数理解析研究所講究録
第754巻 1991年 186-195

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（mcfg）が提案されている．mcfg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，mcfg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（mcfg）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，mcfgでは各非終端記号は終端記号列の組を生成する．またmcfg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]．これらは複数個の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，mcfg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の多項式オーダである．

2．多重文脈自由文法

多重文脈自由文法（multiple context-free grammar,mcfg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∈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∈Nについての最大値をGの**次元**と呼ぶ．各A∈Nを（$A^{[1]}$，…，$A^{[d(A)]}$）と書くこともある．$A^{[1]}$，…，$A^{[d(A)]}$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$(A^{[1]},\cdots,A^{[d(A)]})\rightarrow(\alpha_1,\cdots,\alpha_{d(A)})$のような形をしている（$\alpha_1,\cdots,\alpha_{d(A)}\in(N_{CMP}\cup T)^*$）．左辺は非終端記号A＝（$A^{[1]}$，…，$A^{[d(A)]}$）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∈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．□

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$A \rightarrow (\alpha_1, \cdots, \alpha_{d(A)})$（$\alpha_1, \cdots, \alpha_{d(A)} \in T^*$）がＰ中に存在するならば$(\alpha_1, \cdots, \alpha_{d(A)}) \in L_G(A)$．

(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$B_1, \cdots, B_n$とする．$(\alpha_{1,1}, \cdots, \alpha_{d(B1),1}) \in L_G(B_1)$，…，$(\alpha_{1,n}, \cdots, \alpha_{d(Bn),n}) \in L_G(B_n)$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∈Ｐ，Ａ∈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，$A^{[1]}$，…，$A^{[d(A)]}$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．□

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→（α_1, …, $\alpha_{d(A)}$）∈Pならば，$A^{[1]}\rightarrow\alpha_1$, …, $A^{[d(A)]}\rightarrow\alpha_{d(A)}$ ∈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）⊆L（G'）である．

補題1で構成されたGmに対し，それから派生するcfgをGm'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]Gm'の任意の規則はGm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につけられたﾗﾍﾞﾙが，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∈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 Gm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ﾌﾟ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）・logn，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** $x_1, x_2, \cdots, x_n$ **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$x_1, x_2, \cdots, x_n$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ｸﾞﾗﾌ$U_{G,w}$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

$w=w_1w_2\cdots w_n$($w_1, \cdots, w_n$は終端記号)に対して,$w[i:j] \triangleq w_{i+1}\cdots w_j$（$0 \leq i < j \leq n$）, $w[i:i] \triangleq \varepsilon$（$0 \leq i \leq n$）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.<G',w>に対する部分列導出,実現可能性,$\vdash$,$\vdash^*$を以下のように定義する.

部分列導出とは,任意の3字組$(A\rightarrow B_1\cdots B_k, i, j)$($A\in N, A\rightarrow B_1\cdots B_k\in P, 0\leq i\leq j\leq n$)である. G'において実際に$A \Rightarrow B_1\cdots B_k \triangleq w[i:j]$が成り立つときかつそのときのみ部分列導出（$A\rightarrow B_1\cdots B_k$, 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.「$x, y_1, \cdots, y_k$を部分列導出とする.$y_1, y_2, \cdots, y_k \vdash x$ <=> $y_1, y_2, \cdots, y_k$の並べ替え$y_1', y_2', \cdots, y_k'$が存在して,$x=(A\rightarrow B_1B_2\cdots B_k, i, j)$, $y_1'=(B_1\rightarrow\cdots, i, h_1)$, $y_2'=(B_2$

→…, h_1, h_2),…, y_k'=(B_k→…, h_{k-1}, j)となっている(ただしk≧1かつi≦h_1≦h_2≦…≦h_{k-1}≦j).」

(定義より, y_1,…, y_k⊢xかつy_1,…, y_kが実現可能ならばxも実現可能である.)

関係⊢*を以下のように定義する.

「x, yを部分列導出とする. x⊢* y <=> 部分列導出x_0, x_1, …, x_q (q≧0)が存在して, x=x_0, y=x_qかつx_i⊢x_{i+1}(0≦i≦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⊢* yである部分列導出x,及びy⊢* 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. □

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→B_1B_2…B_kがP'に属するとき, p_k: A→C_kB_k, p_{k-1}: C_k→$C_{k-1}B_{k-1}$, …, p_2: C_3→B_1B_2のすべてがP"に属する. ただし, C_3, …, 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,本ｱﾙｺﾞﾘｽﾞﾑに対する入力をw=w_1w_2…w_n(w_1,…,w_nは終端記号)とする. <G",w>から以下のような有向ｸﾞﾗﾌ$U"_{G,w}$=(V", E")を定義する.

(1)V"は<G", w>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)∈E" <=>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⊢x

(b)y⊢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.

