

属性文法の循環性検査の高速化について

群馬大学 小池 博 (Hiroshi KOIKE)
群馬大学 五十嵐 善英 (Yoshihide IGARASHI)

1. はじめに

属性文法 (attribute grammar) は、文脈自由文法によって生成される文字列の意味付けの形式化として Knuth[6] によって導入された。循環性を持つ属性文法は、属性値が一意に定まらず有効でない。従って、属性文法はそれを使用する前に循環性の有無を検査しておく必要がある。この循環性検査の時間計算量は、本質的に指数関数時間であることが Jazayeri[5] によって証明された。しかし、多くの実例では指数関数時間は必要とせず、循環性検査は一般に Knuth のアルゴリズム [6] を改善した方法で十分実用的である。最近、Chebotar[1] や Deransart[3] によってこの検査の改善が行なわれた。本研究は、彼らの方法を更に改善したものである。我々の方法は、あらかじめ各生成規則に意味規則から得られる情報を抽出し持たせ、アルゴリズムの各段階でこの情報を用いその効率を高めるものである。

2. 概念と定義

定義 2.1 属性文法 G は、3つ組 $\langle Gu, A, F \rangle$ で表わす。 Gu, A, F はそれぞれ次の (1)~(3) で定義される。

- (1) 文脈自由文法 $Gu=(N, T, P, Z)$ は、 G の基底文脈自由文法 (underlying context-free grammar) と呼ばれる。ここで、 N は非終端記号の有限集合、 T は終端記号の有限集合、 P は生成規則の有限集合、 Z は開始非終端記号である。但し、 $N \cap T = \emptyset$ 、生成規則 $p \in P$ は、 $p: X_0 \rightarrow w_0 X_1 w_1 \cdots w_{n_p-1} X_{n_p} w_{n_p}$, ($X_i \in N, w_i \in T^*, i \in [0..n_p], n_p \geq 0$) の形とする。また、 Gu は既約 (reduced) とする。
- (2) Gu の各非終端記号 X に対し、互いに素な 2 つの有限集合 $S(X)$ と $I(X)$ を与える。 $S(X), I(X)$ は、それぞれ X の合成属性 (synthesized attribute), 相続属性 (inherited attribute) の集合である。 X の属性 a を $a(X)$ で、 a が取り得る値の集合を $V(a)$ で、 X の属性の集合を $A(X) (= S(X) \cup I(X))$ で、全ての属性の集合を $A (= \cup_{X \in N} A(X))$ で表わす。また、 $I(Z) = \emptyset$ 、任意の $X \in T$ に対し $A(X) = \emptyset$ とする。
- (3) 各生成規則 $p \in P$ に対する $F(p)$ は、 $S(X_0) \cup I(X_1) \cup \cdots \cup I(X_{n_p})$ の全ての属性を定義する意味規則 (semantic rules) の集合である。集合 $F (= \cup_{p \in P} F(p))$ は G の意味規則の集合である。属性 $a(X_k)$ を定義する意味規則は、 $a(X_k) := f_{a,k,p}(a_1(X_{i_1}), \dots, a_m(X_{i_m}))$, ($k=0$ のとき $a \in S(X_0)$, $1 \leq k \leq n_p$ のとき $a \in I(X_k)$), ($a_j \in A(X_{i_j}), i_j \in [0..n_p], j \in [1..m]$) という形をしている。ここで、 $f_{a,k,p}$ は、 $V(a_1(X_{i_1})) \times \cdots \times V(a_m(X_{i_m}))$ から $V(a(X_k))$ の中への写像である。このとき、 p に関係した属性 $a(X_k)$ は、 $a_1(X_{i_1}), \dots, a_m(X_{i_m})$ に依存するという。ここでの m は、 a, k, p によって一意に定まるものであり、生成規則 p の X_k に付け

られた属性 a の値を決定するために用いられる属性の数を表わす。また、意味規則は Bochmann 正規形であると仮定する。即ち、任意の $j \in [1..m]$ に対し、 $a_j(X_{i_j}) \in I(X_0) \cup (X_1) \cup \dots \cup (X_{n_p})$ である。 □

本論文では以降、各生成規則 p を終端記号を除いて表現する。

定義 2.2 生成規則 p の属性間の依存関係 $DR(p)$ は、次で与えられる。

$$DR(p) = \{ (a(X_i), b(X_j)) \mid b(X_j) := f_{b,j,p}(\dots a(X_i) \dots) \in F(p) \} \quad \square$$

