

## ある拡張文法/オートマトンに関するコメント

東京女子大学 (文理) 守屋悦朗 (Etsuro Moriya)

### § 0 はじめに

文脈自由文法 (CFG) の拡張として様々のものが知られている [7] 中でも、インデックス文法 [1] は最も自然な拡張と言えよう。CFG の持つ望ましい性質 (例えば、導出木が描ける、最左導出と導出木とが 1 対 1 に対応する、多くの決定問題が帰納的に可解である、など) の多くを保存する唯一の拡張といってもよい。

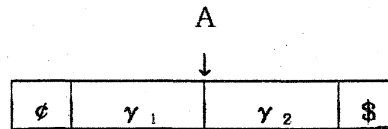
さて、インデックス文法は、各非終端記号がプッシュダウンスタックを持っているような CFG と考えることが出来る。従って、自然の拡張として、非終端記号の持つ記憶機構を拡張することが考えられる。実際、既に西沢 [5] がそのことをきわめて一般的に扱っているが、ここでは具体的な例として、1 本ないし 2 本のスタック (またはキュー、またはカウンター) を持った CFG を考察する。また、それ等の文法をオートマトンで特徴付ける。このオートマトンは、[6] で導入された partitioning オートマトンのプッシュダウンオートマトン版である。

### § 1 スタックを持つ CFG

はじめに、メモリを持つ文脈自由文法 (CFG M) を一般的に定義する。CFG M (context-free grammar with memory) は、 $G = (N, \Sigma, \Gamma, P, S, \phi, \$)$  なる 5 項組によって指定される。ここに、

- (1)  $N$  は非終端記号の有限集合、
- (2)  $\Sigma$  は終端記号の有限集合、
- (3)  $\Gamma$  はメモリ記号の有限集合、
- (4)  $S \in N$  は文記号、
- (5)  $\phi, \$ \in N \cup \Sigma \cup \Gamma$  はメモリのエンドマーカであり、
- (6)  $P$  は次のいずれかの形のプロダクションよりなる有限集合である:
  - ①  $A \rightarrow \alpha$  ( $A \in N; \alpha \in (N \cup \Sigma)^*$ )
  - ② (i)  $A \rightarrow Bf$  (ii)  $A \rightarrow fB$  ( $A, B \in N; f \in \Gamma$ )
  - ③ (i)  $Af \rightarrow \alpha$  (ii)  $fA \rightarrow \alpha$  ( $A \in N; \alpha \in (N \cup \Sigma)^*$ )
  - ④  $Af \rightarrow Bg, Af \rightarrow gB, fA \rightarrow gB, fA \rightarrow Bg$  ( $A, B \in N; f, g \in \Gamma$ )

直感的には、CFG Mとは、次のように各非終端記号がメモリを持っているような文脈自由文法と考えることが出来る。文形式  $(\phi \Gamma^* N \Gamma^* \$ \cup \Sigma)^*$  の元  $\alpha \phi \gamma_1 A \gamma_2 \$ \beta$  (ただし、 $\alpha, \beta \in (\phi \Gamma^* N \Gamma^* \$ \cup \Sigma)^*$ ;  $\gamma_1, \gamma_2 \in \Gamma^*$ ) において、各非終端記号  $A$  はメモリ内容  $\gamma_1 \gamma_2$  を持ち、メモリポインタは  $\gamma_1$  の最右文字(または、 $\gamma_2$  の最左文字)を指していると考え。  $\phi$  と  $\$$  は、メモリの範囲を明示するためのメタシンボルであり、メモリの内容ではない。



$G$  における導出を定義しよう。  $(\phi \Gamma^* N \Gamma^* \$ \cup \Sigma)^*$  の上の2項関係  $\Rightarrow$  を次のように定義する。 $\beta, \gamma \in (\phi \Gamma^* N \Gamma^* \$ \cup \Sigma)^*$ ;  $\delta, \epsilon \in \Gamma^*$ ;  $A_1, \dots, A_k \in N \cup \Sigma$  に対して、

- 1)  $A \rightarrow A_1 \dots A_k$  が①型プロダクションのとき、

$$\beta \phi \delta A \epsilon \$ \gamma \Rightarrow \beta \delta_1 A_1 \epsilon_1 \dots \delta_k A_k \epsilon_k \gamma$$

