

多重文脈自由文法の所属問題に 対する並列アルゴリズム

中西 隆一

関 浩之

嵩 忠雄

Ryuichi Nakanisi

Hiroyuki Seki

Tadao Kasami

大阪大学基礎工学部情報工学科

あらまし 自然言語の構文記述のための形式文法として, 多重文脈自由文法 (m cfg) が提案されている. m cfg の生成能力は文脈自由文法よりも真に大きく, 文脈規定文法よりも真に小さい. 本稿では, m cfg の所属問題に対する P-RAM 上での並列アルゴリズムを提案する. 本アルゴリズムは時間計算量が入力系列長 n に対して $O(\log^2 n)$, プロセッサ数が n の多項式オーダーである.

1. まえがき

自然言語の構文記述のための形式文法として, 多重文脈自由文法 (m cfg) が提案されている^[3]. 文脈自由文法 (cfg) では各非終端記号は終端記号列を生成するが, m cfg では各非終端記号は終端記号列の組を生成する. また m cfg では, respectively 構文や倒置文のような互いに割り込んだ構文を自然に記述できる.

並列計算機のモデルとして P-RAM 及び W-RAM が提案されている^[1]. これらは複数の RAM^[2] (Random Access Machine) が共有メモリを持ち, 同期して動作するモデルである. P-RAM 及び W-RAM における個々の RAM はプロセッサと呼ばれる.

本稿では, m cfg の所属問題に対する P-RAM 上での並列アルゴリズムを提案する. 本アルゴリズムは P-RAM 上で時間計算量が入力系列長 n に対して $O(\log^2 n)$, プロセッサ数が n の多項式オーダーである.

2. 多重文脈自由文法

多重文脈自由文法 (multiple context-free grammar, m cfg と略記) は以下の (1)~(4)を満たす 4 字組 $G = (N, T, P, S)$ で定義される. 理解を容易にするため, 文献[3][4]における表記法を変更している.

(1) N は非終端記号の有限集合である. ただし各 $A \in N$ について, A の次元と呼ばれる正整数 $d(A)$ が定まっている. $d(A)$ は, A が終端記号列の何字組を生成するかを示す. また $d(A)$ の各 $A \in N$ についての最大値を G の次元と呼ぶ. 各 $A \in N$ を $(A^{[1]}, \dots, A^{[d(A)]})$ と書くこともある. $A^{[1]}, \dots, A^{[d(A)]}$ を A の分句記号と呼ぶ. またすべての非終端記号の分句記号からなる集合を N_{CMP} と書く.

(2) T は終端記号の有限集合である.

(3) P は生成規則の有限集合である. 生成規則を単に規則と呼ぶことがある. 生成規則は $(A^{[1]}, \dots, A^{[d(A)]}) \rightarrow (\alpha_1, \dots, \alpha_{d(A)})$ のような形をしている ($\alpha_1, \dots, \alpha_{d(A)} \in (N_{\text{CMP}} \cup T)^*$). 左辺は非終端記号 $A = (A^{[1]}, \dots, A^{[d(A)]})$ であり,

右辺は終端記号と分句記号からなる系列の $d(A)$ 字組である。 A をその生成規則の左辺の非終端記号と呼ぶ。 さらに任意の $p \in P$ に対して以下の条件が成立しなければならない。

[条件 a] 「任意の非終端記号 B について、 任意の分句記号 $B^{[i]}$

$(1 \leq i \leq d(B))$ は p の右辺のいずれかの成分中に高々1回しか現れない。」

非終端記号 B の分句記号が規則 p の右辺に現れるとき、 B を p の右辺に現れる非終端記号と呼ぶ。 また右辺のどの成分にも分句記号が現れないような規則を終端規則、 それ以外の規則を非終端規則と呼ぶ。