定義 2.3

- (1) 生成規則 p に対する依存グラフ (dependency graph) $DG(p)$ は、 $DR(p)$ のグラフ表現である。即ち、 $DG(p)$ は p の属性間の依存関係を表わす有向グラフである。
- (2) 構文解析木 t に対する依存グラフ $DG(t)$ は、 t の構造に従って $DG(p)$ を繋ぎ合わせたグラフである。即ち、 $DG(t)$ は t 内の属性間の依存関係を表わす有向グラフである。
- (3) 生成規則 p に対する $FG(p)$ は、node が p の非終端記号 X_0, \dots, X_{n_p} で、arc が $DG(p)$ 内で $A(X_j)$ 内のある属性が $A(X_i)$ 内のある属性に (但し $i, j \in [0..n_p]$) 依存しているときに限り X_i から X_j へ向く有向グラフである。 □

定理 2.1[6] 属性文法 G が整定義 (非循環) である

⇔ G_u の各文字列に対する構文解析木 t の $DG(t)$ 内にサイクルがない。 □

もし $DG(t)$ がサイクルを含まなければ、 $DG(t)$ 内の属性はある順番に従って評価することができる ([2] 参照)。Knuth[6] はこの動的な性質を静的に検査する方法を示した。また、Jazayeri[5] はこの検査問題が本質的に指数関数時間 ($2^{cn/\log(n)} \leq T(n) \leq 2^{dn}$, n は文法を表現するのに必要なサイズ, c, d は係数) 必要であることを証明した。

3. 循環性を検査する一般的アルゴリズム

この問題の一般的なアルゴリズムとして Knuth のアルゴリズム [6] が知られているがこれはあまり実用的でない。少なくとも、次に示す Lorho による covering 法 (Knuth のアルゴリズムを改善した方法)[7] を用いるべきであることが知られている ([3] 参照)。

アルゴリズム $A1(\text{in } G)$;

- ```

begin
1. for each $X \in N$ do $D(X) := \{d_\phi\}$;
2. repeat
 $stability := true$;
3. for each $p \in P$ such that $p: X_0 \rightarrow X_1 X_2 \dots X_{n_p}$ do
4. if $n_p = 0$ then begin
5. let d_0 be the restriction of $DR(p)$ to $A(X_0)$;
6. if d_0 is not covered by $D(X_0)$ then begin
 $D(X_0) := covering(D(X_0) \cup \{d_0\})$;
 $stability := false$
 end
 end
 end
 else
7. for each $(d_1, \dots, d_{n_p}) \in D(X_1) \times \dots \times D(X_{n_p})$ do begin

```

```

8. d := DR(p) ∪ ∪j=1np j · dj;
9. compute d+;
10. if d+ is circular then begin
 determine G to be circular;
 stop
 end;
11. let d0 be the restriction of d+ to A(X0);
12. if d0 is not covered by D(X0) then begin
 D(X0) := covering(D(X0) ∪ {d0});
 stability := false
 end
13. end
14. until stability
end.

```

□

上のアルゴリズムで、D(X) は A(X) 上の依存関係を表わす二項関係の集合である。d<sub>0</sub> は A(X) 内の属性間に依存関係が存在しない (即ち、各属性は独立である) ことを表わす二項関係である。8 行目は、 $a d_j b \Leftrightarrow a(X_j) j \cdot d_j b(X_j)$  を意味する。9 行目の d<sup>+</sup> は A 上の関係 d の推移包含を表わす。

4. 前処理部

循環性検査の実行時間を短縮する 1 つの方法は、アルゴリズム A1 へ入力する文法に前処理を施すことである。ここでは、生成規則の分類、生成規則依存グラフの生成、生成規則の縮小、生成規則の分割および効率のよい処理の順番付けを前処理として行なう。

4.1 生成規則の分類

性質 4.1 DG(t) 内に現われる任意のサイクルに対し、次の 3 つの性質が成り立つ。

(1) DG(t) 内でサイクルの一部を構成している DG(p) のうち、1 番根に近い DG(p) は Fig. 1 のグラフを部分グラフとして持つ。

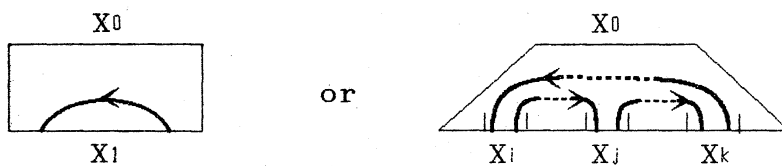


Fig. 1 DG(p)