ここに、 $A_1 \in N$  のとき  $\delta_1 = \phi \delta, \epsilon_1 = \epsilon \$$ ;  $A_1 \in \Sigma$  のとき  $\delta_1 = \epsilon_1 = \lambda$

すなわち、プロダクション左辺の非終端記号のメモリ内容はプロダクション右辺の各非終端記号に等しく継承される。

- 2)  $A \rightarrow Bf$  (or  $A \rightarrow fB$ ) が②型のプロダクションのとき、

$$\beta \phi \delta A \epsilon \$ \gamma \Rightarrow \beta \phi \delta Bf \epsilon \$ \gamma \quad (\text{or } \beta \phi \delta A \epsilon \$ \gamma \Rightarrow \beta \phi \delta fB \epsilon \$ \gamma)$$

これは、メモリへの書き込みプロダクションである。

- 3)  $Af \rightarrow A_1 \dots A_k$  (or  $fA \rightarrow A_1 \dots A_k$ ) が③型のプロダクションのとき、

$$\beta \phi \delta Af \epsilon \$ \gamma \Rightarrow \beta \delta_1 A_1 \epsilon_1 \dots \delta_k A_k \epsilon_k \gamma$$

$$(\text{or } \beta \phi \delta fA \epsilon \$ \gamma \Rightarrow \beta \delta_1 A_1 \epsilon_1 \dots \delta_k A_k \epsilon_k \gamma)$$

ここに、 $A_1 \in N$  のとき  $\delta_1 = \phi \delta, \epsilon_1 = \epsilon \$$ ;  $A_1 \in \Sigma$  のとき  $\delta_1 = \epsilon_1 = \lambda$

これは、メモリ内容消去のためのプロダクションである。

- 4)  $\zeta \rightarrow \theta$  が④型のプロダクションのとき、

$$\beta \phi \delta \zeta \epsilon \$ \gamma \Rightarrow \beta \phi \delta \theta \epsilon \$ \gamma$$

これは、メモリポインタ移動(同時に、メモリ内容も変え得る)のためのプロダクションである。

$\Rightarrow^*$  を  $\Rightarrow$  の反射推移閉包とすると、

$$L(G) = \{ w \in \Sigma^* \mid \phi \$ \$ \Rightarrow^* w \}$$

を、 $G$  の生成する CF L M という。

さて、CFG Mの持つメモリ構造はきわめて一般的なので、次のようにいくつかの制限を考える。

(a) ④型のプロダクションを含まないとき、CFG2P (context-free grammar with two pushdown stores) という。 $\gamma_1$ の部分がAの左プッシュダウンストア ( $\gamma_1$ の最右文字がトップ) にあたり、 $\gamma_2$ の部分が右プッシュダウンストア ( $\gamma_2$ の最左文字がトップ) にあたる。④型のプロダクションがないので、メモリの書き換えはいつも $\gamma_1$ と $\gamma_2$ の境界のところで行なわれる。

(b) CFG2Pで、かつ、 $\Gamma = \{\$, Z\}$ 、かつ、任意の文形式が  $(\emptyset \neq \$Z^*NZ^*\$ \cup \emptyset NZ^*\$ \cup \emptyset \neq Z^*N\$ \cup \emptyset N\$ \cup \Sigma)^*$  の形をしているようなCFG MをCFG2C (context-free grammar with two counters) という。 $\$$ は、プッシュダウンストアの底を表す。

(c) 以下では、②(ii)、③(ii)型のプロダクションを許さず、かつ、④型のプロダクションは次の形のものだけに制限する：

$$Af \rightarrow Bf, Af \rightarrow fB, fA \rightarrow fB, fA \rightarrow Bf$$

すなわち、④型のプロダクションはメモリポインタの移動 (スタックreadingに相当する) だけに使えメモリ内容の書き換えは出来ない。さらに、②(ii)型プロダクション  $A \rightarrow Bf$ 、③(ii)型プロダクション  $Af \rightarrow B$  が適用出来るのは、次の形の文形式に限定する：

$$\beta \neq A\delta \$ \gamma \quad \text{または} \quad \beta \neq Af\delta \$ \gamma$$

$$(A, B \in N; f \in \Gamma; \delta \in \Gamma^*; \beta, \gamma \in (\emptyset \neq \Gamma^*N\Gamma^*\$ \cup \Sigma)^*)$$

