Oritatami Systemによる
無限バイナリカウンタの実装

丸山晃平 *

2020年 1月 20日

*情報・ネットワーク工学専攻 1831144 関 研究室

Contents

1　はじめに

私たちの体は、元は小さな受精卵から細胞分裂を繰り返し成長してきた。 それは外部から手を加えられることなく(自らが増殖し、自らを制御することで成長してゆく)。 その成長の方法を定めているのが遺伝情報であり、私たちの細胞一つ一つの核内に存在するDNAに記録されている。

DNAの遺伝情報は、その一部が体を形成するタンパク質を作るための設計図となっている。 DNAが細胞核の中にあるのに対して、タンパク質の合成は細胞核外のリボソームで行われる。 遺伝情報を細胞核の中から外へ伝えるために一時媒体として、「mRNA」と言うRNAが合成される。 RNAはDNAの情報を基にRNAポリメラーゼという酵素によって生成される。この過程を「転写」と呼ぶ。 RNAはDNAと似た構造をしているが、安定性が低く反応しやすい特徴を持つ。

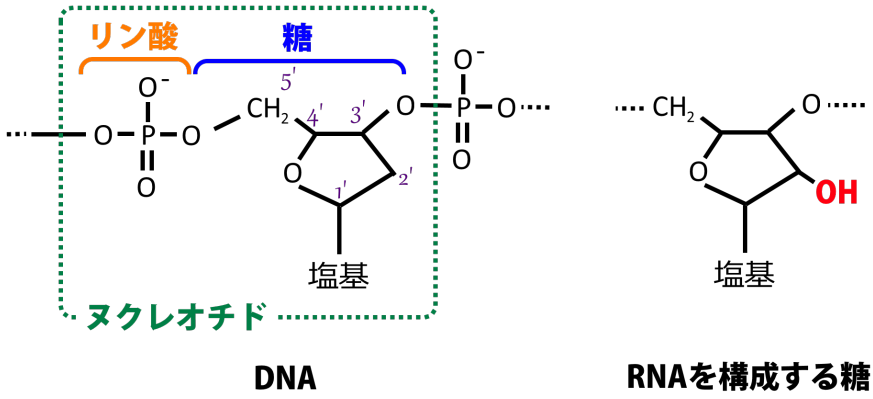


Figure 1: (左)DNAの構造式。DNAは図の緑枠で囲われているヌクレオチドが鎖のように連なってできている。(右)RNAの構造式。RNAの糖は2'末端にヒドロキシ基を持つ。

Figure 2: (左)二本のDNAがお互いに対になる塩基によって水素結合を形成している。 (右)転写されたRNAはDNAと同じ遺伝情報を持っている。ただし塩基TはUへ置き換わる。

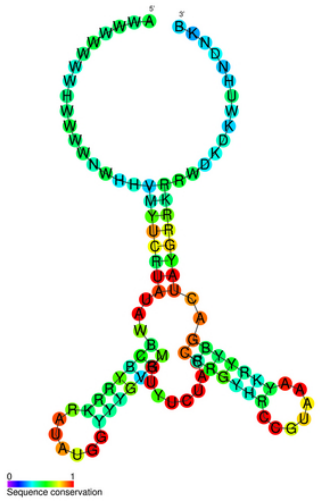


Figure 3: 枯草菌のRNAが形成するリボスイッチの模式図。RNAはその内部で水素結合をすることによって遺伝子発現を調節する機能も持つ。

DNAは図1の左図に置いて示されるヌクレオチドが鎖のように連った分子である。ヌクレオチドはリン酸と糖、そして塩基から構成されるが、これはRNAも同じである。 DNAとRNAは遺伝情報を塩基の配列として保持している。 DNAとRNAともに塩基は4種類のものを持ち、DNAとRNAでそれぞれ(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]。 細胞核内でDNAは同じ遺伝情報を持つ二本のDNAが水素結合をすることによって、冗長性を持っている。 塩基の水素結合は4種類が2対ずつになっているため、ここで言う同じ遺伝情報を持つ二本のDNAとは図 2のように全ての塩基が対になっているようなものを指す。 RNAはDNAの遺伝情報がコピーされいるが、DNAを構成する糖は安定なデオキシリボースであるのに対し、RNAは図1の右図のように2'末端にヒドロキシ基を持つリボースで構成される。その影響でRNAはDNAと比べ不安定となる。

しかし一方で、RNAは様々な反応が進行しやすく、高次構造を形成し、酵素活性を示す。 例えばmRNAはタンパク質設計図のコピーとして使われるだけでなく、内部で水素結合をし折りたたまれることにより図 3で表されるような構造を作り遺伝子発現を調節する。 このような構造は、RNAがエネルギー的に安定な状態になるように水素結合した結果として得られる。 RNAのヌクレオチドが転写される速度は、それが折りたたまれる速度よりも遅い[4]。 そのため、転写と折りたたみはほぼ同時

に進行する。この現象は「co-transcriptional folding」と呼ばれる。

2 oritatami system

2.1 定義

Σを有限の文字集合とし、その元は塩基の種類を表す。例えば実際のRNAについては$\Sigma = \{A, U, G, C\}$となる。 Σ^*とΣ^ωはそれぞれ、有限長の塩基配列、無限長の塩基配列を表し、またλ は空の塩基配列を表す。 $w = b_1 b_2 \cdots b_n \in \Sigma^*$は長さ$n$の塩基配列を表す。 ここで$n$は正数で、$b_1, \ldots, b_n \in \Sigma$である。 また、$w$の長さは$|w|$で表し、$|w| = n$となる。 i、j $(1 \leq i \leq j \leq n)$について、$w[i..j]$は部分配列$b_i b_{i+1} \cdots b_{j-1} b_j$を表し、また$w[i..i]$の場合は単に$w[i]$と表す。

oritatami system はRNAが転写された際に起こる$cotranscriptional folding$という現象を数理モデル化したものであり、塩基配列$w$を三角格子状の平面グラフ$\mathbb{T} = (V, E)$の上に折りたたまれる。 グラフ上の頂点$p \in V$について、$\bigcirc_p^d$は半径$d$、原点$p$の正六角形に含まれる頂点の集合を表す。なお$\bigcirc_p^d$には$3d(d+1)+1$個の頂点が含まれる。 $\mathbb{T}$上の有向パスPを$P = p_1 p_2 \cdots p_n$とすると、$p_1, p_2, \ldots, p_n \in V$すなわち、すべての$i (1 \leq i < n)$について$\{p_i, p_{i+1}\} \in E$でありすべての頂点は互いに異なる。 なお、パスの$i$番目の頂点は単に$P[i]$と表す。 今、一本鎖のRNAがoritatami systemに折りたたまれることによってできる「構造」をCと定義する。Cは(P, w, H)の三つの組で表され、それぞれのパラメータについてPは$\mathbb{T}$上の有向パス、$w \in \Sigma^*$はPと同じ長さの塩基配列、Hは$H \subseteq \left\{\{i, j\} \;\middle|\; 1 \leq i, i+2 \leq j, \{P[i], P[j]\} \in E\right\}$であり、塩基間の水素結合の集合を表す。 なお、$H$についての条件$i+2 \leq j$は隣り合う頂点同士が近すぎて物理的に水素結合を結べないという制約を表している。 塩基配列wはパスPに沿って配置されるため、$w[i]$の配置先は$P[i]$となる。 また、i番目の塩基とj番目の塩基は水素結合を結ぶことは$\{i, j\}$がHに含まれていることと同値である。 構造Cの長さは塩基配列wともパスPとも同じになる。

