社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2005－AL－103（2）
2005／11／11

切断オートマトン

高崎慶輔，伊藤 暁

山口大学工学部

あらまし　ベネンソンらが遺伝子工学の技術によって試験管内に実現させた分子オートマトン (molecular automaton) を一般化した理論モデルを提案し，その純オートマトン理論上の性質を調べる．

Cleaving Automata

Keisuke Takasaki, Akira Ito

Faculty of Engineering, Yamaguchi University

Abstract　This paper proposes a theoretically generalized model of molecular automaton which was experimentally realized in test tube using biotechnology by Benenson et a., and investigates its purely automata-theoretical properties.

1.　まえがき

ベネンソンらは [1, 2]，ある種の制限酵素 (restriction enzyme) が DNA 鎖の特定の場所を切断する (cleave) という機能を用いることで，試験管内に有限オートマトンを実現することに成功した．そこでは，４つの塩基を入力アルファベットとし，それらを任意に連接した DNA 二重鎖を入力データ列としている．有限制御部の状態は入力 DNA 鎖の先頭部分に埋めまれるようにコード化される．特定の状態と入力記号を表す DNA 先頭部位に対して共有結合可能な DNA 断片とそれらに取り付けられた切断酵素の対が一つの状態遷移を表す．これらは鍵付きのハサミの機能を提供する．各ハサミには切断対象の DNA の先頭からの長さが予め設定されている．従って，各ハサミはその鍵に合う先頭部位を持つ DNA をある特定の長さだけ先頭から削り取ることができる．試験管内に対象とするオートマトンの可能な状態遷移を表す鍵付きハサミの集合ならびに入力 NDA 鎖が入れられると，入力 DNA の切断（削り取り）が開始される．切断反応が繰り返された後に残された入力 DNA 鎖が終端記号と受理状態を表しているかどうかはゲル電気泳動法，磁気ビーズ法等で判定される．

本稿では，このように遺伝子工学の技術によって試験管内に実現した分子オートマトン (molecular automaton) を純粋オートマトン理論的に解釈することで得られる新たなオートマトンモデル（切断オートマトンと呼ぶ）を提案する．切断オートマトンは，入力テープを切断する機能しか持た

ず，通常の有限オートマトンのように得られた情報を有限制御部に蓄えることも出来ない．そのかわりに，入力列を切断することで処理単位を細分化し並列処理を行うことで，受理の可否を高速に判定することが出来る．

なお，言語理論の分野においては既にスプライシングシステム (Splicing System)[3] と呼ばれる遺伝子技術をモデル化したシステムが存在するが，言語生成系である点が本切断オートマトンとは異なる．

本稿の第 2 節では，その理論的な取り扱いやすさから本来の分子オートマトンを制限したモデルとして切断オートマトンを導入し，第 3 節で分子オートマトンと同等以上の能力を有するような切断オートマトンに拡張する．

2. 定義

本節では，切断オートマトンを定義し，その基本的性質を調べる．

2.1 切断言語

定義 2.1 切断オートマトン (cleaving automaton) は 3 個組 $M = (\Sigma, R, T)$ からなる．ここに，

(1) Σ は**入力アルファベット** (input alphabet),

(2) $R \subseteq \Sigma^* \dagger \Sigma^*, \dagger \notin \Sigma$ は**切断規則** (cleaving rule) の有限集合,

(3) $T = (P, \mathcal{D})$ は**検知要素** (probe) の有限集合 $P \subseteq \Sigma^*$ ならびに**検出パターン** (detection pattern) の有限集合族 $\mathcal{D} \subseteq 2^P$ の組からなり，**検査キット** (test kit) と呼ばれれる．

規則 $\alpha\dagger\beta$ は 列 $x\alpha\beta y$ を 2 つの列 $\{x\alpha, \beta y\}$ に切断することが可能である．規則 $\alpha\dagger = \alpha\dagger\epsilon$ は列 $x\alpha$ を $\{x\alpha, \epsilon\}$ と切断することに注意.