(2) DG(t) 内でサイクルの一部を構成している DG(p) のうち、各枝の 1 番葉に近い DG(p) は Fig. 2 のグラフを部分グラフとして持つ。

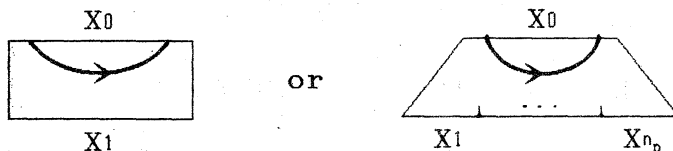


Fig. 2 DG(p)

(3) DG(t) 内でサイクルの一部を構成している DG(p) のうち、1), 2) 以外の DG(p) は Fig. 3 のグラフを部分グラフとして持つ。 □

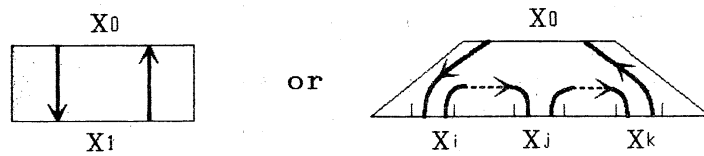


Fig.3 DG(p)

この性質により DG(t) 中に現われるサイクルは、上部、下部、中部に分割できる。

**定義 4.1** 生成規則 p の CAP, TR, CUP, NULL は次のように定義される。

CAP : FG(p) 中で node  $X_i (i \in [1..n_p])$  だけから構成されるサイクルが少なくとも 1 つ存在する。

TR : FG(p) 中で node  $X_0$  から出て 1 つ以上の node  $X_i (i \in [1..n_p])$  を通って再び node  $X_0$  へ戻ってくるサイクルが少なくとも 1 つ存在する。

CUP : FG(p) 中で node  $X_0$  上にループが存在する。

NULL : CAP, CUP, TR のいずれでもない。 □

(注意、生成規則は CAP, TR, CUP のいずれかを単独で持つ場合もあるし、それぞれ組み合わせ合わせて持つ場合もある。)

以上より、次の定理が得られる。

**定理 4.1** DG(t) 内の各サイクルについて、次のことが成り立つ。

DG(p) がサイクルの一部となりうる生成規則 p の中で、1 番根に近い生成規則は CAP 生成規則、各枝の 1 番葉に近い生成規則は CUP 生成規則、その間を連結する生成規則は TR 生成規則である。 □

証明) 性質 4.1 中の各 DG(p) に対する FG(p) (Fig.4) より明らかである。 □

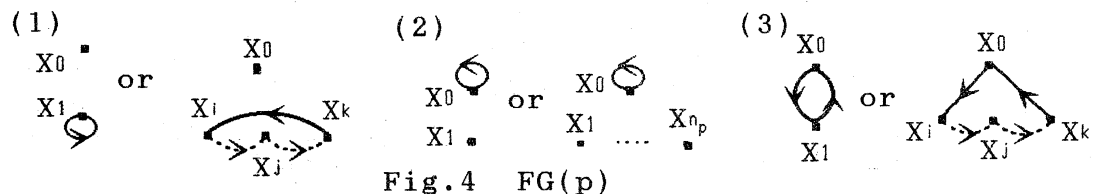


Fig.4 FG(p)

上の定理より、以下の系が導ける。

**系 4.1** NULL 生成規則を循環性検査から取り除いても検査結果は変わらない。 □

**系 4.2** CAP 生成規則を持たない属性文法は非循環である。 □

**系 4.3** CUP 生成規則を持たない属性文法は非循環である。 □

よって、系 4.2, 系 4.3 でいう属性文法のクラスは、各 FG(p) を調べるだけで非循環であるとわかる。このクラスは、File[4] により階層化された 1-visit の属性文法のクラスを真に含む。また、系 4.1 より NULL 生成規則を取り除いてもよいことがわかる。以上より、