$R \subseteq \Sigma \times \Sigma$をルールセットと呼び、この$\Sigma$は対称関係になっている。すなわち、任意の$a, b \in \Sigma$について、$(a, b) \in R$ ならば$(b, a) \in R$である。整数$\alpha \geq 1$は*arity*と呼び、Cが*arity*αであるとき、もしある塩基がαの水素結合をしているなら、それ以上その塩基が水素結合をすることはできない。 $\mathcal{C}_{\leq\alpha}(\Sigma)$は*arity*が多くとも$\alpha$である$\Sigma$上の構造のことを指し、文脈上で$\Sigma$が明らかな場合は省略して表記する。

oritatami systemは転写物を伸長させ、それを折りたたんで行くことで構造を形成して行く。 ここで、Rをルールセット、$C_1 = (P, w, H)$をRが適用された構造とする。 また、構造C_2はC_1に塩基bが伸長されたもの

とし、この「伸長」を$C_1 \xrightarrow{R}_b C_2$と表記する。 なお$C_2$は、$C_1$のパス$P$に含まれていない頂点$p \in V$と、形成している$b$との水素結合を表す集合$H' \subseteq \left\{\{i, |w|+1\} \;\middle|\; 1 \leq i < |w|, \{P[i], p\} \in E, (w[i], b) \in R\right\}$を用いて、$C_2 = (Pp, wb, H \cup H')$である ($H'$は空集合である可能性もある)。 そのため$C_2$にも$R$が適用されていることとなる。 この操作を再帰的に繰り返すことで構造が形成されて行く。 ある有限長の配列$w \in \Sigma^*$と塩基$b \in \Sigma$を用いて、構造C_1が配列wbによって構造C_2に伸長されることを$C_1 \xrightarrow{R}{}^*_{wb} C_2$と表し、これは$C_1 \xrightarrow{R}{}^*_{w} C'$と$C' \xrightarrow{R}_b C_2$を両方満たしている。 なお任意の構造$C$に対して$\lambda$を伸長しても、$C \xrightarrow{R}{}^*_{\lambda} C$である。

oritatami system Ξは、上記で説明したΣとR、また以下に示される4つのパラメータを組み合わせた$(\Sigma, R, \delta, \alpha, \sigma, w)$によって表される。

- *delay*と呼ばれる自然数δ 。
- *arity*と呼ばれる自然数α。
- Rが適用されている初期構造$\sigma \in C_{\leq\alpha}(\Sigma)$。これをシードと呼ぶ。
- 転写物$w \in \Sigma^* \cup \Sigma^\omega$。これの塩基を一つ折りたたむには、後続の$\delta-1$の塩基も含めたまだ「構造」に含まれていない転写物と既に折りたたまれている構造との間でエネルギーを最小になるような塩基の位置を探しそこに固定するという手順を踏む。

構造$C = (P, w, H)$の持つエネルギーは$\Delta G(C)$で表され、$-|H|$と定義する。 つまり、構造内での水素結合の数が多いほど安定する。 ここで、C_iのことをσから$w[1..i]$を伸長することによって形成された構造とする。 その次に転写される$w[i+1]$をoritatami systemΞによってC_iからC_{i+1}へ伸長可能である条件を以下で表す。

$$C_{i+1} \in \underset{\substack{C \in \mathcal{C}_{\leq\alpha}\, s.t. \\ C_i \xrightarrow{R}_{w[i+1]} C}}{\arg\min} \; \min\left\{\Delta G(C') \;\middle|\; C \xrightarrow{R}{}^*_{w[i+2\ldots i+k]} C', k \leq \delta, C' \in \mathcal{C}_{\leq\alpha}\right\}. \tag{1}$$

$w[i+1]$が転写され構造が伸長されることを、$w[i+1]$が固定されると呼ぶ。この、塩基が固定されて行く様子を動画https://www.dailymotion.com/video/x3cdj35にて閲覧することができる。 この動画で動作しているのはGearyらによって実装された$\delta = 3$で動くチューリング完全なoritatami system[2] である。 伸長可能な条件を満たさなくなったとき、oritatami systemは停止し、そこまでで形成された構造が最終構造となる。

2.2 動作例

oritatami system において、転写物がどのように折りたたまれいくかを説明するために以下のようなoritatami system Ξを考える。

$$
\begin{aligned}
\Sigma &= \{N, B, 1, 2, ..., 9\} \\
R &= \{(1,6), (2,5), (2,6), (3,B), (4,9), (7,B)\} \\
w &= ①-②-③-④-⑤-⑥-⑦-⑧-⑨ \\
\delta &= 3
\end{aligned}
$$

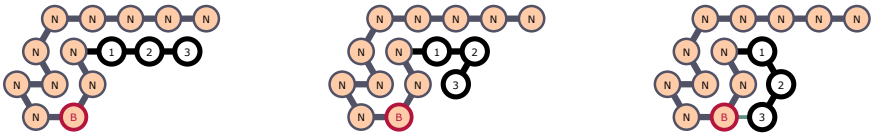


Figure 4: -①-②-③の探索

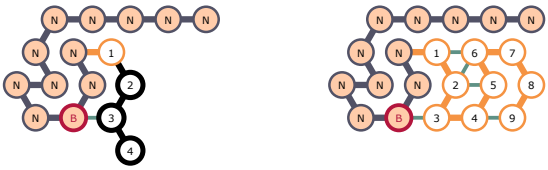


Figure 5: (左)①が固定され、-②-③-④の探索が始まる。(右)最終的に出力される構造。

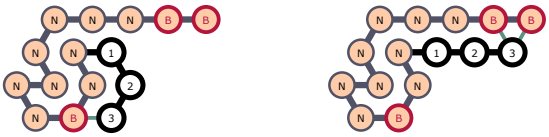


Figure 6: -①-②-③の探索 (シードの一部が変更されている)

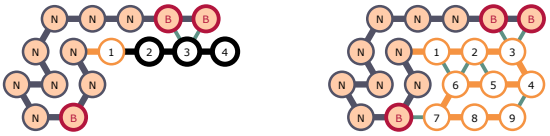


Figure 7: (左)①が固定され、-②-③-④の探索が始まる。(右)最終的に出力される構造。

Figure 8: -①-②-③の探索 (シードの一部が変更されている)