例 2.1 検査キットの定義により，“要素 s が検出される” なる命題の任意の論理結合が実現できる．例えば，$P = \{s, t\}$, $2^P = \{\emptyset, \{s\}, \{t\}, \{s, t\}\}$ のとき，$\mathcal{D} = \{\{s\}, \{t\}, \{s, t\}\}$ は，s, t いずれかが検出されることを意味する．$\mathcal{D} = \{\{s, t\}\}$ ならば，s, t いずれも検出されることを意味する．$\mathcal{D} = \{\emptyset\}$ ならば，s, t いずれも検出されないことを意味する．$\mathcal{D} = \{\{s\}, \{t\}\}$ ならば，s, t いずれか一方が検出されることを意味する．なお，$\mathcal{D} = \{\emptyset, \{s\}, \{t\}, \{s, t\}\}$ は実質的に何ら判定しない検出方法であり，$\mathcal{D} = \emptyset$ は実現不可能な検出方法である．

定義 2.2 切断オートマトン $M = (\Sigma, R, T)$ を考える．M に入力 $x \in \Sigma^*$ が与えられたとする．M の x 上の**計算状況** (configuration) は Σ^* の部分集合である．M の x 上の**初期計算状況** (initial configuration) を $I_M(x) = \{x\}$ とする．M の x 上の 2 つの計算状況 C, D に対して，

$$\exists \alpha\dagger\beta \in R, \exists, x, y \in \Sigma^* [x\alpha\beta y \in C, x\alpha, \beta y \in D, C - \{x\alpha\beta y\} = D - \{x\alpha, \beta y\}]$$

が成り立つとき，$C \vdash D$ と記す．明らかに，$I_M(x)$ から出発し関係 $\vdash$ で到達可能な計算状況の集合は有限な非巡回グラフ $G_M(x)$ をなす．$G_M(x)$ における子を持たない（出次数 0 の）点における計算状況の集合族を $\mathcal{T}_M(x)$ と記し，その要素を**終了計算状況** (terminal configuration) と呼ぶ．

計算状況 C は入力 x の部分列の有限集合であり，物理的には多重集合であるはずのものを集合と見なしている．これは，遺伝子工学の実験において同一種の分子の個数自体を正確に数え上げることの困難さから来ている．$\mathcal{T}_M(x)$ の各要素は，それ以上切断過程が進まない状況を表している．一つの計算状況から他の計算状況への遷移では，一つの列に対し一つの規則が列の一箇所に一回だけ適用されるものの，そこには様々な非決定な動作が含まれている．例えば，現在の計算状況の中のどの列を切断するか，その際にどの切断規則を使うか，また列のどの部分にその切断規則を適用するか，といった事項は予め定められている訳ではない．

例 2.2 (1) $r_1 = \mathtt{a}\dagger\mathtt{b}, r_2 = \mathtt{ab}\dagger\mathtt{c}$ なる 2 つの規則を列 $x = \mathtt{abc}$ に適用する際に，r_1 を先に適用すると，x は $\mathtt{a}, \mathtt{bc}$ と分解され，r_2 は適用できなくなる．一方 r_2, r_1 の順に適用すると，$\mathtt{a}, \mathtt{b}, \mathtt{c}$ と分解される．(2) 規則 $r = \mathtt{aa}\dagger\mathtt{a}$ を 列 $x = \mathtt{aaaa}$ に適用する際に，x の中央が切断されるように適用すると，$\mathtt{aa}, \mathtt{aa}$ と分解され，それ以上は分解できない．一方 x の右端の $\mathtt{a}$ が切り出されるように適用すると，$\mathtt{aaa}, \mathtt{a}$ と分解され，もう一度 r を適用することで，最終的に $\mathtt{aa}, \mathtt{a}, \mathtt{a}$ と分解される．

以上のように，本モデルは非同期的な並列処理の一種と見なすことができる．

定義 2.3 切断オートマトン $M = (\Sigma, R, (P, \mathcal{D}))$ と言語 $L \subseteq \Sigma^*$ に対して，条件

(1) $x \in L \Rightarrow \forall C \in \mathcal{T}_M(x)[C \cap P \in \mathcal{D}]$（“ M は x を受理する ”と言う），

(2) $x \notin L \Rightarrow \forall C \in \mathcal{T}_M(x)[C \cap P \notin \mathcal{D}]$（“ M は x を受理しない ”と言う），