前処理の第1段階として次のアルゴリズムを設計することができる。

```

/* 生成規則の分類 */
procedure F1(in P, DR; out CAP, TR, CUP);
begin
 CUP := ϕ ;
 TR := ϕ ;
 CAP := ϕ ;
 for each $p \in P$ do begin
 construct FG(p) from DR(p);
 if p is CUP-production then CUP := CUP \cup {p};
 if p is CAP-production then CAP := CAP \cup {p};
 if p is TR-production then TR := TR \cup {p}
 end;
 if (size(CUP) = 0) or (size(CAP) = 0) then begin
 determine G to be non-circular;
 stop
 end
end;

```

#### 4.2 生成規則依存グラフの生成

ここでは、生成規則の縮小および生成規則の分割を行なうのに用いる生成規則依存グラフについて述べる。

**定義 4.2**  $\alpha_{(\rho)}$  は生成規則の集合  $\rho$  内の  $p:A \rightarrow \mu$ ,  $p':B \rightarrow \nu$  について系列  $\mu$  の中に B を含むときかつそのときに限り、 $p \alpha_{(\rho)} p'$  であるような  $\rho$  上の関係である。 □

**定義 4.3**  $\alpha$  グラフを、各 node が  $\rho$  内の各生成規則であり、node  $p$ ,  $p'$  間に関係  $p \alpha_{(\rho)} p'$  が成り立つときかつそのときに限り、 $p$  から  $p'$  への arc が存在するような有向グラフとする。 □

次に上で述べた  $\alpha$  グラフの各 node に、定義 4.1 で述べた生成規則の分類情報 (CAP, TR, CUP) を持たせることを考える (注意、 $\rho$  を NULL 生成規則を除いた P の部分集合と考える)。

**定義 4.4**  $\alpha$  グラフの各 node に意味規則から得られる生成規則の分類情報 (CAP, TR, CUP) を持たせたグラフを  $\beta$  グラフという。 □

次に  $\beta$  グラフを変形することを考える。全ての node が TR である  $\beta$  グラフ内のサイクルを T サイクルという。  $\beta$  グラフ中で同じ T サイクルを構成する node (生成規則) を 1 つにまとめた node を S-node という。また、S-node 中に 1 つでも CAP(CUP) である生成規則があるとき、その S-node は CAP(CUP) であるという。