最初に図 4をみて見ると初期構造であるシードはⓃとⒷで表されていて、①の固定すべき場所を探索している。 今、このシステムは$\delta = 3$であるため、-①-②-③の長さ3の部分配列において、最も水素結合が多く結べる位置を探す。 すると$R \in (3, B)$より、図 4のうち一番左で1つ結合が結べる。 これが結合数最大となり、①が固定される(図 5 左)。 そのまま②、③と固定されていき、最終的な構造は図 5の右図のようになる。

次に図 6をみて見ると、シードの一部がⓃからⒷに置き換わっている。 この状態で-①-②-③の探索を始めると、先ほどのパス(図 6 左)よりも右図のパスの方が結合数が2と多いため右図のパスに決定される。 しかしよく観察すると、どちらも①の位置は同じであるため固定位置は先ほどの例と変わらない。 ただ、②の位置は変化するため(図 7)、最終的な構造は図 7の右図の構造となる。

このように、これから折りたたまれる転写物の周囲において、存在する塩基が変化することでその転写物の折りたたまれる方が変化する。 折りたたまれた転写物は、それ以降に転写された塩基と結合する可能性がある。 その際にこの折りたたまれ方の変化が、その後の転写物の折りたたみに影響し伝搬されいく。

最後に図 8をみて見る。 すると、-①-②-③の探索において図 8の左右のどちらのパスも結合数1で最大となる。 この場合、左右どちらも①の

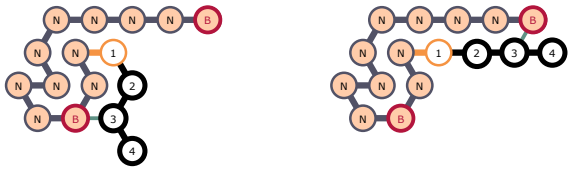


Figure 9: ①が固定され、-②-③-④の探索が始まる。1つの水素結合が最大であり、それが図の左右の２通りある。しかし次に固定される②の位置が異なっているため固定できない。

位置が同じであるため①が固定される。 次に-②-③-④の探索を考えると図 9の左右どちらも結合数1で最大である。 しかしこの時②の位置は左右異なるため一意に固定できない。 そうなるとシステムは構造を決定できずに停止する。