入力は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≦i<n, A∈N',y=(A→w_{i+1},i,i+1),y⊢* x
     in parallel do  REALIZABLE(x)←true
```

1) **for all** x,y **s.t.** $y \vdash^{*} x$ **in parallel do** COND(x,y)=true
2) **repeat** ⌈logn⌉ **times**
begin
2.1) **for all** x,y,z **s.t.** y,z⊢x and REALIZABLE(z)=true **in parallel do**
for all u,v **s.t.** $x \vdash^{*} u, v \vdash^{*} y$ **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$x=(S^{[1]} \rightarrow \cdots, 0, n)$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
（∵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⊢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（logn）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（logn），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)＜G'，w＞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REALIZABLE(x)=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$x_1, \cdots, x_h$および$d(A)=h$である$A \in N$が存在して，

x_1の第1成分の左辺＝$A^{[1]}$，

:

x_hの第1成分の左辺＝$A^{[d(A)]}$，

かつx_1，…，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）∈E　＜＝＞　X＝{x_1，…，x_h}，Y＝{y_1，…，y_k}は以下の(a)(b)のいずれかを満たす．

(a)Z∈Vが存在して，

$$\alpha_{1\_1},\ \alpha_{2\_1},\ \cdots,\ \alpha_{k1\_1} \vdash x_1,$$

:

$$\alpha_{1\_h},\ \alpha_{2\_h},\ \cdots,\ \alpha_{kh\_h} \vdash x_h.$$

ただし，各$\alpha_{i_j}(1 \leqq j \leqq h, 1 \leqq i \leqq k_j)$は互いに異なり，かつY∪Z＝{$\alpha_{1_1}$，$\alpha_{2_1}$，…，$\alpha_{kh_h}$}である．

このとき，Y，Z⊢Xと定義する．また(X，Y)は**タイプaの辺**であるという．

(b)$\alpha_{1\_1},\ \alpha_{2\_1},\ \cdots,\ \alpha_{k1\_1} \vdash x_1,$

:

$\alpha_{1\_h},\ \alpha_{2\_h},\ \cdots,\ \alpha_{kh\_h} \vdash x_h.$

ただし，各$\alpha_{i_j}(1 \leqq j \leqq h, 1 \leqq i \leqq k_j)$は互いに異なり，かつY＝{$\alpha_{1_1}$，$\alpha_{2_1}$，…，$\alpha_{kh_h}$}である．このとき，Y⊢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）∈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≦i≦q，q≧0）∈Vが存在して，X＝Z_0，Y＝Z_kかつ$Z_i \vdash Z_{i+1}$（0≦i≦q－1）が成立するときかつそのときのみX⊩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∈Vとする．Y⊩Xであるy∈V，及びX⊩Zであるz∈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$|U| C_m = O(n^{2m})$プロセッサ，定数時間で実現できる．

```
procedure make-V(V',REALIZABLE)
  V={}
  for all x1',…,xm' s.t. x1',…,xm'∈V' in parallel do
    for all x1,…,xh s.t. [X={x1,…,xh}⊆{x1',…,xm'}かつXは条件8を
    満たす] in parallel do V=V∪{X}
end of procedure make-V
```

4.2.1.3　手続きmake-E

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

```
procedure grouping(Vs)
```

{v_1～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，その各々のｸﾞﾙｰﾌﾟを集合$S_1,S_2,\cdots,S_k$とする．すべてのiについてS_iが条件8を満たすとき{$S_1,\cdots,S_k$}を，そうでなければφ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∈V,c∈{a,b},初期値はfalse)は「ｱﾙｺﾞﾘｽﾞﾑ停止時にEDGE'(X,Y,c)=true <=> （X，Y）∈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∈V in parallel do
  for all x1,…xh∈X in parallel do(x1～xhは条件8の順であるとする)
    for all y11,…,y1k1,…,yh1,…,yhkh∈V' s.t.
    y11,…,y1k1⊢x1,…,yh1,…,yhkh⊢xh in parallel do
    begin
1)    S=grouping({yij|1≦i≦h,1≦j≦ki})
2)    for all Y∈S in parallel do
        if |S|=2 then EDGE'(X,Y,a)=true  else if |S|=1 then
          EDGE'(X,Y,b)=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．

（∵ h ≦ 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．（∵ | S | ≦ 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}）ﾌﾟﾛｾｯｻ, O（logn）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]→…, 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．

①Ｘの出次数が２，かつＸの子Ｙ，ＺについてＹ，Ｚ⊨Ｘ．

②Ｘの出次数が１，かつＸの子ＹについてＹ⊨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∈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)

入力は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∈V s.t. [Yの元の第１成分は終端規則] and Y⋨X
   in parallel do  BROTHER(X)←true
1) for all X,Y s.t. Y⋨X  in parallel do TRANS(X,Y)=true
2  repeat ⌈logn⌉ times
   begin
2.1) for all X,Y,Z s.t. Y,Z⊨X and BROTHER(Z)=true in parallel do
       for all W1,W2 s.t. X⋨W1,W2⋨Y in parallel do
         TRANS(W1,W2)=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→…,0,n)}∈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

(∵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

（∵y，zが決まればy，z⊢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（logn）回実行される．

故にbrother-checkの時間計算量はO（logn），ﾌﾟﾛｾｯｻ数は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．□

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

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.