(4) S は始端記号と呼ばれる非終端記号で、 その次元は1である。 $\square$

次に G において各非終端記号 A が生成する語の集合 $L_G(A)$ を以下の(1)~(3)で定義する。

(1) 終端規則 $A \rightarrow (\alpha_1, \dots, \alpha_{d(A)})$ ($\alpha_1, \dots, \alpha_{d(A)} \in T^*$) が P 中に存在するならば $(\alpha_1, \dots, \alpha_{d(A)}) \in L_G(A)$ 。

(2) p を P に属する非終端規則とし、 p の左辺の非終端記号を A 、 p の右辺に現れるすべての非終端記号を $B_1, \dots, B_n$ とする。 $(\alpha_{1,1}, \dots, \alpha_{d(B_1),1}) \in L_G(B_1)$, $\dots$, $(\alpha_{1,n}, \dots, \alpha_{d(B_n),n}) \in L_G(B_n)$ と仮定する。 p の右辺に現れるすべての分句記号 $B_i^{[j]}$ ($1 \leq i \leq n$, $1 \leq j \leq d(B_i)$) に $\alpha_{j,i}$ を代入して得られる $d(A)$ 字組も $L_G(A)$ に属する。

(3) (1), (2) を有限回適用して得られるもののみが $L_G(A)$ に属する。

また、 G の生成する言語 $L(G)$ を $L(G) \triangleq L_G(S)$ と定義する。

ここで、 以下の補題1が成り立つ。

[補題1] 空系列を生成しない任意の mcfg $G = (N, T, P, S)$ に対して、 以下の条件1~6を満たす、 G と弱等価な mcfg G_m が構成可能である。

[条件1] 任意の $p \in P$ 、 $A \in N$ に対して、 p の右辺のいずれかの成分中に A の分句記号 $A^{[i]}$ ($1 \leq i \leq d(A)$) が現れるならば、 $A^{[1]}, \dots, A^{[d(A)]}$ はいずれも p の右辺のいずれかの成分中に現れる。

[条件2] 任意の規則において、 右辺のどの成分も空系列ではない。

[条件3] 任意の終端規則において、 左辺の非終端記号の次数及び右辺の系列長はともに1である。

[条件4] 非終端規則の右辺には終端記号は現れない。

[条件5] 任意の規則について、 その右辺に現れる非終端記号は高々2個である(分句記号の数ではないことに注意)。

[条件6] 左辺の非終端記号 A が一致する、 任意の相異なる2つの規則を $p_1, p_2 \in P$ とする。 どの i ($1 \leq i \leq d(A)$) についても、 p の右辺の第 i 成分と q の右辺の第 i 成分は一致しない。 $\square$

mcfg $G = (N, T, P, S)$ から派生するcfgを、 $G' = (N_{CMP}, T, P',$

$S^{[1]}$) と定義する。ただし、 P' は以下の (1), (2) のように定義される。

- (1) $A \rightarrow (\alpha_1, \dots, \alpha_{d(A)}) \in P$ ならば、 $A^{[1]} \rightarrow \alpha_1, \dots, A^{[d(A)]} \rightarrow \alpha_{d(A)} \in P'$ 。なお、これらの $d(A)$ 個の規則を、 G' において互いに兄弟であると呼ぶ。
 (2) P' に含まれる規則は (1) で得られるもののみ。

明らかに $L(G) \subseteq L(G')$ である。

補題 1 で構成された G_m に対し、それから派生する cfg を G_m' とすると、条件 6 より、以下の性質 1 が明らかに成立する。

[性質 1] G_m' の任意の規則は G_m のどの規則から構成されたか一意に決定できる。□

次に、派生する cfg の構文木中の頂点間の分句関係を、以下の条件を満たす最小の関係と定義する。「派生する cfg の構文木中の 2 頂点 p, q について、 p の親と q の親が同じか、または親同士が分句関係にあり、かつ p と q につけられた $\lambda \wedge \mu$ が、同一の非終端記号の分句記号であるならば、 p と q もまた分句関係にある。」

mcfg G から派生する cfg を G' とする。 $L(G')$ に属する語 w の G' における構文木を t とする。 t 中の分句関係にある任意の 2 頂点において、適用されている規則が兄弟であるとき、その構文木で表される生成は G で許され、 $w \in L(G)$ が成り立つ。そこでこのとき、 t を G における w の構文木であると定義する。