「周囲の環境」の説明 「グライダー」の説明

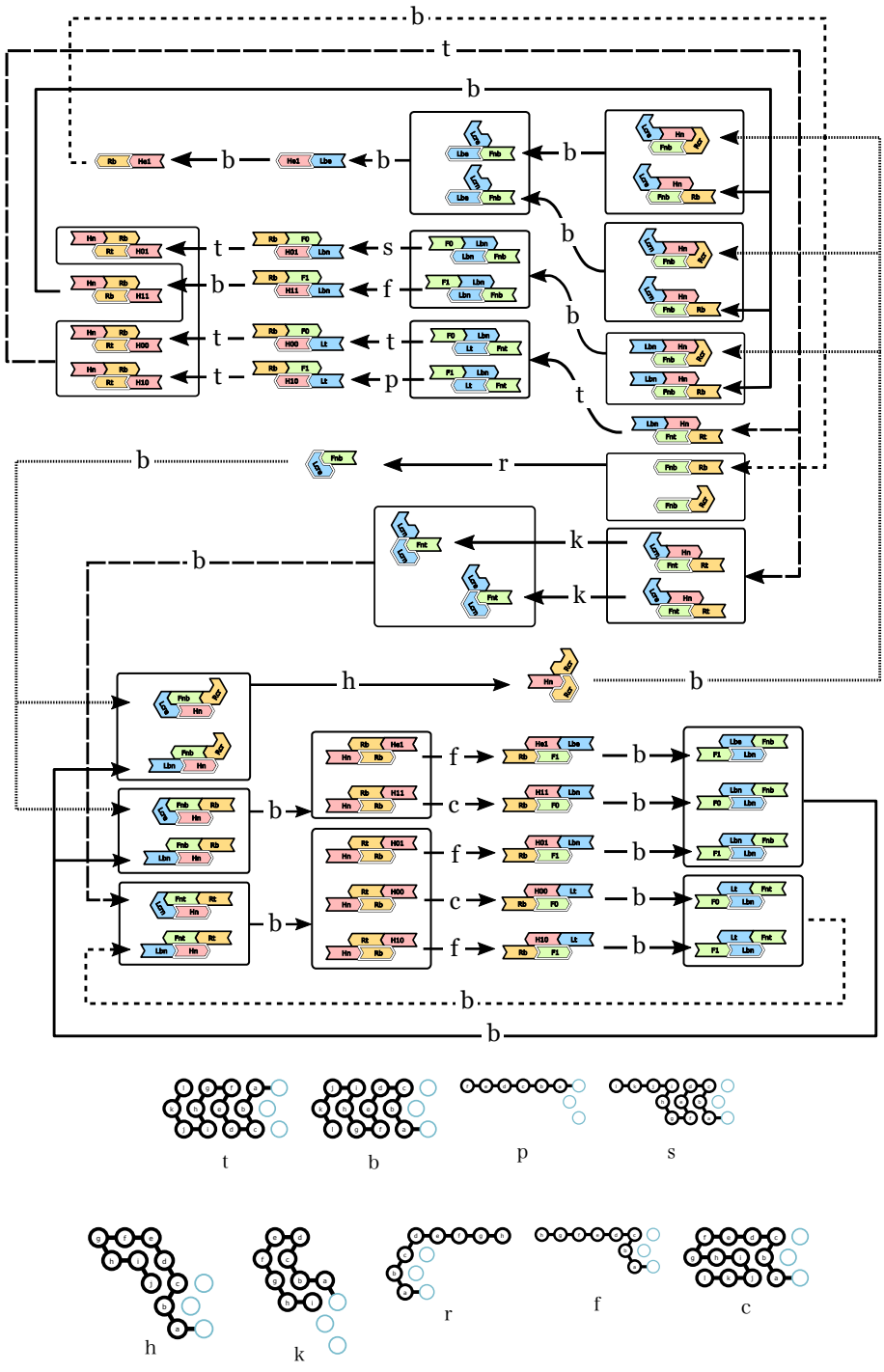


Figure 10: 無限バイナリカウンタのブリックオートマトン

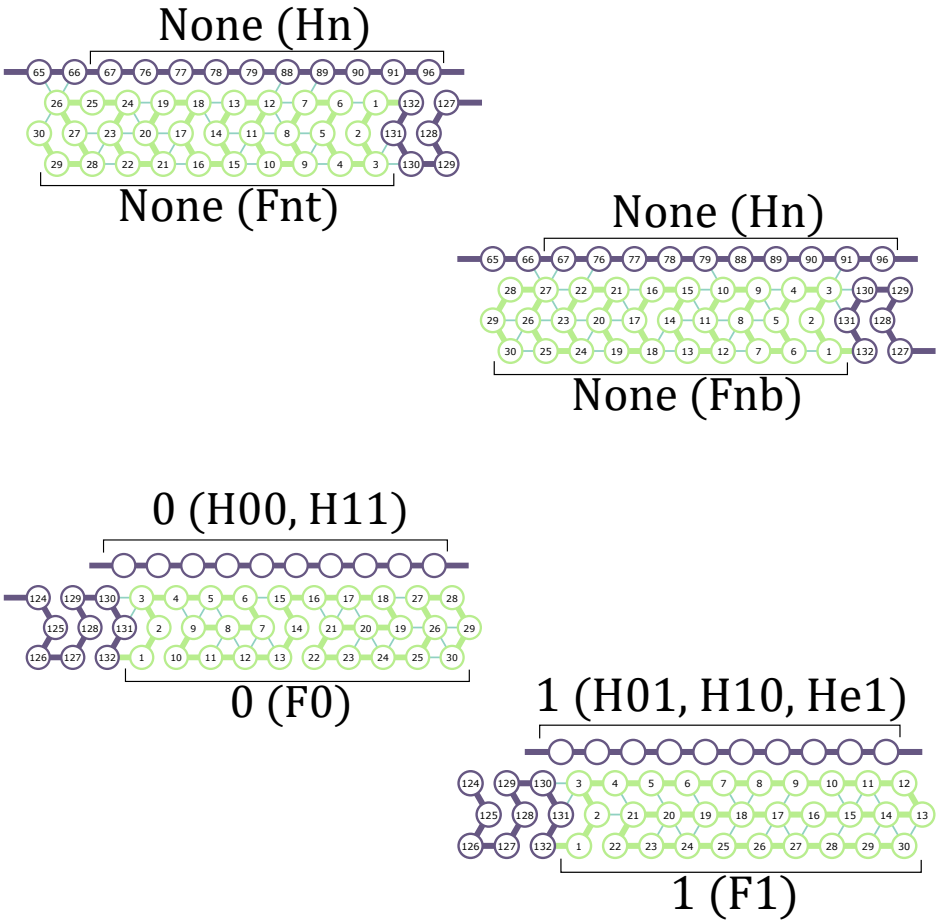


Figure 11: フォーマットモジュール(F)のブリックは全部で4種類存在する。上二つのFntとFnbはジグでのみ現れ、下二つのF0とF1はザグにのみ現れる。

3　無限バイナリカウンタの実装

この章では、oritatami system に実装した無限バイナリカウンタについて説明する。 ここで言う無限バイナリカウンタとは、あるビット幅でカウントアップしていたカウンタが桁上がりのためにオーバーフローした際に、そのビット幅を拡張することでカウントアップを無限に続けていくようなバイナリカウントを指す。

3.1　挙動の概要

Geary らは有限バイナリカウンタを oritatami system に実装することに成功している[3]。 今回実装した無限バイナリカウンタも、基本的な設計方針はこの有限カウンタと同じである。 具体的には、どちらのカウンタの転写物もジグザグ構造に折りたたまれる。 この構造は図において、まずは左方向へ進み、ある段階で折り返して右方向へ進みまた折り返すというように表され、下方向へ段重ねで積み重なっていく。 またジグザク構造のうち、ジグ（右から左への伸長）では現在のカウントが1つカウントアップされ、次のザグ（左から右への伸長）ではジグでカウントアップされた値をフォーマットし、次のジグのためにその値を下方向へコピーするといった挙動も有限カウンタと似ている。 ここで無限カウンタと有限カウンタで異なるのはオーバーフローに遭遇した時の挙動である。無限カウンタはビット幅を1つ拡張することによってカウントアップを続行する。

実装した無限カウンタの転写物は同じ塩基配列を周期的に繰り返す。その1周期分の塩基配列は、1-2-3- ⋯ -132であり、更にこれは「モジュール」と呼ばれる以下のような4つの部分配列に分けられる。

- 1–30: 「フォーマットモジュール」もしくは「F」と呼ぶ
- 31–66: 「左ターンモジュール」もしくは「L」と呼ぶ
- 67–96: 「半加算器モジュール」もしくは「H」と呼ぶ
- 97–132: 「右ターンモジュール」もしくは「R」と呼ぶ

無限カウンタにおける転写物の配列は、この4つのモジュールを用いて$(FLHR)^*$と表すことができる。 また、これらのモジュールは図中ではそれぞれ緑、青、赤、黄色に色分けする。 モジュールはその周囲の環境ごとにそれぞれ特定の平面構造に折りたたまれるように設計されていて、この平面構造のことを「ブリック」と呼ぶ。すなわち、このブリックを「出力」とみなすと、周囲の環境は「入力」であり、oritatami

system がモジュールをブリックに折りたたむ過程は情報の「処理」となる。また、出力として扱われたブリックは、別のモジュールにとっての周囲の環境の一部となることによって、情報が伝搬して行く。

例えばフォーマットモジュールについて、そのブリックを見てみる。フォーマットモジュールは図 11で示される4種類の環境に遭遇し、それぞれの環境で異なるブリックに折りたたまれる。 これにおいてモジュールがそれぞれのブリックに折りたたまれるのは、式 (1)に従って転写配列内の特定の塩基同士が結合するように設計しているためである。

無限カウンタを実装したsystemは、各モジュールが、設計段階で想定された環境にしか遭遇しないことが図 10の「ブリックオートマトン」によって保証されている。 この遷移図では状態として環境とブリックのペアが、状態間の遷移として、折りたたまれたブリックの前半部分の構造が用いられている。 また、この図で示されているブリックオートマトンは閉じている。 そのため、実装したsystemが正常に動作していることを保証するには、状態ごとの全てにおいて、モジュールが正しいブリックに折りたたまれていることを検証し、かつ遷移先もオートマトンに記述されたものと一致すれば良い。 動作確認は専用に開発したシミュレータに無限カウンタのoritatami systemを適用することで行われ、その結果はhttps://komaruyama.github.io/oritatami-infinit-counter/に掲載されている。

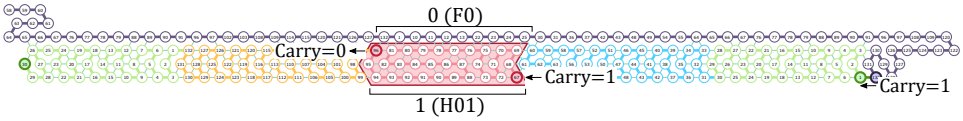


Figure 12: 一番最初のジグ構造。 *seed*には初期値としての0が式 (2)(ビット幅が1すなわち、$k = 1$)のフォーマットにしたがって記されている。また、ジグの始めにキャリーが与えられているので、このジグの中でカウントアップが行われる。 赤色で表された半加算器モジュールHは1を出力し、キャリーを解消している。 またこの1の出力は、正確に言うとザグ内で1と解釈され再フォーマットされるような塩基の並びを指す。

