Oritatami Systemによる 無限バイナリカウンタの実装

丸山晃平 *

令和 2年 1月 20日

1 はじめに

2 oritatami system とは

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

「周囲の環境」の説明

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[1]。 今回実装した無限バイナリカウンタもオーバーフローしていない部分のカウントアップについては、この有限カウンタと同じ

*情報・ネットワーク工学専攻 1831144 関 研究室

ような振る舞いをする。 具体的には、いずれの*transcript*もジグザク構造をとるように折りたたまれ、そのジグザグは段重ねになり一定方向（この章で用いる図では下方向）に伸びていく。 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。 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*transcript*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

無限カウンタの*transcript*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 1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 (**??**)に従って*transcript*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、各モジュールにおいて想定された環境にしか遭遇しないことが図 2の「ブリックオートマトン」によって保証されている。 この図では頂点して環境とブリックのペアが、頂点間の遷移としてブリックの前半部分が示されていて、さらにこのブリックオートマトンは閉じている。 そのため環境とブリックのペアのすべて

について、各環境でモジュールが正常に目的のブリックに折りたたまれることを確認するだけで、このブリックオートマトンによってsystemが正しく動作していることを保証できる。 動作確認は専用に開発したシミュレータを用いて行われ、その結果はhttps://komaruyama.github.io/oritatami-infinit-counter/に掲載されている。

3.2 *Seed*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、*seed*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、

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

上記の配列は、モジュールH、R、F、LがブリックHn、Rb、$\mathrm{F}b_i$、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. **??**、**??**、1、and **??**で確認できる。 例として、初期値$b_0 = 0$、ビット幅$k = 1$の*seed*は、図 **??**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(←)、ジグ(→)、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。

4 $|\Sigma| = 1$における非チューリング完全性について

5 まとめ

6 謝辞

References

[1] 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.