補題 1 を満たす mcfg G_m に対しては、性質 1 が成り立つことより、 t の各頂点において適用されている各規則が兄弟であるか否かが容易に判定できる。

3. P-RAM 及び W-RAM

P-RAM 及び W-RAM は複数個の RAM 及び共有メモリからなる並列計算モデルである。P-RAM 及び W-RAM における個々の RAM はプロセッサと呼ばれる。各プロセッサは共有メモリを介して通信を行いつつ、同期して動作する。ただし共有メモリには以下の (a), (b) の 2 種類がある。

(a) 1 つの番地には、同時に高々 1 つのプロセッサしか書き込みできない。

(b) 各番地がもつ値は true または false だけであり、初期状態ではその値は false である。また各プロセッサが書き込める値は true だけである。そして 1 つのアドレスに同時に複数個のプロセッサが書き込みできる。

((a), (b) のいずれにおいても、読み取りは同時に複数のプロセッサが行ってよい)。

W-RAM は (a), (b) の両方の共有メモリを持ち、P-RAM は (a) の共有メモリのみを持つ。

P-RAM, W-RAM において各プロセッサは、初期値として入力系列を分割して与えられ、その後同一のプログラムを実行する。

以下の補題 2 が成り立つことが知られている^[1]。

[補題 2] W-RAM 上の、時間計算量 $T(n)$ 、プロセッサ数 $P(n)$ の並列アルゴリズム A が存在するとする。任意の入力に対する出力が A と等しく、時間計算量 $T(n) \cdot \log n$ 、プロセッサ数 $P(n)$ の P-RAM 上の並列アルゴリズムが A から構成可能である (ただし、 n は入力系列長)。□

なお、アルゴリズムの記述であるが、通常のPASCAL風記述に加えてfor all $x_1, x_2, \dots, x_n$ s.t. condition in parallel do statementという記述を用いる。これはconditionで表される条件を満たすすべての変数の組 $x_1, x_2, \dots, x_n$ について並列にstatementで表される命令を実行することを示す（従って、conditionを満たす変数の組の数だけプロセッサを必要とする）。

また、アルゴリズム中で複数個のプロセッサが同時に同一の変数の値を変更しようとする可能性がある場合、(b)の制約より変更後の値はtrueでなければならない。このような形で記述されたアルゴリズムはW-RAM上で実現できる^[1]。

4. mcfgの並列認識アルゴリズム

簡単のため、与えられたmcfg G は、空系列を生成せず、かつ、補題1の条件1~6を満たすと仮定する。 G の次元を m とする。 G から派生するcfg G' を構成し、更に G' と弱等価で、各規則の右辺の系列長が1または2であるようなcfg G'' を構成しておく（構成法は後述）。補題1の条件3, 4, 5より、 G について以下の性質2が成立する。

[性質2] 任意の規則において、右辺の系列長の全成分の総和は高々 $2m$ である。□

本アルゴリズムの概要は以下の(1)~(3)の通り。

- (1) G'' について入力系列 w を並列構文解析し、その結果から G' の各非終端記号が w のどの部分列を生成できるかを求める（4.1のderived-cfg-check）。
- (2) (1)の結果から、互いに兄弟である規則（2.参照）が適用されている頂点の組をすべて求める。さらに、このような頂点の組 X, Y で、次の条件を満たすものを求める。「 X の各元が互いに分句関係にあるならば、 Y の元も互いに分句関係にある」これらの結果は有向グラフ $U_{G,w}$ として表される（4.2.1のmake-V及びmake-E）。
- (3) $U_{G,w}$ に基づき、mcfg G における w の構文木が存在するか否かを判定する。すなわち、mcfgにおける構文木の定義（2.参照）に従い、cfg G' における構文木で次の条件を満たすものが存在するかどうかを判定する：「互いに分句関係にある頂点同士には互いに兄弟である規則が適用されている」（4.2.2のbrother-check）。

