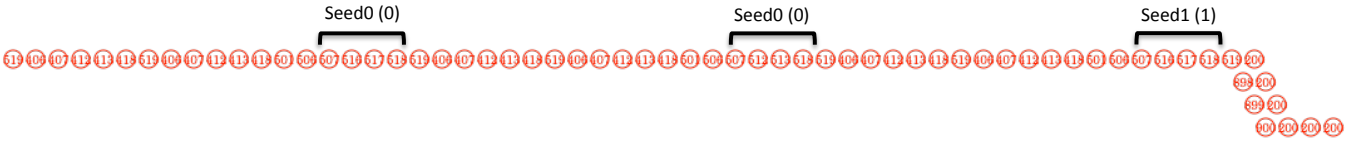


図 28: Seed の構造

4.2.2 **カウンターパート**

ジグパス (←, **モジュール** A)

カウンターパートのジグパスにおいて、モジュール A は 4 種類の構造が同じルールセットで成り立っている。また、使用するビードタイプは 1~12 とする。図 29 にモジュール A 内の構造と次のザグパスに情報を伝達するための底に現れるビードタイプを示す。

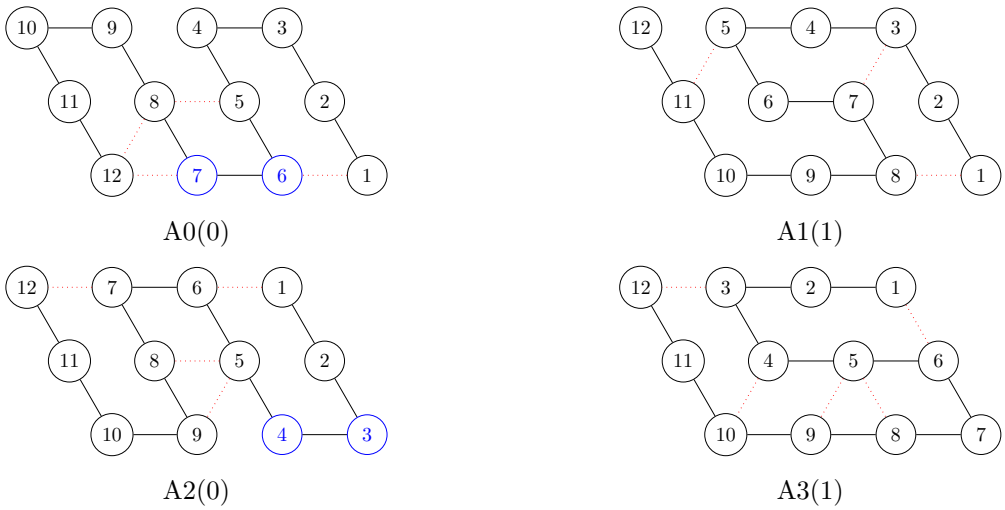


図 29: モジュール A の構造。赤色の破線は水素結合を表す。() 内の数字は下の列に出力する値を指す。青色のビーズは 4 種類の構造の中で 0 を表す時のみ現れる。これらのビーズに反応して次の列で 0 と 1 を区別する。

まず $carry = 1$ の状態、つまりスタートの位置が下の場合、4 個目のビーズまでは同じ折りたたみ方をする。上に 0 を表すビードタイプが配置されている時、モジュール A は 5 個目のビーズと上の 10, 11 個目のビーズとの間に水素結合を持つ。ビーズは上に引き寄せられ、A1 へと折りたたまれる。この時 $0 + 1 = 1$ で $carry = 0$ となるので、終了位置が上となる。上に 1 を表すビードタイプが配置されていた場合、5 個目のビーズは上と反応しないので、A0 へと折りたたまれる。$carry = 0$ の状態、つまりスタートの位置が上の場合、3 個目までのビーズが Seed1 のビードタイプと水素結合を持つ。ビーズは上に引き寄せられ、A3 の形に折りたたまれる。Seed0 の場合ビーズは反応しないので、A2 の形に折りたたまれる。スタートの位置が下の時はこれらのビーズの探索範囲外となるので、問題なく共存している。