すなわち、どの非終端記号もプッシュダウンストアを1つだけ持ち (それは右プッシュダウンストアである)、②(i)型プロダクションによるメモリへの書き込み (プッシュダウンに相当) および、③(i)型プロダクションによるメモリの消去 (ポップアップに相当) は右プッシュダウンストアのトップでのみ行なわれる。このようなCFG Mを CFG S (context-free grammar with a stack) という。

(d) ④型のプロダクションを含まないCFG SをCFG P (context-free grammar with a pushdown store) という。これは、インデックス文法に他ならない。

(e) CFG Pで、かつ、 $\Gamma = \{\$, Z\}$ 、かつ、任意の文形式が  $(\emptyset \neq NZ^*\$ \cup \emptyset N\$ \cup \Sigma)^*$  の形をしているようなものをCFG C (context-free grammar with a counter) という。

これ等以外にも、CFG SC (context-free grammar with a stack-counter [3])、CFG Q (context-free grammar with a queue) 等も自然に定義出来るが省略する。X型の文法によって生成される言語のクラスを  $\mathcal{L}_X$  で表すことにする。

定理1  $\mathcal{L}_{CFG2C} = \mathcal{L}_{CFG2P} = \mathcal{L}_{CFG2Q} = \mathcal{L}_{CFG M} =$  帰納的可算言語のクラス

証明は、任意のチューリングマシンは2-counter machineによってシミュレート出来るので、2-counter machine をCFG2Cでシミュレート出来ることをいえばよい。詳細は省略する。なお、この結果は一部 [5] で既に示されている。

定理2  $\mathcal{L}_{CFG} \subseteq \mathcal{L}_{CFG C} \subseteq \mathcal{L}_{CFG P} \subseteq \mathcal{L}_{CFG S} \subseteq \mathcal{L}_{CSG}$

証明は省略するが、例えば

$$\{ a^n b^n c^n \mid n \geq 0 \} \in \mathcal{L}_{CFGC} - \mathcal{L}_{CFG},$$

$$\{ ww \mid w \in \{a, b\}^* \} \in \mathcal{L}_{CFGP} - \mathcal{L}_{CFGC},$$

$$\{ (w\#)^{|w|} \mid w \in \Sigma^* \} \in \mathcal{L}_{CFGS} - \mathcal{L}_{CFGP}$$

である。なお、 $\mathcal{L}_{CFGS} \subseteq \mathcal{L}_{CSG}$ かどうかは不明であるが、CFG Sに対する emptiness problemはrecursively solvableであろうと予想しており、それが示せば $\mathcal{L}_{CFGS} \subseteq \mathcal{L}_{CSG}$ である。

## § 2 分割的オートマトン

分割的オートマトン (partitioning automaton) は、池川 [6] によって、有限オートマトンとプッシュダウンオートマトンに対して初めて導入された概念である。ここではスタックオートマトンに対して分割的スタックオートマトンを定義し、それによってCFG Sが特徴付けられることを示す。

PSA (partitioning stack automaton) とは、 $M = (Q, U, \Sigma, \Gamma, \delta, q_0, Z_0, F)$  のことである。ここに、

- 1)  $Q$  は状態の有限集合、
- 2)  $U \subset Q$  は partitioning state の集合、
- 3)  $\Sigma$  は 入力記号 の有限集合、
- 4)  $\Gamma$  は スタック記号 の有限集合、
- 5)  $q_0 \in Q$  は 初期状態、
- 6)  $Z_0 \in \Gamma$  は 初期スタック記号、
- 7)  $F \subset Q$  は 受理状態 の集合、
- 8)  $\delta$  は  $Q \times (\Sigma \cup \{\lambda\}) \times \Gamma$  から  $\bigcup_n (Q \times \Gamma^*)^n \cup \bigcup_n (Q \times \{-1, 1\})^n$  の有限部分集合への関数 (遷移関数) である。