4.1 派生するcfgの構文解析

$w = w_1 w_2 \dots w_n$ ($w_1, \dots, w_n$ は終端記号) に対して、 $w[i:j] \triangleq w_{i+1} \dots w_j$ ($0 \leq i < j \leq n$)、 $w[i:i] \triangleq \varepsilon$ ($0 \leq i \leq n$) と定義する。

$G' = (N, T, P, S)$ をcfg、 w を終端記号列とする。 w の系列長を n とする。 $\langle G', w \rangle$ に対する部分列導出、実現可能性、 $\vdash$, $\models$ を以下のように定義する。

部分列導出とは、任意の3字組 $(A \rightarrow B_1 \dots B_k, i, j)$ ($A \in N, A \rightarrow B_1 \dots B_k \in P, 0 \leq i \leq j \leq n$) である。 G' において実際に $A \Rightarrow B_1 \dots B_k \triangleq w[i:j]$ が成り立つときかつそのときのみ部分列導出 $(A \rightarrow B_1 \dots B_k, i, j)$ は実現可能であるという（ $\triangleq$ は関係 $\Rightarrow$ の反射推移閉包を表す）。また、関係 $\vdash$ を次のように定義する。

「 $x, y_1, \dots, y_k$ を部分列導出とする。 $y_1, y_2, \dots, y_k \vdash x \Leftrightarrow y_1, y_2, \dots, y_k$ の並べ替え $y_1', y_2', \dots, y_k'$ が存在して、 $x = (A \rightarrow B_1 B_2 \dots B_k, i, j)$, $y_1' = (B_1 \rightarrow \dots, i, h_1)$, $y_2' = (B_2$

$\rightarrow \dots, h_1, h_2), \dots, y_k' = (B_k \rightarrow \dots, h_{k-1}, j)$ となっている (ただし $k \geq 1$ かつ $i \leq h_1 \leq h_2 \leq \dots \leq h_{k-1} \leq j$).」

(定義より, $y_1, \dots, y_k \vdash x$ かつ $y_1, \dots, y_k$ が実現可能ならば x も実現可能である.)

関係 $\models$ を以下のように定義する.

「 x, y を部分列導出とする. $x \models y \iff$ 部分列導出 $x_0, x_1, \dots, x_q$ ($q \geq 0$) が存在して, $x = x_0, y = x_q$ かつ $x_i \vdash x_{i+1}$ ($0 \leq i \leq q-1$).」

以下の補題が成立する (証明は非終端記号の個数が定数個であることによる).
[補題3] y を任意の部分列導出とする. $x \models y$ である部分列導出 x , 及び $y \models z$ である部分列導出 z はいずれも入力系列 w に依存しない定数個しか存在しない. $\square$

G から派生する cfg を $G' = (N_{CMP}, T, P', S^{[1]})$ とする. G' と弱等価で, 各規則の右辺の系列長が常に 1 または 2 であるような cfg $G'' = (N', T, P'', S^{[1]})$ を以下の (1)~(3) のように構成する.

- (1) P' 中の規則のうち, 右辺の系列長が 2 以下のものは P'' に属する (G' に ϵ 規則が存在しないので, G'' にも ϵ 規則は存在しない).
- (2) 右辺の系列長が 3 以上の規則 $p: A \rightarrow B_1 B_2 \dots B_k$ が P' に属するとき, $p_k: A \rightarrow C_k B_k, p_{k-1}: C_k \rightarrow C_{k-1} B_{k-1}, \dots, p_2: C_3 \rightarrow B_1 B_2$ のすべてが P'' に属する. ただし, $C_3, \dots, C_k$ は新しい非終端記号.
- (3) P'' に属する規則は (1)(2) によるもののみであり, N' に属する非終端記号は N_{CMP} に属するものと (2) で導入したもののみ.

G'' に属する規則は G' のどの規則から構成されたかが一意に決定可能である.