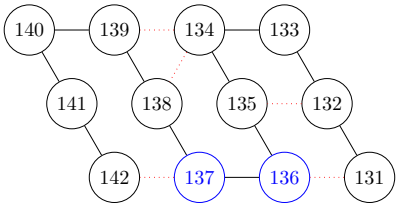
ザグパス (→, **モジュール** B)

モジュール B は上記のモジュール A の底に現れるビードタイプによって、B0 と B1 の二種類の構造へと折りたたまれる。使用するビードタイプは 87~98 としている。図 30 にモジュール B 内の 2 種類の構造と、次のジグパスに伝達するビードタイプを示す。入力が 0, つまり上の列に折りたたまれた構造が 0 を表していた場合、構造の底に現れるビーズは (3-4-9-10), もしくは (7-8-9-10) となる。モジュール B の 3 個目までのビーズ (87 ~ 89) はビードタイプ 3, 4, 7, 8 と水素結合を持つ。その結果、ビーズは上に引き寄せられ、0 を意味する B0 に折りたたまれる。逆に入力が 1, つまり上に存在する構造が 1 を表していた場合、上に水素結合を持つビーズはないので、1 を意味する B1 へと折りたたまれる。

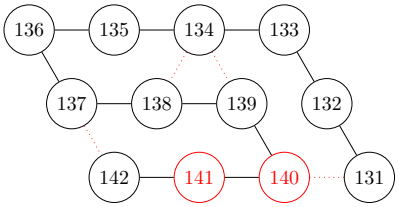
図 30: モジュール B の構造。赤色の破線は水素結合を表す。() 内の数字は下の列に出力する値を指す。青色のビーズは 2 種類の構造の中で 0 を表す時、赤色のビーズは 1 を表す時のみ現れる。

4.2.3 伝達パート (モジュール C,D)

伝達パートは前節で述べたようにジグパス、ザグパスともに 2 種類の構造で構成されたモジュール C, D によって情報を伝達している。また、使用するビードタイプはそれぞれ 131~142 と 197~208 としている。図 31, 32 にモジュール C, D 内の構造と底に現れるビーズのどの部分が情報を伝達しているかを示した。

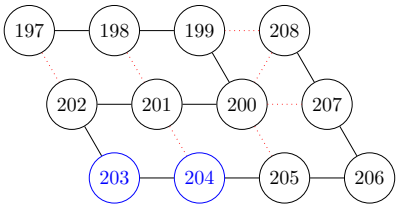


C0(0)

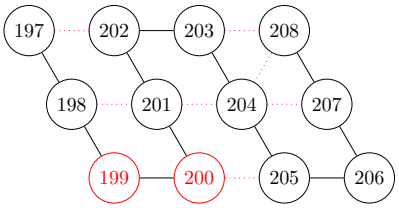


C1(1)

図 31: モジュール C の構造。赤色の破線は水素結合を表している。() 内の数字は下の列に出力する値を指す。青色のビーズは 2 種類の構造の中で 0 を表す時、赤色のビーズは 1 を表す時のみ現れる。



D0(0)



D1(1)

図 32: モジュール D の構造。赤色の破線は水素結合を表す。() 内の数字は下の列に出力する値を指す。青色のビーズは 2 種類の構造の中で 0 を表す時、赤色のビーズは 1 を表す時のみ現れる。

伝達パートを繰り返し折たたむことで直線の高さを伸ばす工程は次のように再起的に行っている。

伝達パートでの動作

1. ジグパス (←) でモジュール B を読み込みながらモジュール C を折りたたむ.
2. ザグパス (→) でモジュール C を読み込みながらモジュール D を折りたたむ.
3. ジグパス (←) でモジュール D を読み込みながらモジュール C を折りたたむ.