が成り立つとき，L は M で**受理** (accept) されると言う．M で受理される言語を $L(M)$ と記す．切断オートマトンで受理される言語を**切断言語** (cleaving language) とよぶ．

上記定義における $P \cap C \in \mathcal{D}$ は，最終計算状況 C の中の幾つかの検知要素 P のみに着目したとき，そのパターンが $\mathcal{D}$ 内のある検出パターンに一致することを意味する．受理非受理いずれの場合も到達可能なすべての最終計算状況での条件成立を要求している．受理と非受理の判定が通常のような互いの否定関係ではないことに注意されたい．この定義は最も確実性を重んじた受理判定方法と言えるが，同じ実験を多数回繰り返すことを想定して，ラスベガス型あるいはモンテカルロ型の判定条件に変更することも可能である．

2.2 切断言語の例

定理 2.1 (パターン照合) $\Sigma^* w \Sigma^*$, $w \in \Sigma^*$ は切断言語である．

（証明）切断オートマトン $M = (\Sigma, \{a\dagger w, w\dagger a \mid a \in \Sigma\}, (\{w\}, \{\{w\}\}))$ がそのような言語を受理できる．規則 $a\dagger w, w\dagger a$ は入力列から部分列 w の切り出しを行う．最終計算状況において w が検出されることが受理の条件である． □

例 2.3 $(0+1)^*11(0+1)^*$ は切断オートマトン $M = (\{0,1\}, \{0\dagger 11, 1\dagger 11, 11\dagger 0, 11\dagger 1\}, (\{1\}, \{\{1\}\})$ で受理される．

定理 2.2 $a_1^* a_2^* \cdots a_k^*, k \geq 1, a_i \neq a_j\ (i \neq j)$ は切断言語である．

（証明）$k = 1$ の場合： $\mathtt{a}^*$ は切断オートマトン $M = (\{\mathtt{a}\}, \emptyset, (\emptyset, \{\emptyset\}))$ で受理される．M は常に何もしないで受理する．
$k = 2$ の場合： $\mathtt{a}^*\mathtt{b}^*$ は切断オートマトン $M = (\{\mathtt{a}, \mathtt{b}\}, \{\mathtt{a}\dagger\mathtt{a}, \mathtt{a}\dagger\mathtt{b}, \mathtt{b}\dagger\mathtt{b}\}, (\{\mathtt{ba}\}, \{\emptyset\}))$ で受理される．$(\{\mathtt{ba}\}, \{\emptyset\})$ は ba が検出されないことを意味する．M の規則では ba なる部分列が切断されないことに注意．
$k = 3$ の場合: $\mathtt{a}^*\mathtt{b}^*\mathtt{c}^*$ は切断オートマトン $M = (\{\mathtt{a}, \mathtt{bc}\}, \{\mathtt{a}\dagger\mathtt{b}, \mathtt{b}\dagger\mathtt{c}, \mathtt{a}\dagger\mathtt{a}, \mathtt{b}\dagger\mathtt{b}, \mathtt{c}\dagger\mathtt{c}\}, (\{\mathtt{ac}, \mathtt{cb}, \mathtt{ba}, \mathtt{ca}\}, \{\emptyset\}))$ で受理される．
k が一般の場合も上と同様にして示される（詳細は省略する）． □

定理 2.3 有限集合は切断言語である．

（証明）有限集合 $L \subseteq \Sigma^*$ に対して，$M = (\Sigma, \emptyset, (L, \mathcal{D}))$, $\mathcal{D} = \{\{x\} \mid x \in L\}$（各 $x \in L$ の何れか一つが検出される）とおく．任意の入力 $x \in \Sigma^*$ に対して，$\mathcal{T}_M(x) = \{\{x\}\}$ であるから，$x \in L \Leftrightarrow \{x\} \in \mathcal{D}$ が成り立つ． □

例 2.4 切断オートマトン $M = (\{\mathtt{a}\}, \emptyset, (\{\epsilon, \mathtt{a}\}, \{\{\epsilon\}, \{\mathtt{a}\}\}))$ は有限言語 $\epsilon + \mathtt{a}$ を受理する．