以下, 本アルゴリズムに対する入力を $w = w_1 w_2 \dots w_n$ ($w_1, \dots, w_n$ は終端記号) とする.
 $\langle G'', w \rangle$ から以下のような有向グラフ $U''_{G, w} = (V'', E'')$ を定義する.

- (1) V'' は $\langle G'', w \rangle$ に対するすべての部分列導出の集合 (規則の個数は w に依存しない定数なので, 合計 $O(n^2)$ しか存在しない).
- (2) $(x, y) \in E'' \iff$ 以下の (a), (b) のいずれかを x, y が満たす.
(a) 実現可能な部分列導出 z が存在して, $y, z \vdash x$
(b) $y \vdash x$

4.1.1 手続き derived-cfg-check

$U''_{G, w}$ における E'' の反射推移閉包 (COND), 及び各部分列導出の実現可能性 (REALIZABLE) を求める W-RAM 上の手続き derived-cfg-check を以下に示す (正当性は文献 [5] を参照されたい). COND, REALIZABLE の初期値はすべて false である.

入力 V'' , 出力は「受理」または「拒否」である. ただし REALIZABLE は後で $U''_{G, w}$ から $U_{G, w}$ (定義は 4.2.1.1 参照) を構成する際に用いられる.

procedure derived-cfg-check

0) **for all** x, y, i, A **s.t.** $0 \leq i < n, A \in N', y = (A \rightarrow w_{i+1}, i, i+1), y \models x$
 in parallel do REALIZABLE(x) $\leftarrow$ true

```

1) for all x,y s.t.  $y \models x$  in parallel do COND(x,y)=true
2) repeat [log n] times
    begin
    2.1) for all x,y,z s.t.  $y, z \vdash x$  and REALIZABLE(z)=true in parallel do
        for all u,v s.t.  $x \models u, v \models y$  in parallel do COND(u,v)=true
    2.2) for all x,y,z s.t. COND(x,z) and COND(z,y) in parallel do
        COND(x,y)=true
        for all x,y,z s.t. COND(x,z) and COND(z,y) in parallel do
            COND(x,y)=true
    2.3) for all x,y s.t. COND(x,y) and REALIZABLE(y)=true in parallel do
        REALIZABLE(x)=true
    end
3) REALIZABLE(x)=trueである部分列導出 $x=(S^{[1]} \rightarrow \dots, 0, n)$ が存在すれば
    「受理」, 存在しなければ「拒否」を出力
end of procedure derived-cfg-check

```

ステップ2.2では同一の文を2回繰り返して実行していることに注意されたい。

4.1.2 derived-cfg-checkの計算複雑度

部分列導出の総数が $O(n^2)$ であること等から、以下が成り立つ。

ステップ0 : $O(n)$ プロセッサ, 定数時間

($\because$ 実現可能になり得るのは、第1成分が終端規則で、第2成分+1 = 第3成分となっている部分列導出のみ)

ステップ1 : $O(n^2)$ プロセッサ, 定数時間

ステップ2.1 : $O(n^4)$ プロセッサ, 定数時間

(y, z が決まれば $y, z \vdash x$ であるような x は定数個しか存在しないため。)

ステップ2.2 : $O(n^6)$ プロセッサ, 定数時間

ステップ2.3 : $O(n^4)$ プロセッサ, 定数時間

ステップ3 : 定数プロセッサ, 定数時間

また、repeatループのためステップ2は $O(\log n)$ 回実行される。

以上よりderived-cfg-checkの時間計算量は(W-RAM上で) $O(\log n)$, プロセッサ数は $O(n^6)$ となる。

これと補題2より、derived-cfg-checkはP-RAM上で時間計算量 $O(\log^2 n)$, プロセッサ数 $O(n^6)$ で実現できる。

4.2 分句関係のチェック

4.2.1 $U_{G,w}$ の構成

4.2.1.1 $U_{G,w}$ の定義

$U''_{G,w}$ から $U_{G,w} = (V, E)$ を以下のように定義する。

(1) $\langle G', w \rangle$ の部分列導出のうちで実現可能なものの集合を V' とする (すなわち V' は, 補題 4 の (1) より, derived-cfg-check の停止時に $\text{REALIZABLE}(x) = \text{true}$ を満たすすべての x からなる集合). 次に V を, 以下の条件 8 を満たす V' の部分集合 X のすべてからなる族として定義する.