動作の流れは $1 \to\ 2 \to\ 3 \to 2\ \to 3\ \to \ldots$ となる。

図 33 に伝達パートが再起的に動作している例を示した。

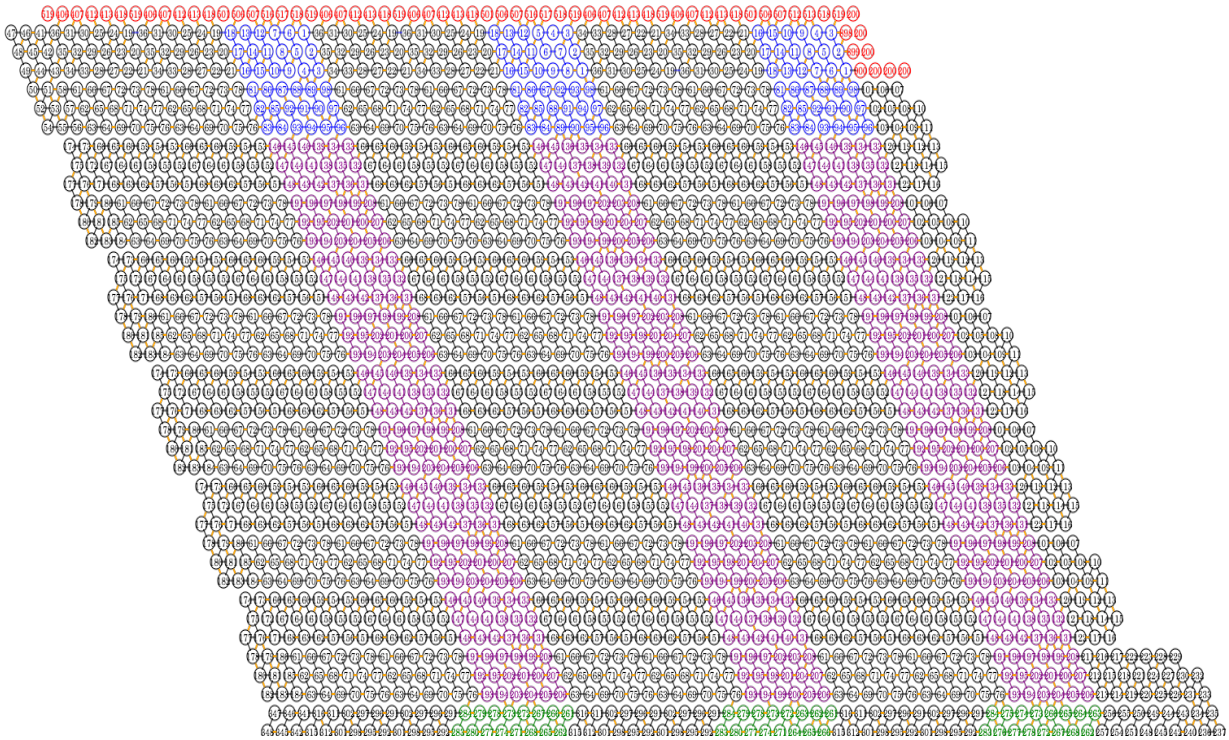


図 33: 伝達パートを再起的に動作した例

4.2.4 DFAO パート

前半部分-ジグパス (←, **モジュール** E)

前半部分のジグパスでは前節で述べたように、最初の 0 を探索しながら情報を次のパスに伝達している。モジュール E は 5 種類の構造で構成されており、使用するビードタイプは 261~284 としている。図 34 にモジュール E 内の構造と、底に現れるビーズのどの部分が情報を伝達しているかを示した。