**定義 4.5**  $\beta$  グラフの中で T サイクルを構成する node を全て S-node に変換したグラフを  $\gamma$  グラフという (注意、T サイクルを構成する arc は消去される)。 □

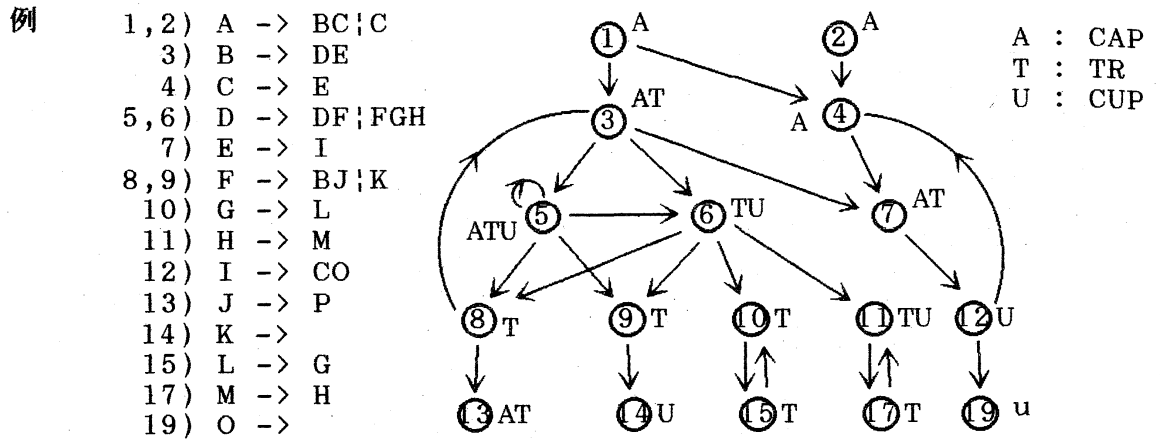


Fig.5 An example of  $\beta$ -graph

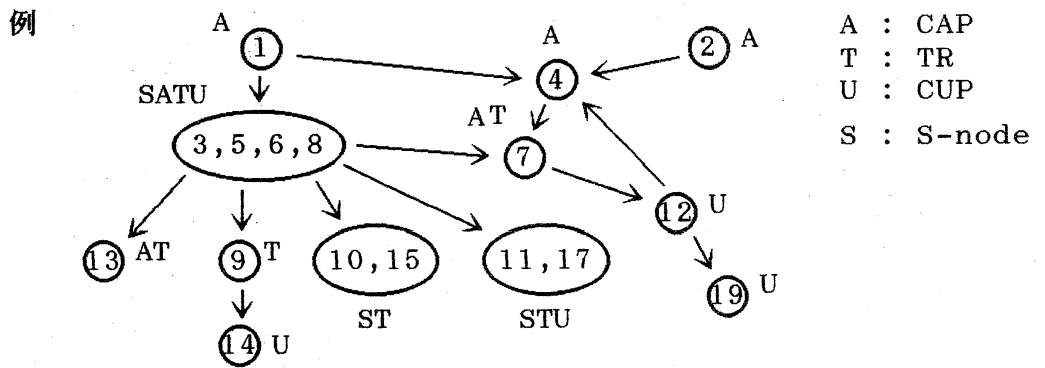


Fig.6 An example of  $\gamma$ -graph

$\gamma$ グラフを求めるアルゴリズムを以下に示す。

/\*  $\gamma$ グラフを作る \*/

```

procedure MAKEGRAPH(in CAP, TR, CUP; out γ -graph);
begin
 construct β -graph by using CAP, TR, CU;
 construct γ -graph from β -graph;
end;

```

### 4.3 生成規則の縮小

ここでは $\gamma$ グラフ内の useless node を消去して生成規則の縮小を行なう前処理部を設計する。 $\gamma$ グラフ内でCAP-node から出て、0個以上のTR-node を経由し、CUP-node に至るパスをAT\*U-パスという。 $\gamma$ グラフ内のAT\*U-パスを構成するnode と、CAP-CUP-TR-S-node(注意、このことはこのnode 内にAT\*U-パスが存在することを表わす)を useful node と呼び、useful node でないnode を useless node と呼ぶ。

**定理 4.2** useless node である生成規則 p の DG(p) は、どのような DG(t) 内でもサイクルの一部を構成しない。 □

(証明) 定理 4.1 より、ある節からその子孫のある葉に向かって、生成規則を CAP, 0 回以上

の TR, CUP の順に用いて、構文解析木  $t$  が作られるときだけ、 $DG(t)$  内にサイクルが存在する可能性がある。この導出順は、 $\gamma$  グラフ内の  $AT^*U$ -パスに対応する。このことより、このパスに関係しない node (useless node) である生成規則  $p$  の  $DG(p)$  は、どのような  $DG(t)$  内でもサイクルの一部を構成することはない。 □

系 4.4  $\gamma$  グラフから useless node を取り除いても循環性検査の結果は変わらない。 □

例

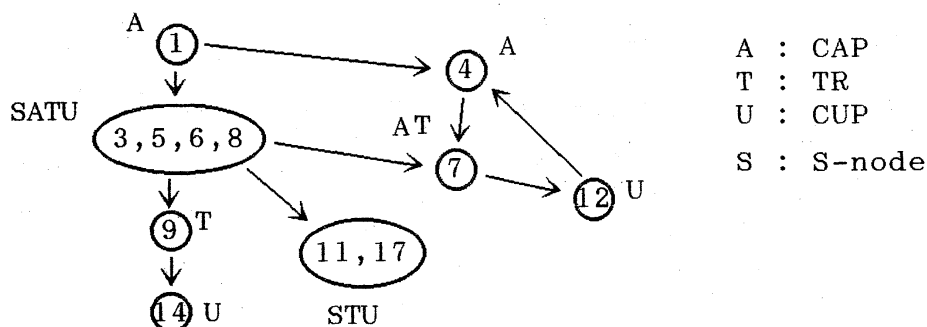


Fig. 7 An example of  $\gamma$ -graph removing useless production

以上より、前処理の第 2 段階として、 $\gamma$  グラフから useless node を除くアルゴリズムを設計することができる。

/\* 生成規則の縮小 \*/

**procedure** F2(in  $\gamma$ -graph; out  $\gamma$ -graph);

**begin**

find the  $AT^*U$ -paths and the S-nodes of CAP, CUP in  $\gamma$ -graph;

choose all the useless nodes by using the  $AT^*U$ -paths  
and S-nodes of CAP, CUP;

remove all the useless nodes from  $\gamma$ -graph;