[条件 8] X の元のある数え上げ $x_1, \dots, x_h$ および $d(A) = h$ である $A \in N$ が存在して,

$$x_1 \text{ の第 1 成分の左辺} = A^{[1]},$$

:

$$x_h \text{ の第 1 成分の左辺} = A^{[d(A)]},$$

かつ $x_1, \dots, x_h$ の第 1 成分である生成規則は互いに兄弟である.」

(2) $(X, Y) \in E \iff X = \{x_1, \dots, x_h\}, Y = \{y_1, \dots, y_k\}$ は以下の (a) (b) のいずれかを満たす.

(a) $Z \in V$ が存在して,

$$\alpha_{1-1}, \alpha_{2-1}, \dots, \alpha_{k1-1} \vdash x_1,$$

:

$$\alpha_{1-h}, \alpha_{2-h}, \dots, \alpha_{kh-h} \vdash x_h.$$

ただし, 各 $\alpha_{i-j} (1 \leq j \leq h, 1 \leq i \leq k_j)$ は互いに異なり, かつ $Y \cup Z = \{\alpha_{1-1}, \alpha_{2-1}, \dots, \alpha_{kh-h}\}$ である.

このとき, $Y, Z \vdash X$ と定義する. また (X, Y) はタイプ **a** の辺であるという.

(b) $\alpha_{1-1}, \alpha_{2-1}, \dots, \alpha_{k1-1} \vdash x_1,$

:

$$\alpha_{1-h}, \alpha_{2-h}, \dots, \alpha_{kh-h} \vdash x_h.$$

ただし, 各 $\alpha_{i-j} (1 \leq j \leq h, 1 \leq i \leq k_j)$ は互いに異なり, かつ $Y = \{\alpha_{1-1}, \alpha_{2-1}, \dots, \alpha_{kh-h}\}$ である. このとき, $Y \vdash X$ と定義する. また, (X, Y) はタイプ **b** の辺であるという.

定義より, $(X, Y) \in E$ かつ X の各元が互いに分句関係にあるならば, Y の元も互いに分句関係にある.

また $Z_i (0 \leq i \leq q, q \geq 0) \in V$ が存在して, $X = Z_0, Y = Z_k$ かつ $Z_i \vdash Z_{i+1} (0 \leq i \leq q-1)$ が成立するときかつそのときのみ $X \vdash Y$ と書く.

ここで以下の補題が成立する (証明は非終端記号が定数個であることによる).

[補題 5] $X \in V$ とする. $Y \vdash X$ である $Y \in V$, 及び $X \vdash Z$ である $Z \in V$ はどちらも入力 w に依存しない定数個しか存在しない. □

以下では, $U_{G,w} = (V, E)$ を構成するアルゴリズムを示す.

4.2.1.2 手続き make-V

以下に, V' 及び手続き derived-cfg-check 実行後の変数 REALIZABLE の内容から V を求める手続き make-V を示す. make-V は明らかに $|V| : C_m = O(n^{2m})$ プロセッサ, 定数時間で実現できる.

procedure make-V(V', REALIZABLE)

V = {}

for all $x_1', \dots, x_m'$ **s. t.** $x_1', \dots, x_m' \in V'$ **in parallel do**

for all $x_1, \dots, x_h$ **s. t.** $[X = \{x_1, \dots, x_h\} \subseteq \{x_1', \dots, x_m'\}]$ **かつ** X は条件 8 を満たす **in parallel do** $V = V \cup \{X\}$

end of procedure make-V

4.2.1.3 手続き make-E

まず, V' の部分集合 $V_s = \{v_1, v_2, \dots, v_h\}$ を引数とする手続き grouping を以下のように定義する.

procedure grouping(V_s)

$\{v_1 \sim v_h\}$ をその第 1 成分の左辺の非終端記号がどの非終端記号の分句記号であるかによって分類し, その各々の $\mathcal{N}^*$ を集合 $S_1, S_2, \dots, S_k$ とする. すべての i について S_i が条件 8 を満たすとき $\{S_1, \dots, S_k\}$ を, そうでなければ $\emptyset$ を出力する.

end of procedure grouping

[補題 6] 手続き grouping は定数プロセッサ, 定数時間で実現可能である.