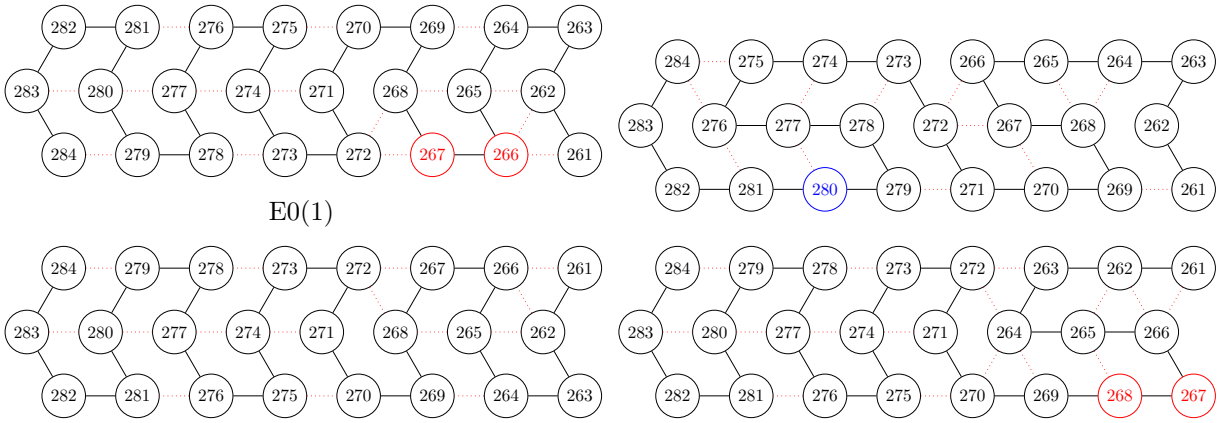


図 34: モジュール E の構造。赤色の破線は水素結合を表す。() 内の数字は下の列に出力する値を指す。青色のビーズは 4 種類の構図の中で最初の 0 を表す時、赤色のビーズは 1 を表す時のみ現れる。

モジュールの役目は最初に現れる 0 の探索だが、その動作はカウンターの半加算器と似ている。最初の折りたたみのスタートの位置は下となり、最初の 0 を見つけると終了の位置は上となる。それ以降は伝達のみを行う。ただし、モジュール E はモジュール A と違い、後半ビーズ 12 個分の折りたたみで最初の 0 を区別して折りたたまれる。15 番目のビーズは内部では 10 番目のビーズとのみ結合を持つが、E1 の構造のみ 10 番目のビーズは 15 番目のビーズの探索範囲外に折りたたまる。そのため E1 のみ後半部分の折りたたみ方が異なる構造となる。

前半部分-ザグパス (→, モジュール F)

前節で述べたように前半部分のザグパスでは “0”, “最初の 0”, “1”、の 3 種類の情報を伝達している。モジュール F は前半ビーズ 12 個分の折りたたみ方で 0 か最初の 0 を区別し、後半ビーズ 12 個分の折りたたみ方で 0 か 1 を区別している。また、使用しているビードタイプは 401~424 としている。図 35 にモジュール F 内の構造と底に現れるビーズの情報を伝達している部分を示した。

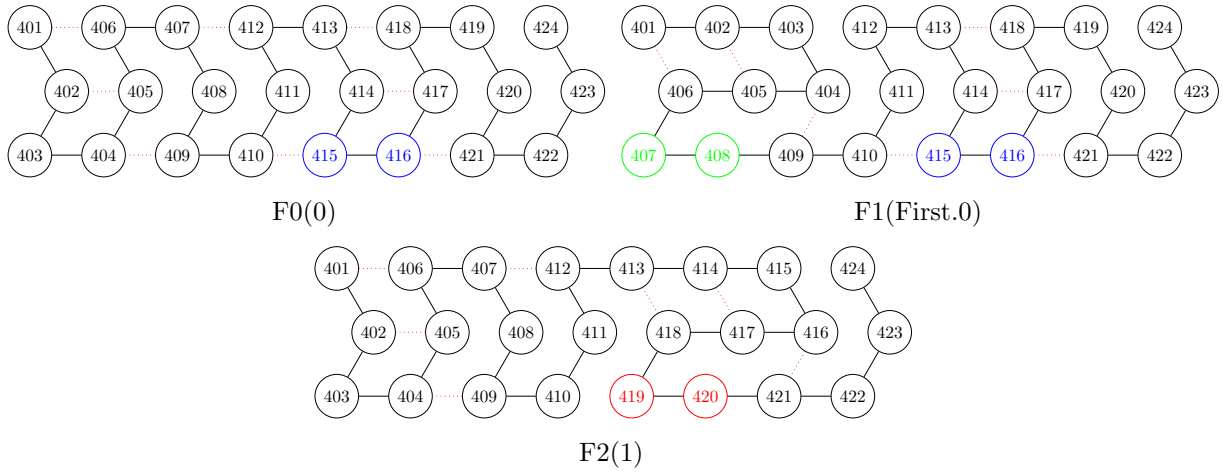


図 35: モジュール F の構造。赤色の破線は水素結合を表す。() 内の数字は下の列に出力する値を指す。青色のビーズは 2 種類の構造の中で 0 を表す時、赤色のビーズは 1 を表す時、緑色のビーズは最初の 0 を表す時のみ現れる。