定理 2.4 切断集合の補集合は切断集合である．

（証明）切断オートマトン $M = (\Sigma, R, (P, \mathcal{D}))$ に対して，$M' = (\Sigma, R, (P, 2^P - CALD))$ とおいた切断オートマトン M' は M の補集合を受理する． □

例 2.5 切断オートマトン $M' = (\{\mathtt{a}\}, \{\mathtt{a}^2\dagger\mathtt{a}^2\}, (\{\epsilon, \mathtt{a}\}, \{\emptyset\}))$ は $\mathtt{a}^2\mathtt{a}^* = \mathtt{a}^* - (\mathtt{a} + \epsilon)$ を受理する（$\mathtt{a}^2\mathtt{a}^*$ は $(\{\mathtt{a}\}, \{\mathtt{a}^2\dagger\mathtt{a}^2\}, (\{\mathtt{a}^2, \mathtt{a}^3\}, \{\{\mathtt{a}^2\}, \{\mathtt{a}^3\}, \{\mathtt{a}^2, \mathtt{a}^3\}\}))$（$\mathtt{a}^2$ か $\mathtt{a}^3$ のいずれかが検出される）でも受理できる）．

補題 2.1 (非切断言語の例) 正規言語 $(\mathtt{aa})^* = \{a^2 m \mid m \geq 0\}$ は切断言語ではない．

（証明）$L = (\mathtt{aa})^*$ を受理する切断オートマトン $M = (\{\mathtt{a}\}, R, (P, \mathcal{D}))$ が存在するものと仮定する．R のある特定の要素 $r = \mathtt{a}^k\dagger\mathtt{a}^l$ を考える．ここで，$m = \max\{k, l\}$, $x = \mathtt{a}^{2m+1}$ とおく．$xx = \mathtt{a}^{2(2m+1)} \in L$ より，xx 上の任意の最終計算状況 $C_M^*(xx)$ に対して，$C_M^*(xx) \in \mathcal{D}$ が成り立たなければならない．

一方，規則 r がこの入力 xx に最初に適用されれば $C_M^{(1)}(xx) = \{x, x\} = \{x\} = \{\mathtt{a}^{2m+1}\}$ と切断する可能性がある．$C_M^{(1)}(xx)$ からの遷移は，M に入力 x が単独に与えられた場合と同じであり，その最終計算状況どうしも一致する．すなわち，$C_M^*(xx) = C_M^*(x) \cup C_M^*(x) = C_M^*(x)$ が成り立つ．ここに，$C_M^*(x)$ は M の x 上の最終計算状況である．$x \notin L$ より，任意の $C_M^*(x)$ に対して，$C_M^*(x) \notin \mathcal{D}$ とならなければならない． □

3. 終端付き切断オートマトン

本節では，先に定義した切断オートマトンに対して入力の終端が認識可能な機能を付加したモデルについて調べ，このオートマトンがベネンソンらによる分子オートマトンと同等以上の能力を持つことを示す．

定義 3.1 切断規則が $R \subseteq (\not\epsilon + \epsilon)\Sigma^*\dagger\Sigma^*(\$ + \epsilon)$, $\dagger, \not\epsilon\$ \notin \Sigma$，検知集合が $P \subseteq \not\epsilon\Sigma^*\$$ であるような切断オートマトン $M = (\Sigma, R, T)$ を**終端付き切断オートマトン** (cleaving automaton with terminaters) とよぶ．

例えば，$r = \not\epsilon\alpha\dagger\beta$ なる規則は，列 x 上の部分列 $\alpha\beta$ で左端にあるもののみを切断できる．

定義 3.2 終端付き切断オートマトン $M = (\Sigma, R, T)$ を考える．M に入力 $x \in \Sigma^*$ が与えられたとする．M の x 上の計算状況は $\not\epsilon\Sigma^*\$$ の部分集合である．M の $x \in \Sigma^*$ 上の初期計算状況は $\{\not\epsilon x\$\}$ である．M の x 上の 2 つの計算状況 C, D に対して，

$$\exists\alpha\dagger\beta \in R, \exists x, y \in \Sigma^*[\not\epsilon x\alpha\beta y\$ \in C, \not\epsilon x\alpha\$, \not\epsilon\beta y\$ \in D, C - \{\not\epsilon x\alpha\beta y\$\} = D - \{\not\epsilon x\alpha\$, \not\epsilon\beta y\$\}]$$