(証明) grouping が入力 V_s に対して空集合以外を出力するならば, $|V_s| \leq m \cdot |N|$ であることによる. □

V, V' から $U_{G,w}$ を構成する手続き make-E を以下に示す. 変数 $EDGE'(X, Y, c)$ ($X, Y \in V, c \in \{a, b\}$, 初期値は false) は「アルゴリズム停止時に $EDGE'(X, Y, c) = \text{true}$ $\Leftrightarrow (X, Y) \in E$ かつ辺 (X, Y) はタイプ c である」を満たす.

procedure make-E(V, V')

for all $X \in V$ **in parallel do**

for all $x_1, \dots, x_h \in X$ **in parallel do** ($x_1 \sim x_h$ は条件 8 の順であるとする)

for all $y_{11}, \dots, y_{1k_1}, \dots, y_{h1}, \dots, y_{hk_h} \in V'$ **s. t.**

$y_{11}, \dots, y_{1k_1} \vdash x_1, \dots, y_{h1}, \dots, y_{hk_h} \vdash x_h$ **in parallel do**

begin

1) $S = \text{grouping}(\{y_{ij} \mid 1 \leq i \leq h, 1 \leq j \leq k_i\})$

2) **for all** $Y \in S$ **in parallel do**

if $|S| = 2$ **then** $EDGE'(X, Y, a) = \text{true}$ **else if** $|S| = 1$ **then**

$EDGE'(X, Y, b) = \text{true}$

end

end of procedure make-E

正当性は $U_{G,w}$ の定義より明らか. 以下では make-E の計算複雑度を解析する.

$|V| = O(n^{2m})$ より,

ステップ 1: 定数プロセッサ, 定数時間. ただしステップ 1 を囲む for 文により, 総プロセッサ数は $O(n^{6m})$ となる.

($\because h \leq m$, y_{ij} の合計は性質2より $2m$ 以下, $|V'| = O(n^2)$, かつ y_{ij} がすべて決まれば X は高々定数個しか存在しない)

ステップ2: 定数プロセッサ, 定数時間. ただしfor文によりステップ1と同様にプロセッサ数は $O(n^{6m})$ となる. ($\because |S| \leq 2$)

故にmake-EはW-RAM上で $O(n^{6m})$ プロセッサ, 定数時間で実現でき, それ故補題2よりP-RAM上で $O(n^{6m})$ プロセッサ, $O(\log n)$ 時間で実現できる.

4.2.2 mcfgにおける構文木の存在のチェック

下の条件9を満たす任意の T を考える.

[条件9] 以下の(1)~(4)がすべて成立する.

(1) T は $U_{G,w}$ の部分グラフである2分木である.

(2) T の根は $\{(S^{[1]} \rightarrow \dots, 0, n)\}$ である.

(3) T の葉のすべての元の第1成分は終端規則である.

(4) T 中の任意の内部頂点 X は以下の①②のいずれかを満たす.

① X の出次数が2, かつ X の子 Y, Z について $Y, Z \models X$.

② X の出次数が1, かつ X の子 Y について $Y \models X$. □

条件9を満たす T は G における S からの w の生成を表しており, このような T が存在することと $w \in L(G)$ は同値である.