モジュール F の 3 個目までのビーズは E1 の 19, 20 個目のビーズと水素結合を持つ。そのため E1 が上に存在するときのみ、ビーズが引き寄せられ F1 へと折りたたまれる。3 つの *conformation* のうち、F1 のみ前半部分が他 2 つと異なる形に折りたたまれる、つまり図 35 の緑色のビーズが底に現れる。

後半部分-ジグパス (←, モジュール G)

後半部分のザグパスでは前節で述べたように最初の 0 の後に現れる数字の探索、つまり上記の F1 の次に現れる構造を探索している。モジュール G で使用しているビードタイプは 481~504 としており、図 36 にモジュール G 内の構造と底に現れるビーズのどの部分が情報を伝達しているかを示した。

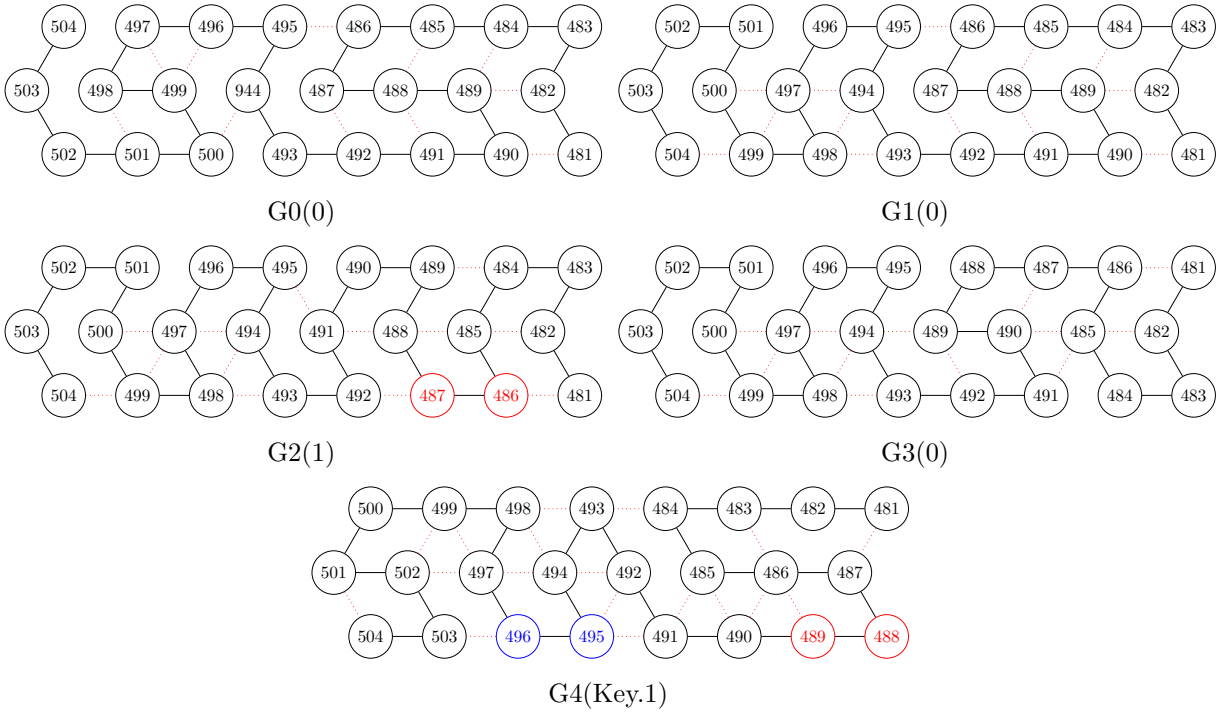


図 36: モジュール G の構造。赤色の破線は水素結合を表す。青色のビーズは 2 種類の構造の中で出力 0 を表す時、赤色のビーズは 1 を表す時のみ現れる。

このパスでは、前節で述べたモジュール G の動作を行うために、構造が 5 種類必要となっている。ただしスタートの位置が上となるのは、G0 の次に折りたたまれる構造のみなので、1 つの列には G3 と G4 のどちらか 1 つしか現れない。G0、G1 は前半ビーズ 12 個分の折りたたみ方が共通となっている。G0 のみ後半 12 個分の折りたたみによって終了の位置が上となっている。この違いは、上記のように F1 の前半 12 個分の折りたたみによって現れる 7, 8 個目のビーズと水素結合を持つことによって、行なわれている。スタートの位置が上となる G3, G4 の前半 12 個分の折りたたみは 3 個目までのビーズと F2 の底に現れる 19, 20 個目のビーズとの間にある水素結合によって 2 つに分かれる。この時 G4 のみ終了の位置が上となる。後半ビーズ 12 個分の折りたたみ方は、前半 12 個分の終了の位置によって区別される。