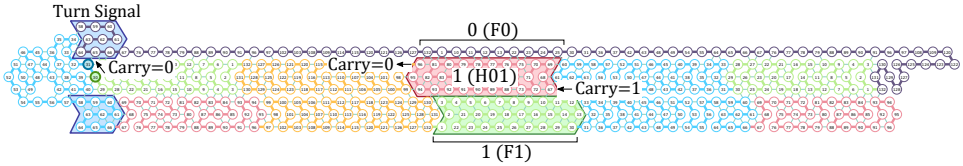


Figure 13: モジュールLが左ターンし、最初のジグが始まる様子。*seed*の左端にターンシグナルがあり、かつジグがキャリーなしで終わっているので、Lはターンシグナルと結合し折り返す。また、ターンしたLのブリックもまたターンシグナルを持つ。 ザグの中ではモジュールFはHの出力を読み、下方向へその値をコピーする。

3.2 シードへの初期カウント数の記述方法

初期カウント数がバイナリ表記で$b_{k-1}b_{k-2}\cdots b_1b_0$と表されるとき、シードは以下のように記述される。

$$64\text{–}65\text{–}66\text{–}\left(\prod_{i=k-1}^{0}\left(w_{Hn}w_{Rb}w_{Fb_i}w_{Lbn}\right)\right)w_{Hn} \tag{2}$$

ここで、

$$\begin{aligned}
w_{Hn} &= 67\text{–}76\text{–}77\text{–}78\text{–}79\text{–}88\text{–}89\text{–}90\text{–}91\text{–}96,\\
w_{Rb} &= 97\text{–}102\text{–}103\text{–}108\text{–}109\text{–}114\text{–}115\text{–}120\text{–}121\text{–}126\text{–}127\text{–}132,\\
w_{F0} &= 1\text{–}10\text{–}11\text{–}12\text{–}13\text{–}22\text{–}23\text{–}24\text{–}25\text{–}30,\\
w_{F1} &= 1\text{–}22\text{–}23\text{–}24\text{–}25\text{–}26\text{–}27\text{–}28\text{–}29\text{–}30,\\
w_{Lbn} &= 31\text{–}36\text{–}37\text{–}42\text{–}43\text{–}48\text{–}49\text{–}54\text{–}55\text{–}64\text{–}65\text{–}66
\end{aligned}$$

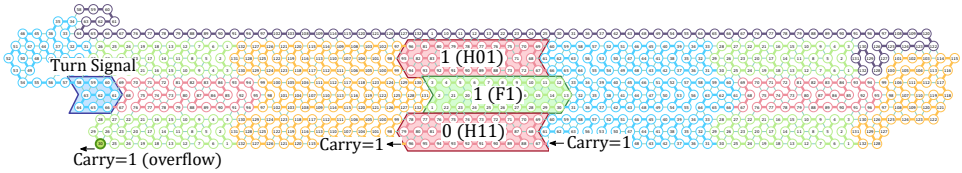


Figure 14: キャリーありのまま左端に到達した様子。 この次に転写されるLはジグの下側から転写が開始されるため、上のターンシグナルと結合するには距離が遠すぎる。そのため、Lはターンすることなく直進する。

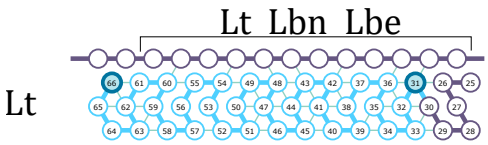
上記の配列は、モジュールH、R、F、LがブリックHn、Rb、Fb_i、Lbnに折りたたまれた時にそのブリックの下部に現れる配列であり、それぞれ図 16、17、11、15で確認できる。 例として、初期値$b_0 = 0$、ビット幅$k = 1$のシードは、図 12における紫色の箇所である。

3.3 ブリック単位での動作説明

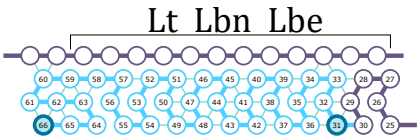
oritatami systemは最初に*seed*部分が折りたたまれた状態から始まる。今回実装したsystemでは、ジグ(↞)、左ターン(↪)、ザグ(↠)、右ターン(↩)の4つを周期的に繰り返す。 *seed*に記録された初期カウントがkビットである無限カウンタでは、最初のジグの*transcript*は$(FLHR)^kF$で表される。 ジグの中では、モジュールFとHのブリックはどちらも高さ3、幅10に折りたたまれ、またモジュールLとRのブリックはどちらも高さ3、幅12に折りたたまれる。 それゆえジグは高さ3の線形構造となる。 最初のジグの中で、i番目のモジュールHは、式 2に従ってフォーマットされたb_{i-1}のすぐ下から転写が開始し、そのフォーマット配列に応じたブリックに折りたたまれる。 つまり、そのHはb_{i-1}の値を読んでいることになる。 最初のジグが全て折りたたまれた後、その直後の左ターンモジュールLはターンシグナルのすぐ下に転写される。これによって高さ3のジグがその上側で終了した場合、このターンシグナル(特に58、63、64)とLの33と34が結合をし、Lが左ターン用のブリック(Lcre)に折りたたまれる(図 13)。 このブリックLcreは、更に次のジグ終了後にそのブリックの下に転写される左ターンモジュールのためにターンシグナルを持つ。

最初のジグでターンが終わった後に、続いて最初のザグの転写が開始される。 その部分配列は$(HRFL)^kH$で表され、各モジュールについてもジグ同様にモジュールFとHが高さ3、幅10、モジュールLとRが高さ3、幅12のブリックに折りたたまれる。 そのためザグもジグと同様に高さ3の線形構造となる。 このザグが最後まで折りたたまれた直後に転写されるRは、ターンシグナル125-124-123-122と結合することによってブリックRcrとなり右ターンする。 Rcrも次の右ターンのためにターンシグナルを持つ。 このジグとザグが折りたたまれることがカウントの値を一つカウントアップさせることに相当する。

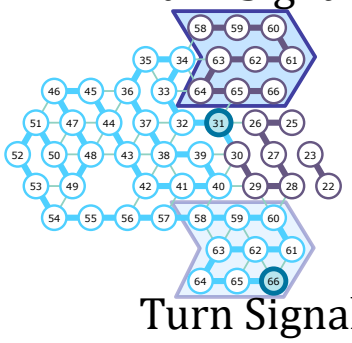
ここで図 12-14を見てみると、モジュールHとFのブリックが縦方向で交互に並んでいることがわかる。 その列の右からi番目がカウンターの$i-1$ビットに相当し下方向へカウントの値を伝搬している。