4.2.2.1 手続きbrother-check

条件9を満たす T が存在するか否かを判定する手続きbrother-checkは以下の通り.
(TRANS, BROTHERの初期値はすべてfalseである)

入力 $U_{G,w} = (V, E)$ であり, 出力は「受理」または「拒否」である.

procedure brother-check(V, E)

0) **for all** $X, Y \in V$ **s.t.** [Y の元の第1成分は終端規則] **and** $Y \not\models X$

in parallel do BROTHER(X) $\leftarrow$ true

1) **for all** X, Y **s.t.** $Y \models X$ **in parallel do** TRANS(X, Y) = true

2 **repeat** $\lceil \log n \rceil$ **times**

begin

2.1) **for all** X, Y, Z **s.t.** $Y, Z \models X$ **and** BROTHER(Z) = true **in parallel do**

for all W_1, W_2 **s.t.** $X \not\models W_1, W_2 \models Y$ **in parallel do**

TRANS(W_1, W_2) = true

2.2) **for all** X, Y, Z **s.t.** TRANS(X, Z) **and** TRANS(Z, Y) **in parallel do**

TRANS(X, Y) = true

for all X, Y, Z **s.t.** TRANS(X, Z) **and** TRANS(Z, Y) **in parallel do**

TRANS(X, Y) = true

2.3) **for all** X, Y **s.t.** TRANS(X, Y) = true **and** BROTHER(Y) = true

in parallel do BROTHER(X) = true

end

3) $BROTHER(X)=true$ である $X=\{(S \rightarrow \dots, 0, n)\} \in V$ が存在すれば「受理」, 存在しなければ「拒否」を出力

end of procedure brother-check

4.2.2.3 brother-checkの解析

$|V| = O(n^{2m})$ なので, W-RAM上では

ステップ0 : $O(n)$ プロセッサ, 定数時間

($\because$ 第1成分が終端規則でかつ実現可能な部分列導出は $O(n)$ 個しか存在しない)

ステップ1 : $O(n^{2m})$ プロセッサ, 定数時間

ステップ2.1 : $O(n^{4m})$ プロセッサ, 定数時間

($\because y, z$ が決まれば $y, z \models x$ であるような x は定数個しか存在しない)

ステップ2.2 : $O(n^{6m})$ プロセッサ, 定数時間

ステップ2.3 : $O(n^{4m})$ プロセッサ, 定数時間

ステップ3 : 定数プロセッサ, 定数時間

ただしrepeatループのためステップ2, 3, 4は $O(\log n)$ 回実行される.

故にbrother-checkの時間計算量は $O(\log n)$, プロセッサ数は $O(n^{6m})$ となる.

これと補題2よりP-RAM上で時間計算量 $O(\log^2 n)$, プロセッサ数 $O(n^{6m})$ で実現できる.

以上の結果より, 次の定理及び系が得られる.

[定理] G を条件1~6を満たす任意のmcfgとする. 与えられた w が $L(G)$ に属するか否かは, 入力系列長 n に対して, P-RAM上で $O(n^{6m})$ プロセッサ, $O(\log^2 n)$ 時間で判定できる. $\square$

[系] G を任意のmcfgとする. 与えられた $w \in L(G)$ かどうかは, 入力系列長 n に対して, P-RAM上で n の多項式プロセッサ, $O(\log^2 n)$ 時間で判定できる. $\square$

文 献

- [1] A. Gibbons and W. Rytter: "Efficient Parallel Algorithms", Cambridge University Press, New York, USA, 1988.
- [2] J. Hopcroft and J.D. Ullman: "Introduction to Automata Theory, Languages, and Computation", Addison-Wesley, Reading, Mass., U.S.A., 1979.
- [3] 嵩 忠雄, 関 浩之, 藤井 護: "一般化文脈自由文法と多重文脈自由文法", 信学論(D), J71-D, 5, pp. 758-765, 昭63-05.
- [4] 嵩 忠雄, 関 浩之, 藤井 護: "Head Languageおよび多重文脈自由言語の所属問題", 信学論(D), J71-D, 6, pp. 935-941, 昭63-06.
- [5] 中西 隆一: "語彙機能文法の生成能力及び多重文脈自由文法に関する研究", 大阪大学大学院基礎工学研究科修士学位論文, 1991-02.