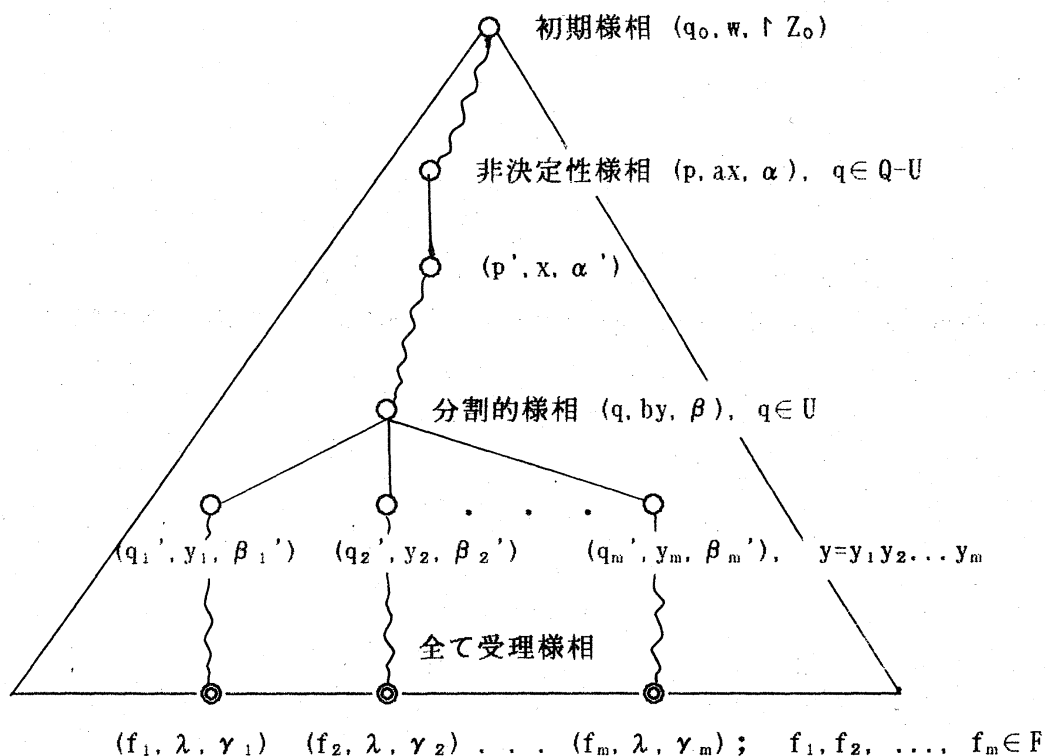
$w \in \Sigma^*$  が  $M$  に受理されるのは、次のような計算木  $T$  が存在する場合である：

- (i)  $T$  の各節には  $M$  の様相  $(Q \times \Sigma^* \times \Gamma^* \cup \{-1, 1\} \times \Gamma^*)$  の元がラベル付けされている。
- (ii)  $T$  の根のラベルは初期様相  $(q_0, w, \uparrow Z_0)$  である。
- (iii)  $T$  のどの葉のラベルも受理様相  $(F \times \{\lambda\} \times (\Gamma^* \cup \{-1, 1\} \times \Gamma^*))$  の元である。
- (iv)  $\pi$  をラベル  $(p, ax, Z_1 \dots \uparrow Z_1 \dots Z_k)$  を持つ、 $T$  の内点とする。ただし、 $a \in \Sigma \cup \{\lambda\}$ 、 $x \in \Sigma^*$ 、各  $Z_j \in \Gamma$  とする。次の2通りの場合がある。
  - (iv-1)  $p \in Q - U$  のとき 非決定性モード という。  $\pi$  の子供は一人だけで、そのラベルは
    - (a: プッシュダウンモード)  $(q, x, \uparrow \gamma Z_2 \dots Z_k)$  である。ただし、 $i=1$  かつ  $\delta(p, a, Z_1)$  が  $Q \times \Gamma^*$  の元  $(q, \gamma)$  を含んでいるとする。

(b: スタック読みモード)  $(q, x, Z_1 \dots \uparrow Z_{i+d} \dots Z_k)$  である。ただし  $\delta(p, a, Z_i)$  が  $Q \times \{-1, 1\}$  の元  $(q, d)$  を含んでいて、 $1 \leq i+d \leq k$  であるとする。

(iv-2)  $p \in U$  のとき 分割モード という。  $\pi$  の子供は  $m$  人で、それ等のラベルは左から順に  $(q_1, x_1, \alpha_1), \dots, (q_m, x_m, \alpha_m)$  である。ただし、 $\delta(p, a, Z_i) = \{(q_1, r_1), \dots, (q_m, r_m)\}$  で、かつ、各  $\alpha_j$  は  $(q_j, r_j)$  が  $Q \times \Gamma^*$  の元であるか  $Q \times \{-1, 1\}$  の元であるかに従って、(iv-1) (a) または (iv-1) (b) のように定義されたものであり、 $x = x_1 \dots x_m$  でなければならない。