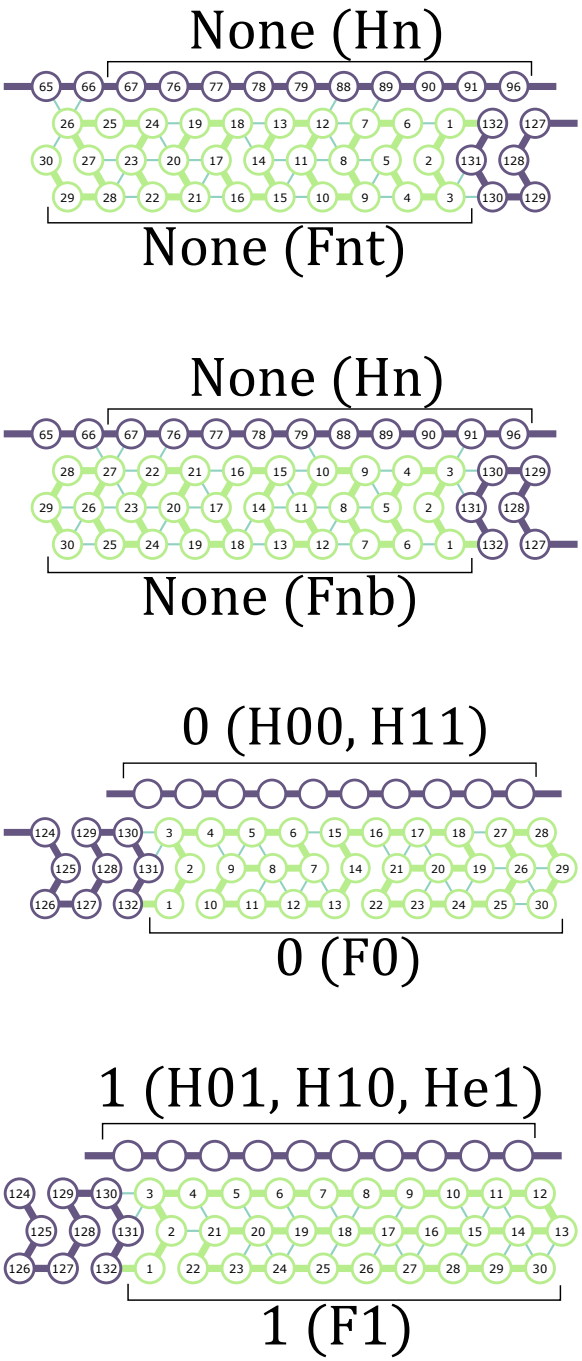

Figure 1: フォーマットモジュールのブリックは全部で4種類存在する。上二つのFntとFnbはジグでのみ現れ、下二つのF0とF1はザグにのみ現れる。

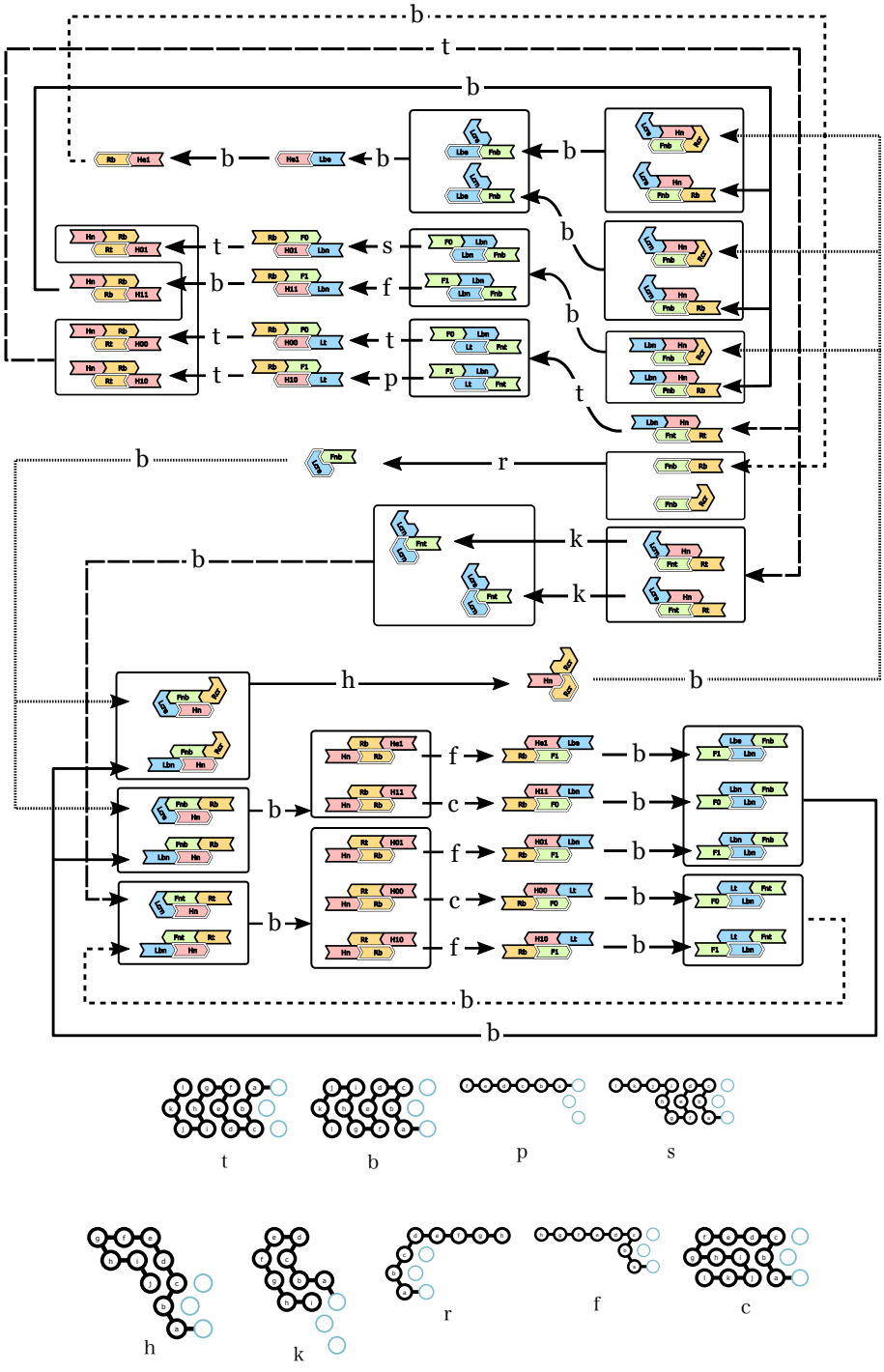

Figure 2: 無限バイナリカウンタのブリックオートマトン