**if**  $\gamma$ -graph has no nodes **then begin**

determine G to be non-circular;

stop

**end**

**end;**

□

#### 4.4 生成規則の分割および効率のよい処理の順番付け

循環性検査アルゴリズムの改善法として、Chebotar による入力文法の分割法 [1] と Deransart による week stability 法 [3] が知られている。 $\gamma$  グラフを導入することで、この 2 つの方法は組み合わせて用いることができる。また CUP, TR, CAP の情報を用いることによって更に改善することができる。

アルゴリズム A1 の実行中、 $D(X)$  にこれ以上新しい関係が追加されない状態を  $D(X)$  の安定状態といい、生成規則  $p: X \rightarrow \mu$  を処理するとき  $D(X)$  にこれ以上新しい関係が追加されなくなった状態をその生成規則の安定状態という。ここでは、安定順による生成規則のクラス分けと、そのクラスに処理の順番付けを与える前処理部を設計する。

アルゴリズム A1 中の変数  $D(X)$  は、まず CUP である生成規則を処理することで定まる  $A(X)$  上の依存関係にセットされる。続いて、TR である生成規則を処理することで定まる

A(X) 上の依存関係が追加される。また、生成規則が CAP であることは、その生成規則の処理においてサイクル検査を行なうことだけを意味する。よって、安定順による次の生成規則の分類が考えられる。

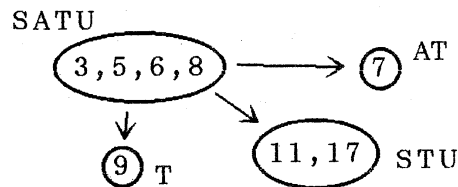
- 1) TR でなく CUP である生成規則群
- 2) TR である生成規則群
- 3) TR でなく CAP である生成規則群

(注意、CAP-CUP 生成規則は、1), 2) の両方に入る)

TR である生成規則群は更に安定順に細分できる。この目的のために、 $\gamma$  グラフ中で TR-node だけに注目したグラフを考える。

**定義 4.6** TR-node と両端が TR-node である arc だけからなる  $\gamma$  グラフの部分グラフを  $\delta$  グラフという。 □

例



A : CAP  
T : TR  
U : CUP  
S : S-node

Fig. 8 An example of  $\delta$ -graph

$\delta$  グラフから TR 生成規則のクラス分けと処理順番が決定できる。 $\delta$  グラフ中の任意の node について、その node に属する生成規則が安定するための必要条件是、その node から到達できる全ての node に属する生成規則が安定していることである。この条件を満たせば、各生成規則を node 単位に分割して、個々を検査アルゴリズムで検査してもよい。このことにより、TR 生成規則の分割と処理順番は、 $\delta$  グラフ内の node を topological sort することで得られる。このクラスの生成規則の安定化が指数関数時間かかる要因となる。

```

/* ソート・アルゴリズム */
procedure SORT(in δ -graph; out BLOCKS);
begin
 i := 0;
 while δ -graph has nodes do begin
 i := i + 1;
 choose a node π in δ -graph with no edge departing from it;
 let BLOCKS[i] be the set of productions associated with π ;
 remove π and all the arcs entering into π from δ -graph;
 end
end;

```

以上より、前処理の第 3 段階として次のアルゴリズムを設計することができる。

```

/* 生成規則の分割と処理順番 */
procedure F3(in γ -graph; out START, BLOCKS, END)
begin
 let START be the set of productions associated

```



```

with CUP but not TR nodes in γ -graph;
let END be the set of productions associated
with CAP but not TR nodes in γ -graph;
construct δ -graph from γ -graph;
SORT(δ -graph, BLOCKS)
end;

```

□

### 5. 主検査部

生成規則に CUP, TR, CAP の情報を持たせたことより、生成規則の各クラスに合った処理が可能になった。循環性検査アルゴリズムを以下に記す。

/\* TRでない CUP 生成規則群の処理 \*/

```

procedure PCUP(in START, DR; out D)
begin
 for each $p \in \text{START}$ such that $p: X_0 \rightarrow X_1 X_2 \dots X_{n_p}$ do begin
 let d_0 be the restriction of $\text{DR}(p)$ to $A(X_0)$;
 if d_0 is not covered by $D(X_0)$ then begin
 $D(X_0) := \text{covering}(D(X_0) \cup \{d_0\})$;
 end
 end;