が成り立つとき，$C \vdash D$ と記す．終了計算状況その他は終端無しの切断オートマトンの場合と同様に定義される．

規則 $\alpha\dagger$ は列 $\not\epsilon\alpha\$$ を $\{\not\epsilon\alpha\$, \not\epsilon\$\}$ と切断することに注意．終端付き切断オートマトンで受理される言語を**終端付き切断言語** (cleaving language with terminaters) と呼ぶ．

補題 3.1 $(\mathtt{aa})^*$ は終端付き切断言語である．

（証明）$M = (\{\mathtt{a}\}, (\not\epsilon\mathtt{a}\dagger\mathtt{a}\}, (\{\not\epsilon\mathtt{a}\$\}, \{\emptyset\}))$ は $(\mathtt{aa})^*$ を受理する． □

補題 3.2 終端なし切断言語は終端付き切断言語である．

（証明）切断オートマトン $M = (\Sigma, R, (P, \mathcal{D}))$ が受理する言語は，$M' = (\Sigma, R, (P', \mathcal{D}'))$ なる切断オートマトンで受理できる．ここに，$P', \mathcal{D}'$ は $P, \mathcal{D}$ 内に現れる全ての列 s を $\not\epsilon s\$$ に置き換えたものである． □

補題 2.1, 3.1 ならびに 3.2 より，以下を得る．

定理 3.1 (終端なし切断言語との関係) 終端付き切断言語族は終端なし切断言語族を真に含む．

定理 3.2 終端付き切断言語は正規言語である．

（証明）まず，非終端切断オートマトンの模倣を考える．与えられた切断オートマトン M に対して，それを模倣するような非決定性有限オートマトン M' は次のように動作する．入力 x を左から右に走査しながら，M による切断箇所を逐次的に推測し，切断される断片で検知要素と一致するものを集合変数 P' に追加していく（断片の登録処理と呼ぶ）．最終的に入力を全て読み終えた

時点で P' が $\mathcal{D}$ に含まれているかどうかを確認し，$P' \in \mathcal{D}$ ならば M' は受理状態に入る．切断箇所の逐次的推測においては以下のように動作する（図 1 参照）．

1. **前進切断**：現切断箇所 p から右に $\max\{|\alpha|, |\beta| \mid x\alpha\dagger\beta y \in R\}$ まで離れた区間内に，切断可能な地点があるかどうかチェックする．もし切断可能な場合には，次期切断点 p'，それに適合する切断規則 r'，ならびに現切断点との時間的前後関係を推測し，後退チェックに進む．もし可能でなければ，切断された断片の登録処理を行ってから終了する．

2. **後退チェック**：前切断点と現切断点の隙間が更に切断できないかチェックする．もし切断可能な場合には切断点 $p_1, p_2 \ldots$，適合する切断規則 $r_1, r_2, \ldots$，ならびにそれら切断点どうしの時間的前後関係を推測し，再帰的に後退チェックを行う．もし切断可能でないならば，切断された断片の登録処理を行ってから前進切断に戻る．