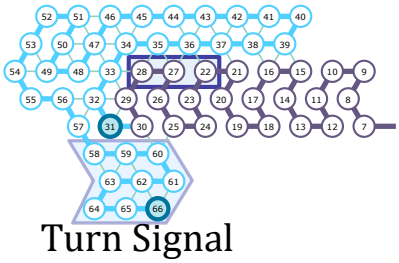
Lt



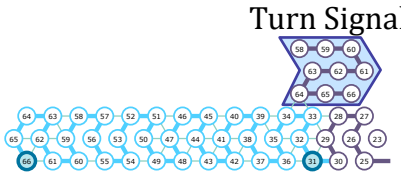
Lbn



Lcrn



Lcre



Lbe

Figure 15: 左ターンモジュール(L)のブリックは５種類あり、図の上から順にLt、Lbn、Lcrn、Lcre、Lbeとなっている。 ジグの中でLは、左端に到達するまでの間はLtかLbnに折りたたまれ、そのどちらになるかはLの折りたたみの開始地点に応じて決定される。 Lが左端に到達した時オーバーフローしていなければLcrn、もしオーバーフローしていた時はLbeに折りたたまれる。 一方でザグの中ではLはLbnにのみ折りたたまれる。

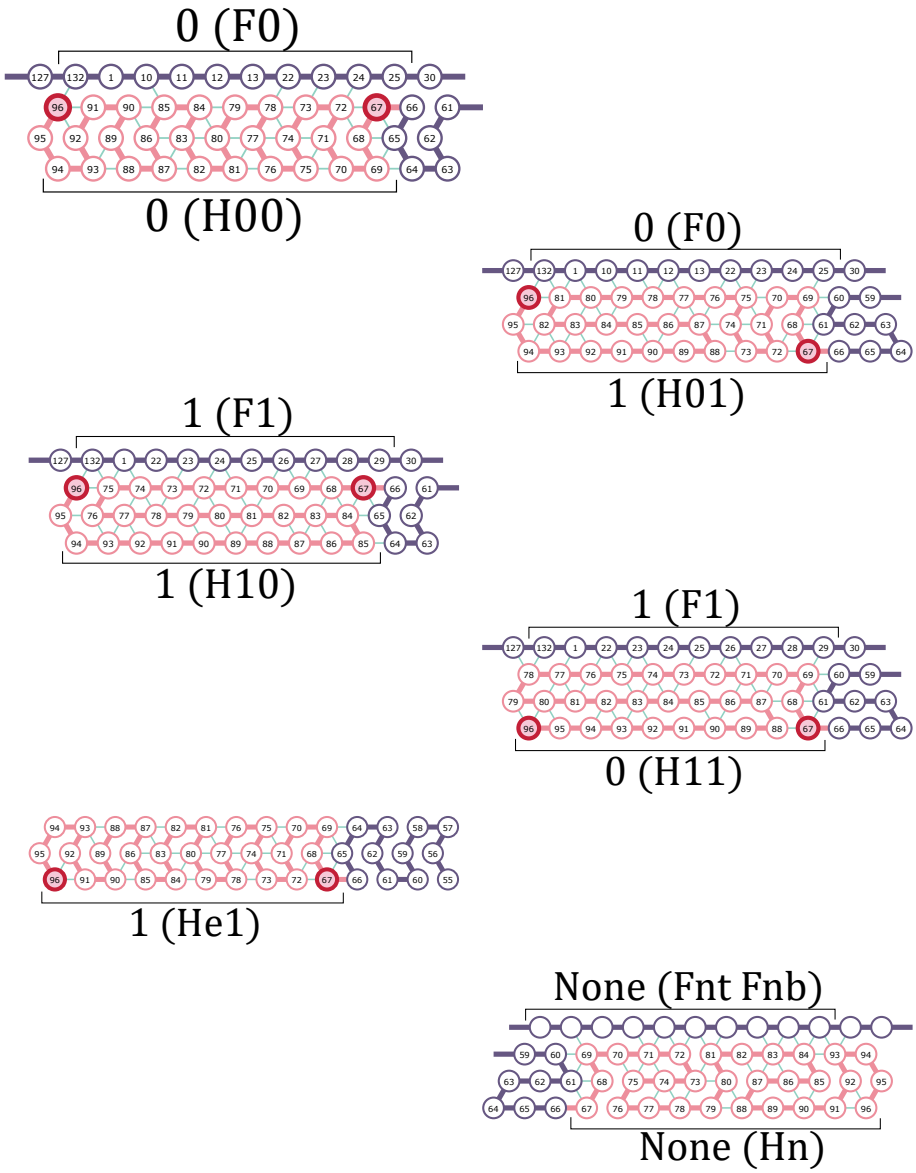


Figure 16: 半加算器モジュール(H)のブリックは 6 種類存在し、上から順にH00、H01、H11、H10、He1、Hnとなっている。 ザグの時、HはHnにのみ折りたたまれ、ジグの時はそれ以外のブリックに折りたたまれる。