後半部分-ザグパス (→, **モジュール** H, I)

モジュール H で使用するビードタイプは 591~614 としており、5 種類の構造と底に現れるビーズのどの部分が情報を伝達しているかを図 37 に示した。

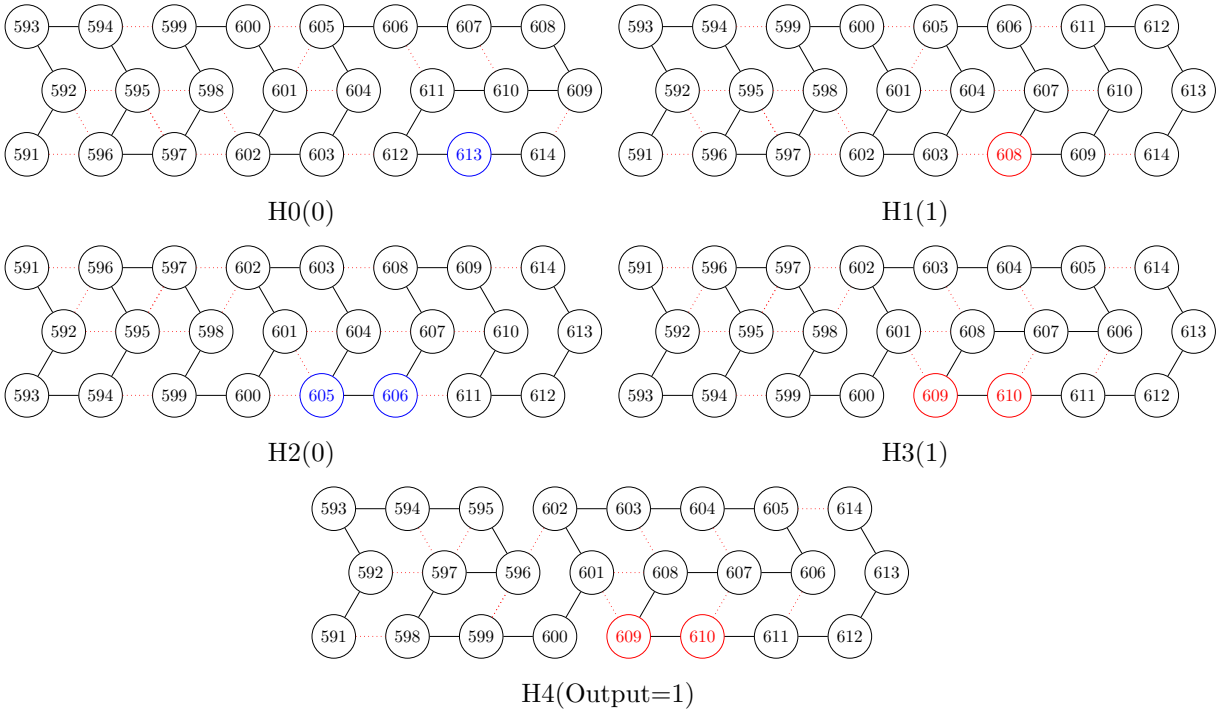


図 37: モジュール H の構造。赤色の破線は水素結合を表す。() 内の数字は下の列に出力する値を指す。Key.1 は最初の 0 の後の 1 としている。青色のビーズは 2 種類の構造の中で Key.1 を表す時、赤色のビーズは 1 を表す時のみ現れる。

前節で述べたように、モジュール H は情報の伝達と DFAO の出力を行っている。H0∼H3 の 4 つの構造において、スタートと終了の位置は変わらない。つまり、DFAO の出力は変わらずただ情報のみを伝達している。上に G4 が存在する時のみ、モジュール H の 5 個目のビーズと G4 の 16、23 個目のビーズが水素結合を持ち、H4 へと折りたたまれる。この時のみ、スタートの位置が下から終了の位置が上へと変わり、DFAO の出力が 0 となる。