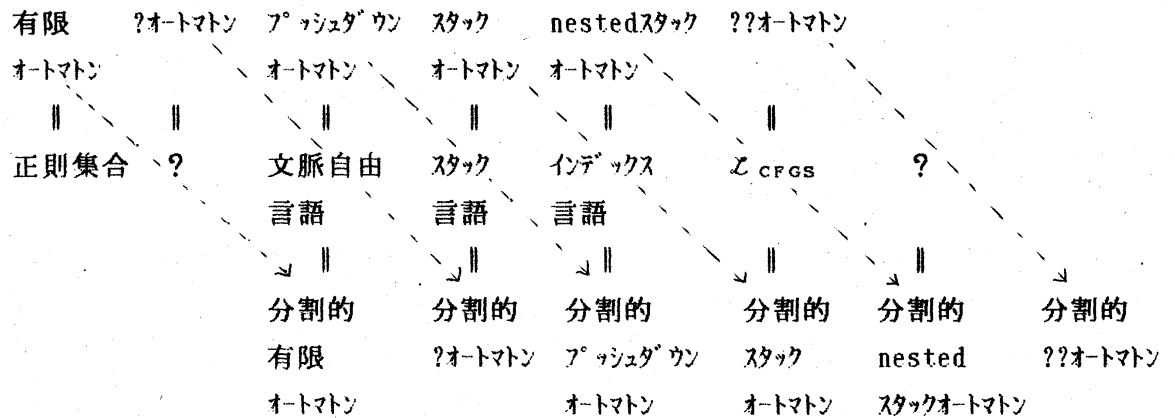
分割的オートマトンは alternating オートマトンの 1 変形であると考えてよい。 partitioning state ( $U$  の元) は alternating state に相当し、 $Q-U$  の元は existential state に対応する。オートマトンが existential state にいるときは、その様相から移り得る次の様相は  $\delta$  によって与えられる遷移可能な様相の中から非決定的に 1 つ選ばれる (iv-1: 非決定性モード) のに対し、partitioning state を持つ様相からは、 $\delta$  によって与えられる遷移可能な様相の全てに計算が引き継がれ、以後計算は平行に進められるという点は alternating オートマトンと同じであるが、partitioning な様相 (計算木における親様相) からの遷移の際入力記号列のまだ読まれていない部分が分割されて子様相達に渡される点 (iv-2: 分割モード) が partitioning オートマトンたる名前の所以である。平行に進められた計算の全てが受理様相に至ったとき、入力記号列は受理される (iii)。



$M$ が受理する語の全体を  $L(M)$  で表し、PSA言語と呼ぶ。上記は、スタックオートマトンに対してそのpartitioning版を定義したものであるが、同様にして、いろいろのオートマトンに対してそのpartitioning版が定義出来る。

定理3 PSA言語のクラスは  $\mathcal{L}_{CFG}$  と一致する。

同様に、分割的counter machine言語のクラス= $\mathcal{L}_{CFG}$  が示される。池川 [6] の結果 (分割的有限オートマトンの受理する言語のクラス= $\mathcal{L}_{CFG}$ 、分割的プッシュダウンオートマトンの受理する言語のクラス= $\mathcal{L}_{CFGP}$  (インデックス言語のクラス)) と合わせると、partitioningによってオートマトンの受理能力が1つシフトアップすることが分かる。



### 文献

1. A. V. Aho, Indexed grammars - an extension of context-free grammars, J. ACM 15 (1968), 647-671.
2. A. V. Aho, Nested stack automata, J. ACM 16 (1969), 383-406.
3. R. V. Book and S. Ginsburg, Multi-stack-counter languages, Math. Systems Theory 6 (1972), 37-48.
4. S. Ginsburg, S. A. Greibach and M. A. Harrison, One-way stack automata, J. ACM 14 (1967), 389-418.
5. 西沢輝泰, General indexed grammarとmonitored pushdown stack acceptor, 日本数学会応用数学分科会, 1979年4月.
6. M. Ikekawa, Modified one-way alternating pushdown automata and indexed languages, Trans. IECE Japan E69 (1986), 1213-1216.
7. A. Salomaa, "Formal Languages", Academic Press, 1973.