終端付きの場合には，各切断片の両端に終端記号 $\not{c}, \$$ を仮定すればよい． □

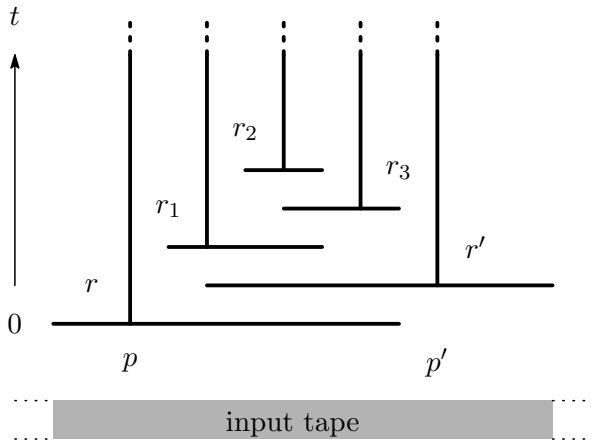

図 1. 切断箇所の逐次的推測の概念図

定理 3.2 の逆については以下が言える．終端付き切断オートマトンであれば有限オートマトンと同じく入力の左端から逐次的に文字を消費できる．しかしながら，有限オートマトンの現在の状態を記憶しておくような機能が元々組み込まれていない．そこで，入力の一文字ごとにダミー文字列を付加し，入力の切断点で残されるダミー文字列の長さで状態を表すことにする．これがベネンソンらが物理的に分子オートマトンを実現した際に用いた根本的なアイデアである [1].

定理 3.3 (ベネンソン分子オートマトン) 任意の正規言語は，ある終端付き切断言語の射影 (projection) である．

（証明）L を受理する（決定性）有限オートマトンを $M = (\Sigma, Q, q_0, F, \delta)$ とする．一般性を失うことなく，$Q = \{q_0, q_1, \ldots, q_k\}$, $F = \{q_k\}$ と仮定する．終端付き切断オートマトン $M' =$

$(\Sigma \cup \{\#\}, R, (P, \mathcal{D}))$ を次のように定義する．

$$\begin{cases} R &= \{\not{\mathsf{c}} a^i \# v^j \dagger v^{k-j+1} \# \mid a \in \Sigma, v \in \Sigma \cup \{\#\}, \delta(q_{i-1}, a) = q_{k-j}\} \\ P &= \{\not{\mathsf{c}} \#^{k+1} \#\$, \not{\mathsf{c}} a^i \# v^j \$ \mid a \in \Sigma, v \in \Sigma, \delta(q_{i-1}, a) = q_{k-j}\} \\ \mathcal{D} &= \{\{\not{\mathsf{c}} \#^{k+1} \#\$\} \cup T \mid T \subseteq \{\not{\mathsf{c}} a^i \# v^j \$ \mid a \in \Sigma, v \in \Sigma, \delta(q_{i-1}, a) = q_{k-j}\}\} \end{cases}$$

ここで，射影 h を $h(a^{|Q|}) = a$, $h(\#) = \epsilon$ と定義する．従って，M' に対する入力が $a_1 \# a_2^{|Q|} \# \cdots \# a_n^{|Q|} \# \# ^{|Q|} \#$ のとき，M に対する入力は $a_1 a_2 \cdots a_n$ となる．$h(L(M')) = L(M)$ の厳密な証明は省略する． □

例 3.1 $L = (\mathtt{a} + \mathtt{b})^* \mathtt{b}$ を受理する有限オートマトンを $M = (\{\mathtt{a}, \mathtt{b}\}, \{q_0, q_1\}, q_0, \{q_1\}, \delta)$ とする．ここに，

$$\delta(q_0, \mathtt{a}) = q_0, \delta(q_1, \mathtt{a}) = q_0, \delta(q_0, \mathtt{b}) = q_1, \delta(q_1, \mathtt{b}) = q_1.$$

M に対する入力 $x = a_1 a_2 \cdots a_n$ に対して，M' に対する入力は $a_1 \# a_2^2 \# \cdots \# a_n^2 \# \#^2 \#$ である．

対応する終端付き切断オートマトン $M' = (\{\mathtt{a}, \mathtt{b}, \#\}, R, (P, \mathcal{D}))$ は次のようになる．

$$\begin{cases} R &= \{\not{\mathsf{c}}\mathtt{a} \# v \dagger v \#, \not{\mathsf{c}}\mathtt{aa} \# v \dagger v \#, \not{\mathsf{c}}\mathtt{b} \# \dagger v v \#, \not{\mathsf{c}}\mathtt{bb} \# \dagger v v \# \mid v \in \{\mathtt{a}, \mathtt{b}, \#\}\} \\ P &= \{\not{\mathsf{c}} \#^2 \#\$, \not{\mathsf{c}}\mathtt{a} \# v \$, \not{\mathsf{c}}\mathtt{aa} \# v \$, \not{\mathsf{c}}\mathtt{b} \# \$, \not{\mathsf{c}}\mathtt{bb} \# \$ \mid v \in \{\mathtt{a}, \mathtt{b}\}\} \\ \mathcal{D} &= \{\{\not{\mathsf{c}} \#^2 \#\$\} \cup T \mid T \subseteq \{\not{\mathsf{c}}\mathtt{a} \# v \$, \not{\mathsf{c}}\mathtt{aa} \# v \$, \not{\mathsf{c}}\mathtt{b} \# \$, \not{\mathsf{c}}\mathtt{bb} \# \$ \mid v \in \{\mathtt{a}, \mathtt{b}\}\}\} \end{cases}$$

