属性文法の高速循環性検査アルゴリズム†

小池 博†† 五十嵐善英†† 佐渡一広††

属性文法の循環性検査の時間計算量は，本質的に文法の大きさの指数時間であることが知られている．この検査アルゴリズムは，最初 Knuth によって与えられたが，その後，Chebotar, Räihä, Deransart らによって効率の改善が行われた．本論文では，属性文法の構文解析木に対する依存グラフの性質を調べ，計算時間をさらに短縮した循環性検査アルゴリズムについて述べる．我々のアルゴリズムでは，生成規則に意味規則から得られる情報を付加し，循環性検査に不必要な情報を取り除く前処理を行っている．このことにより主検査の実行時間はかなり短縮される．幾つかの属性文法の記述例について，我々のアルゴリズムとすでに知られている循環性検査アルゴリズムの効率の比較を計算機によって行った．

1. まえがき

属性文法 (attribute grammar) は，文脈自由文法で構文が定義されている言語の，意味の仕様記述のために Knuth[8] によって導入された．属性文法は意味規則の集合から，構文解析木の各節に与えられる意味が一意に定まるかどうかをあらかじめ知ることが要求される．このことは属性文法の循環性問題として知られており，その時間計算量は本質的に文法の大きさの指数関数であることが Jazayeri[4),5)] によって証明されている．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 のアルゴリズムを改善した方法で十分実用に耐えうると考えられている．その後，Chebotar[1)], Räihä[11)], Deransart[2)] らによって循環性検査の効率の改善が行われた．本論文では，属性文法の構文解析木に対する依存グラフの性質を調べ，さらに効率のよい循環性検査アルゴリズムを設計する．我々のアルゴリズムでは，生成規則に意味規則から得られる情報を付加し，循環性検査に不必要な情報を取り除く前処理を行っている．このことにより，主検査の実行時間はかなり短縮される．

以下の2章では，属性文法に関する定義と記法を述べる．3章では，循環性検査の改善方法を述べる．4章では，本論文で与えた改善アルゴリズムと既に知られている他の循環性検査のアルゴリズムを，四つの属性文法の記述例に対して適用して比較する．5章はむすびである．

2. 定義と記法

本論文で用いる記法は，主に文献 2), 3), 12) による．

属性文法 G は，三つ組 $\langle Gu, A, F\rangle$ で表される．Gu, A, F はそれぞれ次の(1)～(3)で与えられる．

(1) $Gu=(N, T, P, Z)$ は既約な文脈自由文法で，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で，$N\cap T=\phi$, Z は N の要素で開始記号，P は生成規則の有限集合である．終端記号は，属性値評価には何ら貢献をしないので，本論文では各生成規則は終端記号を除いて表す．各 $p\in P$ について，$p: X_0\rightarrow X_1X_2\cdots X_{n_p}$, ただし，各 X_i $(0\leq i\leq n_p)$ は N の要素である．

(2) Gu の各非終端記号 X に対し，互いに素な二つの有限集合 $S(X)$ と $I(X)$ が与えられる．$S(X)$ と $I(X)$ は，それぞれ X の合成属性 (synthesized attribute) と相続属性 (inherited attribute) の集合である．生成規則 $p: X_0\rightarrow X_1X_2\cdots X_{n_p}$ について，その i 番目 $(0\leq i\leq n_p)$ の非終端記号 X_i の属性 a の生起 (occurrence) を (a,i) で表す．X の属性の集合を $A(X)(=S(X)\cup I(X))$ で，G のすべての属性の集合を $A(=\cup_{X\in N}A(X))$ で表す．開始記号 Z に対し，$I(Z)=\phi$, 任意の $X\in T$ に対し，$A(X)=\phi$ とする．

(3) 各生成規則 $p: X_0\rightarrow X_1X_2\cdots X_{n_p}$ に対する $F(p)$ を，$S(X_0)\cup I(X_1)\cup\cdots\cup I(X_{n_p})$ の各属性について，それぞれの属性生起の属性値を定める意味規則 (semantic rule) のすべての集合とする．集合 $F(=\cup_{p\in P}F(p))$ は G の意味規則の集合である．属性生起 (a,k) を定める意味規則は，$(a,k)=$

† Some Fast Algorithms for Testing Circularity of Attribute Grammars by HIROSHI KOIKE, YOSHIHIDE IGARASHI and KAZUHIRO SADO (Department of Computer Science, Faculty of Engineering, Gunma University).