```

□

/\* TRである生成規則群の処理 \*/

```

procedure PTR(in BLOCK, DR, D; out D);
begin
 repeat
 stability := true;
 for each $p \in \text{BLOCK}$ such that $p: X_0 \rightarrow X_1 X_2 \dots X_{n_p}$ do
 for each $(d_1, \dots, d_{n_p}) \in D(X_1) \times \dots \times D(X_{n_p})$ do
 $d := \text{DR}(p) \cup \bigcup_{j=1}^{n_p} j \cdot d_j$;
 compute d^+ ;
 if p is CAP-production then
 if d^+ is circular then begin
 determine G to be circular;
 stop
 end;
 let d_0 be the restriction of d^+ to $A(X_0)$;
 if d_0 is not covered by $D(X_0)$ then begin
 $D(X_0) := \text{covering}(D(X_0) \cup \{d_0\})$;
 stability := false
 end
 until (stability) or not(S_node(BLOCK))
 end;

```

□

/\* TRでない CAP 生成規則群の処理 \*/

```

procedure PCAP(in END, DR, D);
begin
 for each $p \in \text{END}$ such that $p: X_0 \rightarrow X_1 X_2 \dots X_{n_p}$ do
 for each $(d_1, \dots, d_{n_p}) \in D(X_1) \times \dots \times D(X_{n_p})$ do begin
 $d := \text{DR}(p) \cup \bigcup_{j=1}^{n_p} j \cdot d_j$;
 compute d^+ ;
 if d^+ is circular then begin
 determine G to be circular;
 stop
 end;
 end;

```

```

 end
 end
end;
/* 高速循環性検査アルゴリズム */
アルゴリズム A2(in G);
procedure FILTER(in P, DR; out START, BLOCKS, END);
begin
 F1(P, DR, CAP, TR, CUP);
 MAKEGRAPH(CUP, TR, CAP, γ -graph);
 F2(γ -graph, γ -graph);
 F3(γ -graph, START, BLOCKS, END)
end;
procedure TEST(in DR, START, BLOCKS, END);
begin
 for each $X \in N$ do $D(X) := \{d_\phi\}$;
 PCUP(START, DR, D);
 for $i:=1$ to size of BLOCKS do
 PTR(BLOCKS[i], DR, D, D)
 PCAP(END, DR, D)
end;
begin
 FILTER(P, DR, START, BLOCKS, END);
 TEST(DR, START, BLOCKS, END);
 determine G to be non-circular
end.

```

## 6. おわりに

ここで用いた前処理部分の各アルゴリズムは、高々、多項式時間で終わる。この前処理の採用は、一般に主検査部 (Knuth のアルゴリズム [6]) の実行時間を短縮し、記憶領域も短縮する。本改善アルゴリズムは、他の改善アルゴリズム (Deransart[3] 参照) を含みまた改良したものであり、Deransart 等の改善法より効率が良い。

## 参考文献

- [1] Chebotar, K.S.: Some modifications of Knuth's algorithm for verifying cyclicity of attribute grammars, Progr. Comput. Soft., Vol. 7, pp. 58-61, 1981.
- [2] Courcelle, B. and Franchi-Zannettacci, P.: Attribute grammars and recursive program schemes, Theor. Comput. Sci., Vol 17, pp. 163-191 and pp. 235-257, 1982.
- [3] Deransart, P. and J., M. and L., B.: Speeding up Circularity Tests for Attribute Grammars, Acta Informatica, Vol. 21, pp. 375-391, 1984.
- [4] File, G.: The Formal power of one-visit Attribute Grammar, Acta Informatica, pp. 275-302, 1981.
- [5] Jazayeri, M.: A simpler construction for showing the intrinsically exponential complexity of the circularity problem for attribute grammars, JACM, Vol. 28, pp. 715-720, 1981.
- [6] Knuth, D.E.: Semantics of context free languages, Mathematical Systems Theory, Vol. 2, pp. 127-145, 1968; Correction to article in Mathematical Systems Theory, Vol. 5, pp. 95-96, 1971.
- [7] Lorho, B. and Pair, C.: Algorithms for checking Consistency of attribute grammars. in: Proving and Improving Programs, G. Huent, G. Kahn(eds.). Colloque IRIA, Arc-et-Senan France, pp. 29-54, 1975.