モジュール I が使用するビードタイプは 641∼652 としている。図 38 にモジュール I 内の構造と底に現れるビーズのどの部分が情報を伝達しているかを示した。

図 38: モジュール I の構造。赤色の破線は水素結合を表す。() 内の数字は下の列に伝達する、DFAO の出力の値を指す。青色のビーズは 2 種類の構造の中で 0 を表す時、赤色のビーズは 1 を表す時のみ現れる。

モジュール I ではスタートの位置によって、I0 と I1 へと折りたたみ方を変える。3 個目までのビーズは同じ

形に折りたたまれるが、4~8 個目までのビーズは異なる形に折りたたまれる。スタートの位置が下だった場合のみ、5 個目のビーズが上に存在するビーズと水素結合を持つことができる。

4.2.5 方向転換パートへの伝達 (モジュール J, K)

モジュール J, K における使用するビードタイプは 661~672 と 681~692 となっている。図 39, 40 にモジュール J, K 内の構造と底に現れるビーズのどの部分が情報を伝達しているかを示した。2 つのモジュール内の構造の形は同じだが、ビードタイプは異なるので次の列のモジュールは問題なく区別することができる。

図 39: モジュール J の構造。赤色の破線は水素結合を表す。() 内の数字は下の列に出力する値を指す。青色のビーズは 2 種類の構造の中で 0 を表す時、赤色のビーズは 1 を表す時のみ現れる。

図 40: モジュール K の構造。() 内の数字は下の列に出力する値を指す。赤色の破線は水素結合を表す。青色のビーズは 2 種類の構造の中で 0 を表す時、赤色のビーズは 1 を表す時のみ現れる。

4.3 入力と動作の流れ

例えば入力を 2、つまり 010 としてこれらのモジュールを実際に稼働すると、カウンターパートで入力は $010 + 1 = 011$ となり, DFAO の出力は $P_n = 1$ となるはずである。図 41 にシミュレーター上で実際に動かした図を示した。