“$\not{\mathsf{c}}\mathtt{a} \# \cdots$” (“$\not{\mathsf{c}}\mathtt{b} \# \cdots$”) は状態 “$q_0$” で文字 “a” (“b”) を読もうとしており，同様に “$\not{\mathsf{c}}\mathtt{aa} \# \cdots$” (“$\not{\mathsf{c}}\mathtt{bb} \# \cdots$”) は状態 “$q_1$” で文字 “a” (“b”) を読もうとしている状況を表す．

M が x を読み終えたときの状態が q_0 ならば，$\not{\mathsf{c}} \# \# \$ \in \mathcal{T}^*_{M'}(x)$ であることに注意． □

4. 課題

今後本稿で提案した切断言語の基本的な性質，特に類似の性質を持つ単純スプライシング言語 [4] との関係を調べる必要がある．また，以下に挙げるような観点から検討を行うことも興味深い．

切断オートマトンの拡張

- 多切断型 (multi-cleave) 切断オートマトン：切断規則が $R \subseteq (\Sigma \cup \{\dagger\})^*$ のような切断オートマトンは通常の単一切断型に変換可能か．
- 多段階 (multi-stage) 切断オートマトン：一旦終了計算状況に達した後に，他の切断規則を用いて再計算を行う．$\{(\mathtt{a}\#)^n (\mathtt{b}\#)^n \mid n \geq 1\}$ が終端記号付き多段階切断オートマトンで受理されることは容易に示せる．従って，多段階切断言語族は正規言語でない文脈自由言語を含む．

切断オートマトンの制限

- 片側 (one-sided) 切断オートマトン：切断規則が $C \subseteq \Sigma^*\dagger$ のような切断オートマトンの受理能力.
- 時間計算量 (time complexity)：切断規則は非同期並列的に適用されると考えれば，言語 $\mathtt{a}^*\mathtt{b}^*$ は $O(1)$ 時間で認識される.
- 記述計算量 (descriptional complexity)：例えば，$a_1^*a_2^*\cdots a_k^*$ を受理する切断オートマトンの記述計算量（$|R|+|T|$）は k に関してどれほどのオーダーが必要になるか.

その他

- 標準形 (standard form)：切断オートマトンの標準形はあるか．無用な規則の除去，規則同士の包含性判定，オートマトンの等価性判定問題.
- 切断パターンの数え上げ (enumeration)：終了計算状況時に生成されているパターンの数え上げ問題．規則 $r = \mathtt{a}^k\dagger\mathtt{a}^l$ によって，系列 a^n を切断するとき，各パターン $p_i = \mathtt{a}^i$ は幾つずつ生成されるか，各切断地点に対する規則の適用順の違いによるパターン頻度分布はどのようになるか.
- 規則適用順による曖昧性 (ambiguity)：項書き換え系としての合流性.

参考文献

[1] Y. Benenson, R. Adar, T. Paz-Elizur, Z. Livneh, and E. Shapiro, DNA molecule provide a computing machine with both data and fuel, Proc. Natl Acad. Sci. USA 100, 2191-2196 (2003).

[2] Y. Benenson, B. Gil, U. Ben-Dor, R. Adar, and E. Shapiro, An autonomous molecular computer for logical control of gene expression, Nature 429 (2004), 423-428.

[3] T. Head, D. Pixton, and E. Goode, Splicing systems: regularity and below, Proc. 8th Workshop on DNA Computers, xxxx, xx-xx.

[4] A. Mateescu, Gh. Păun, G. Rozenberg, and A. Salomaa, Simple splicing systems, Disc. Appl. Math. 84, 1998, 145-163.