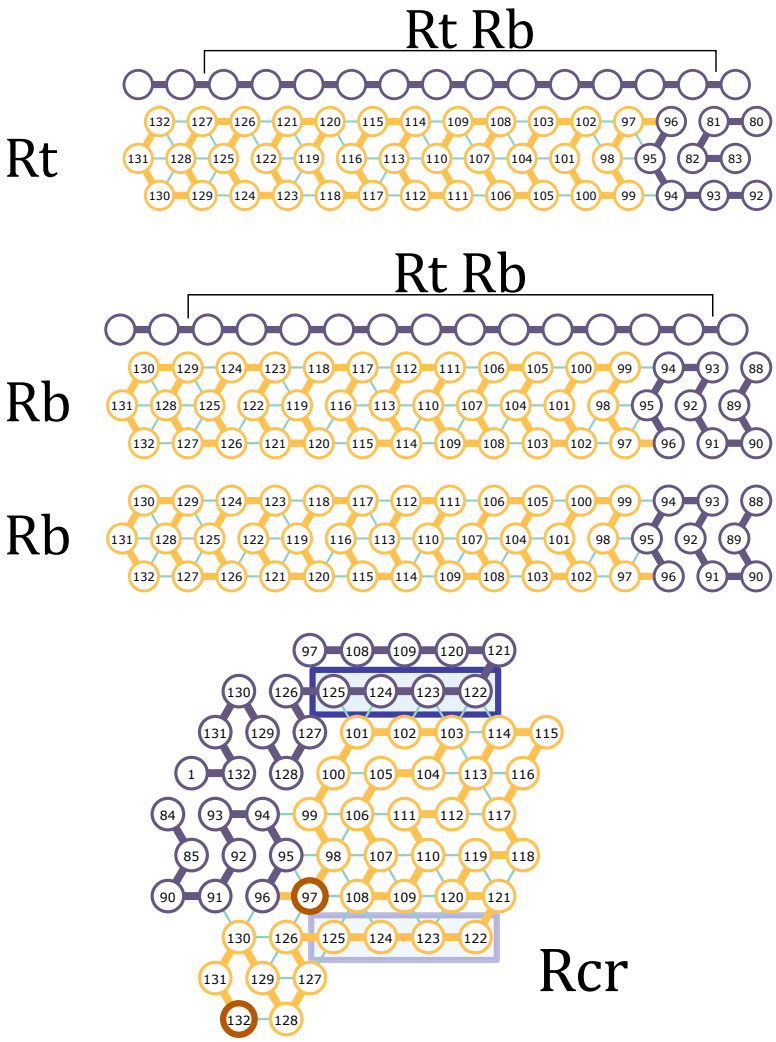


Figure 17: 右ターンモジュール(R)は３種類のブリックに折りたたまれる。図は上から順にRt、Rb(上に構造あり)、Rb(折りたたまれた構造が上に存在しない)、Rcrである。 ジグの時、RはRtかRbに折りたたまれ、どちらになるかはRの開始地点に依存する。またここではRbに関して、２つの内どちらの構造も取り得る。 ザグの時は、右端に到達するまでの間いつもRはRb(上に構造あり)のブリックに折りたたまれる。 右端にたどり着いたらRcrへと折りたたまれる。

3.4 カウントアップの方法

ジグでは、モジュールHが半加算器に相当する役割を果たし、あるHの出力したキャリーは他のモジュール(F、L、R)によって1つ上位ビットのモジュールHへと伝えられる。 無限カウンタ内でキャリーの有無はモジュールそれぞれの転写開始位置によって表現される。 ジグ内で、モジュールF、L、Rはそれぞれ2種類のブリックずつに折りたたまれ、FはFntかFnb、LはLtかLbn、RはRtかRbになりうる。 これらは図 11、15、17を見ると、各モジュールごと一方のブリックが上から転写が始まり上で終わっているならば、もう一方のブリックは下で始まり下で終わっている。 つまり、開始位置(終了位置)が異なる2対のブリックを用いてそれぞれのモジュールはキャリーを伝えている。

ジグはRcrとシードの開始位置が下側であるのでキャリーが与えられた状態で折りたたみが開始し、カウントアップをしている。 カウントがオーバーフローするまでの間、モジュールHが遭遇し得る環境は、上からの入力が0か1、キャリーが有るか無いかと言う4種類の環境である。 ここで言う上からの入力は、0がw_{F0}として、1がw_{F1}として表現されている。 そして、この4種類の環境に応じてモジュールHはH00、H01、H10、H11の4つのブリックにそれぞれ折りたたまれる(図 16)。 なおこのモジュールHについて、ブリックの名称Hxcは、xは入力を、cはキャリー(つまり$c = 0$ならキャリー無し$c = 1$ならキャリー有り)として表している。

図 12は一番最初のジグが折りたたまれている様子を表している。 ジグは部分配列$(FLHR)^kF$で表され、図中のジグにおいて$k = 1$でシードに記述されているカウントの値は0である。 ジグは*seed*の終了位置の影響で下側から始まるためキャリーが入力される。 それを、最初に転写されるモジュールFとLがそれに続くモジュールHにまで伝える。 キャリーが入力されたモジュールHは、もしその上に記述された値が0であったら図 12のようにH01に折りたたまれ、もし値が1ならH11に折りたたまれる。 H01はキャリーを解消するために上側で終了し、一度キャリーが消滅すればそのジグは左端に到達するまでキャリーがない状態が続く。 そしてその状態でジグが終了し次にモジュールLが転写されると、そのLは一つ上のターンシグナルと結合することで通常ターン用のブリックLcrnに折りたたまれる。 もし、ジグの中で一度もキャリーが解消されない場合、それはすなわちオーバーフローしている場合であり、そのままジグが左端に到達すると次のLはLbeへと折りたたまれ、オーバーフローの処理に移る。

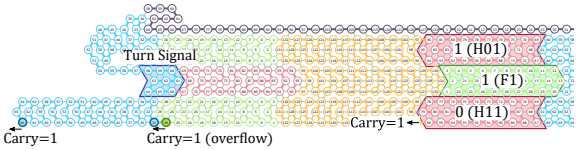


Figure 18: モジュールLはターンシグナルと結合せずにグライダー形に折りたたまれ、直進する。

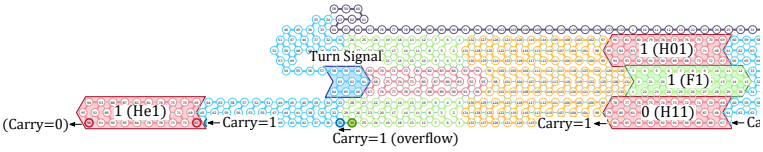


Figure 19: モジュールHはオーバーフローのキャリーを受け取るが、H自体は自立できるグライダー形をとる必要があるため、そのキャリーをキャンセルしない。 そのためHの配列の最後は下側に配置されるがキャリーは0として扱われる。

3.5 オーバーフローの処理

Gearyらのカウンターではジグが下側で終了した時、つまりカウントの値がオーバーフローした時にそれを処理できない。 そこで無限カウンタではオーバーフローを処理するために、ジグが下で終わった際にはモジュールLがブリックLbeに折りたたまれるように設計し、カウントアップの続行を実現した(図 18)。 図を見ると、このモジュールLの一つ上にはターンシグナル(58、63、64)が存在するが、それがLのターンシグナルである33、34と結合するには遠すぎるので、Lは左ターンしない。 Lbeは自立する構造であるグライダー形であり、周囲に結合を結べるものがなくともそれ自身の内部で結合をすることによって折りたたまれる。 Lに続くモジュールH、R、Fについても何もないところで折りたたみを進行する必要があるため、Lと同様グライダー形となり、それぞれのブリック名はHe1、Rb、Fnbである(図 19、20)。 なおブリックHe1については本質的にH00と同じである。 しかし、この二つは上下が反転していて、このブリックの一段下に来るザグのモジュールから見るとHe1は1を表す配列が露出していて一方でH00は0を表す配列が露出している。

オーバーフローした後に、モジュールL、H、R、Fと転写され、さらにその次に転写されるLを考える。 このモジュールLは、この周囲に何もない空間で左ターンする必要があるが、ターンシグナル存在しなければ再びグライダー形であるLbeとなってしまう。 そこで、それを避けるために、Lの35、36がFnbの上部にある28-27-22と結合することにする