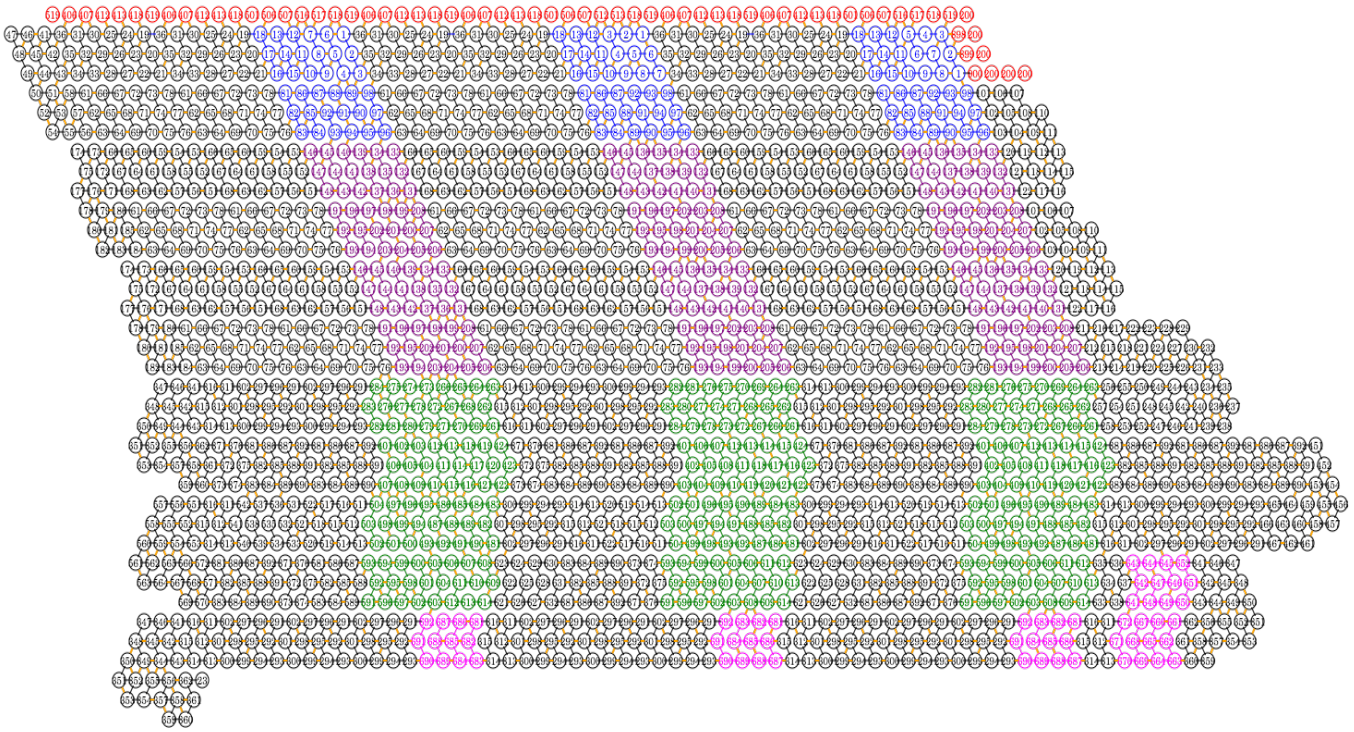


図 41: 入力 n=2 の時の動作例

5　考察 (今後の課題)

今回、1 つの構造の幅がとても大きいものになってしまった。これは下の列に伝達する情報が多くなればなるほど、モジュールの役割は複雑となってしまうからである。伝達方法として、スタートの位置と上に存在する構造のビードタイプの 2 つを利用したが、伝達の形を他にも作ることができれば、複雑な挙動を求められるモジュールも幅が小さく、少数のビードタイプで作ることが可能だろう。例えば [8] では 60 個のビードタイプを折りたたむことでバイナリーカウンターをターン部分等含めて全てサイクリック的に稼働することができる。もっと言えば生物の身体の中の DNA, RNA は 4 種類の分子で成り立っており、将来的には実際のこれらの環境下を想定して Oritatami System を動かす必要がある。また、今回使用した DFA は都合良く二列で実装することができたが、全ての DFA がこのようにいくとは限らない。DFA は自分がどの状態に遷移しているのか常に把握している必要があるので、上記に述べたように今までとは異なるアプローチが必要となる。

6 謝辞

一年間指導をいただいた関先生、および関先生を訪ね研究室に来られた多くの方々、特に Hwee Kim 氏から多くのアイディアやアドバイスをいただきました。ここに、感謝の意を表します。また、ともに研究を進めた生方氏にも感謝致します。

参考文献

[1] Bruce Alberts, Dennis Bray, Karen Hopkin, Alexander Johnson, Julian Lewis, Martin Raff, Keith Roberts, and Peter Walter. *Essential cell biology*. Garland Science, 2013.

[2] A Xayaphoummine, V Viasnoff, S Harlepp, and H Isambert. Encoding folding paths of rna switches. *Nucleic acids research*, Vol. 35, No. 2, pp. 614–622, 2007.

[3] Makoto Ota and Shinnosuke Seki. Rule set design problems for oritatami system. *Theoretical Computer Science*, 2016.

[4] Kirsten L Frieda and Steven M Block. Direct observation of cotranscriptional folding in an adenine riboswitch. *Science*, Vol. 338, No. 6105, pp. 397–400, 2012.

[5] C Geary, Paul WK Rothemund, and Ebbe S Andersen. A single-stranded architecture for cotranscriptional folding of rna nanostructures. *Science*, 2014.

[6] C Geary, Pierre-Étienne Meunier, Nicolas Schabanel, and S Seki. Efficient universal computation by greedy molecular folding. *arXiv preprint arXiv:1508.00510*, 2015.

[7] Jean-Paul Allouche and Jeffrey Shallit. *Automatic Sequences: Theory, Applications, Generalizations*. Cambridge University Press, 2003.

[8] Cody Geary, Pierre-Étienne Meunier, Nicolas Schabanel, and Shinnosuke Seki. Programming biomolecules that fold greedily during transcription. In *LIPIcs-Leibniz International Proceedings in Informatics*, Vol. 58, pp. 43:1–43:14. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[9] Makoto Ota. Oritatami system の設計問題に関する研究. Technical report, The University of Electro-Communications, 2016.