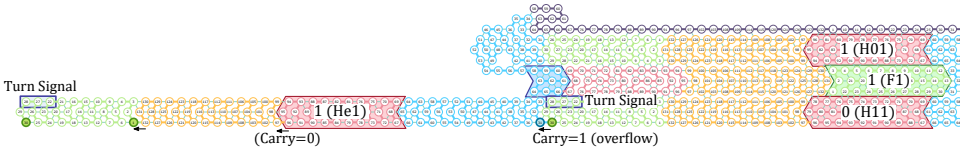


Figure 20: モジュールRとFが転写され、続いてモジュールLの折りたたみが開始される。 ここでは、通常のターンのようにターンシグナルを持ったLは周囲に存在しない。 なので、直前のモジュールFの上部をターンシグナルとして扱うことにする。 。

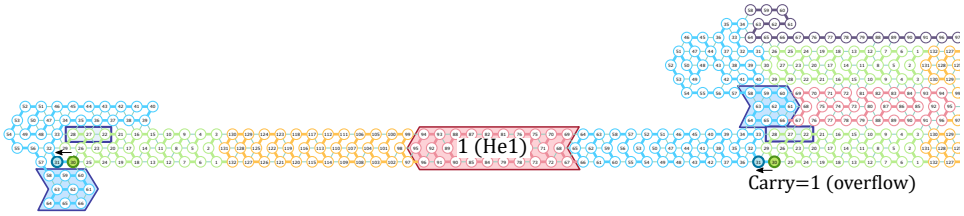


Figure 21: モジュールLはFのターンシグナルに結合することによって左ターンし、ザグが開始する。 また、ここで左ターンしたLの最後もターンシグナルとなる。

(図 20、21)。 つまり、Fnbの左上をターンシグナルとみなすことによってLは左ターン用のブリックLcre(図 15)に折りたたまれる。 このためブリックFnbは常にターンシグナルを持つこととなるが、この左上の箇所は通常(オーバーフローしていない状態)では、一つ上のザグによってターンシグナル部分が隠れているため、必要のない時にモジュールLと結合することを避けることができる。

3.6 フォーマット

ジグにおいて半加算器モジュールHが働くことによってカウントアップがされることがわかった。 しかし、カウントの値は式 (2)に従ってフォーマットされている必要があり、ジグが終わった段階ではそれは達成されていない。 フォーマットを行うには、モジュールHとFは縦に交互に並んでいることに注目すると、ザグにおいてモジュールHの出力をモジュールFが読み、そして式 (2)に従いフォーマットするように設計すれば良い。 すべてのザグはその列の下側から始まる。 これはモジュールLが左ターンする時のブリックはLcrnもしくはLcreであり、どちらも下側で終わるためである。 ザグにおいて、モジュール間で横方向には信号を伝える必要がないため、すべてのモジュールはザグの中では下から始まり、下で

終わる。 なおかつモジュールH、L、Rについては縦方向にも信号を伝える必要がないため、これらのブリックはそれぞれHn、Lbn、Rbにのみ折りたたまれれば十分である。 なおモジュールFについては、Fの一つ上のモジュールHがブリックHxcに折りたたまれたのであれば、ブリックFy ($y = (x + c) \mod 2$) に折りたたまれ、フォーマットを行うように設計した。

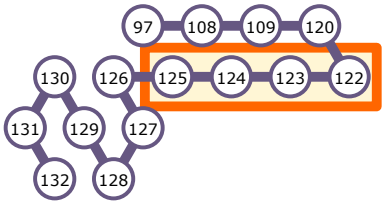


Figure 22: 0ビットからカウントを始める場合のシード。モジュールR用のターンシグナル部分とキャリーを与える部分が含まれている。

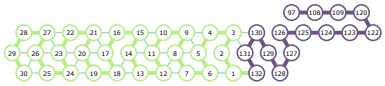


Figure 23: モジュールFが自立する構造(グライダー)に折りたたまれジグ方向へ進む。

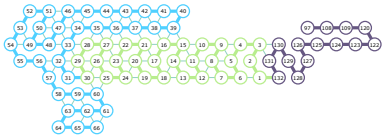


Figure 24: モジュールLがFのターンシグナルと結合して左ターンする。

3.7 0ビットからのカウントアップ

有限カウンタでは、1からカウントを始めたとしてもビット幅を定めるために、そのビット幅分の0が記述される。 しかし、この無限カウンタではビット幅が拡張できるため0ビットからのカウントアップすなわち、カウントをの値をシードに記述せずにカウントアップを始めることができる。 シードがカウントの値を持たないのであれば、右ターン用のターンシグナルさえあればいいため、図 22 のようなシードから転写が始まる。転写が始まると、その上部には折りたたまれた構造が存在しないので、

Figure 25: モジュールHがザグ方向へ進む。

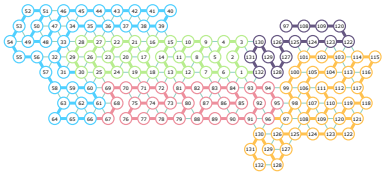


Figure 26: モジュールRがシードのターンシグナルと結合して右ターンし通常のカウントが開始される。

最初のモジュールFからオーバーフローした時と同じ振る舞いをする。モジュールFはグライダー形に折りたたまり、次のLはFと結合し左ターンする (図 23、24)。 ジグのモジュールFは下側に何も信号を出力しないため、この段階ではまだカウント値は0ビットである。 次にザグに移り、モジュールH、右ターンをするRと続きジグへと移る (図 25)。 このジグで再びオーバーフローが起こり、ここではカウント値のビット幅が拡張され、値「1」のカウントが行われ、これまで通りのカウントが続く。

3.8 実装したoritatami system

今回実装した無限バイナリカウンタのoritatami systemは以下の通りである。
(表〜)

この無限カウンタは、Gearyらのカウンタと設計の考え方が似ているが、左ターンモジュールにオーバーフローした際の処理と、その後に左ターンしザグに戻る処理を加えることでカウントビット幅の拡張を実現している。 しかし、oritatami systemでは既にGearyらによってチューリングマシンが実装されている[2]。 チューリングマシンを用いれば無限カウンタも実装できる。 ただ、このチューリングマシンの使用している塩基の種類数は542種類であるのに対して、今回実装した無限カウンタの塩基の種類数は132であり、より少なくできると言う利点がある。

4 まとめ

5 謝辞

References

[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] Cody Geary, Pierre-Étienne Meunier, Nicolas Schabanel, and Shinnosuke Seki. Proving the Turing universality of oritatami cotranscriptional folding. In *Proc. ISAAC 2018*, pages 23:1 – 23:13, 2018.

[3] Cody Geary, Pierre-Étienne Meunier, Nicolas Schabanel, and Shinnosuke Seki. Oritatami: A computational model for molecular co-transcriptional folding. *International Journal of Molecular Sciences*, 20(9):2259, 2019.

[4] A Xayaphoummine, V Viasnoff, S Harlepp, and H Isambert. Encoding folding paths of rna switches. *Nucleic acids research*, 35(2): 614–622, 2007.