†† 群馬大学工学部情報工学科

$f_{a,k,p}((a_1,i_1),\cdots,(a_m,i_m))$, $(a_j \in A(X_{i_j})$, $i_j \in [0..n_p]$, $j \in [1..m])$ なる形をしている．ただし，$k=0$ のとき $a \in S(X_0)$, $1 \le k \le n_p$ のとき $a \in I(X_k)$ である．このとき，p に関係した属性生起 (a,k) は，$(a_1,i_1),\cdots,(a_m,i_m)$ に依存するという．

上の定義で，$(a,k)=f_{a,k,p}((a_1,i_1),\cdots,(a_m,i_m))$ が，各 $j \in [1..m]$ について，$(a_j,i_j) \in I(X_0) \cup S(X_1) \cup \cdots \cup S(X_{n_p})$ を満たすとき，この意味規則は Bochmann 正規形であるという．

以下，本論文では Bochmann 正規形の意味規則だけを持つ属性文法のみを考える．

生成規則 p の属性生起間の依存関係 $DR(p)$ を，$DR(p)=\{((a,i),(b,j)) \mid (a,i)=f_{a,i,p}(\cdots,(b,j),\cdots) \in F(p)\}$ とする．生成規則 p に対する依存グラフ (dependency graph) $DG(p)$ は，$DR(p)$ のグラフ表現であり，$((a,i),(b,j)) \in DR(p)$ のとき，かつそのときに限り節 (b,j) から節 (a,i) に向かう枝がある有向グラフである．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．すなわち，$DG(t)$ は t 内の属性生起間の依存関係を表す有向グラフである．生成規則 $p: X_0 \to X_1 \cdots X_{n_p}$ について，$FG(p)$ は，節の集合が $\{X_0,\cdots,X_{n_p}\}$ で，$DG(p)$ 内の $A(X_j)$ にある属性生起が $A(X_i)$ にある属性生起に依存しているとき (ただし $i,j \in [0..n_p]$)，かつそのときに限り X_i から X_j へ向く枝を持つ有向グラフである．

属性文法 G の各構文解析木 t について，$DG(t)$ 内にサイクルがないとき G は非循環であるという．

3. 循環性検査アルゴリズム

3.1 Knuth のアルゴリズム

属性文法の循環性検査アルゴリズムとして，Knuth の方法[8]が一般に知られている．Chebotar[1], Räihä[11], Deransart[2] らの高速循環性検査アルゴリズムも，本論文で導くアルゴリズムも，Knuth のアルゴリズムを改善したものである．Knuth のアルゴリズムを図 1 に示す．以下，このアルゴリズムを KCT と呼ぶ．

KCT 中で，$D(X)$ は $A(X)$ 上の依存関係を表す二項関係の集合である．d_ϕ は $A(X)$ 内の属性間に依存関係が存在しないことを表す二項関係である．よって，$d_\phi=\phi$ と見なしてよい．8 行目は，$ad_jb \Leftrightarrow (a,j)j \cdot d_j(b,j)$ を意味する．9 行目の d^+ は A 上の関係 d の推移閉包を表す．

関係の集合 $D(X)$ から，その中の他の関係に含まれるような関係を取り除いても，循環性検査の結果は変わらない．この改善法は covering 法と呼ばれ，Lorho[10] によって導入された．このように改善したアルゴリズムは，KCT の 6 行目と 12 行目の **if** 文をそれぞれ次の **if** 文に置き換えることで得られる．

```
if d0 is not covered by D(X0) then begin
    D(X0) := covering (D(X0) ∪ {d0});
    escape := false
    end
```

このようにして得られたアルゴリズムを KCTC で表す．以下，KCTC を循環性検査の主検査部とする．

本論文では，生成規則の分類，生成規則の縮小，生

```
    algorithm KCT(in G: grammar);
      begin
1.      for each X ∈ N do D(X) := {dφ};
2.      repeat
          escape := true;
3.        for each p ∈ P (p:X0→X1X2···Xnp) do
4.          if np = 0 then begin /* the right hand side
                                    of p is a terminal string */
5.            let d0 be the restriction of DR(p) to A(X0);
6.            if d0 ∉ D(X0) then begin
                  D(X0) := D(X0) ∪ {d0};
                  escape := false
                  end
              end
            else
7.            for each (d1, ···, dnp) ∈ D(X1)×···×D(Xnp) do begin
8.              d := DR(p) ∪ ∪_{j=1}^{np} j·dj;  /* adjb ↔ (a, j)j·dj(b, j) */
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 ∉ D(X0) then begin
                      D(X0) := D(X0) ∪ {d0};
                      escape := false
                      end
13.           end
14.     until escape
    end.
```

図 1 Knuth の循環性を検査するアルゴリズム
Fig. 1 Knuth's algorithm for testing circularity.

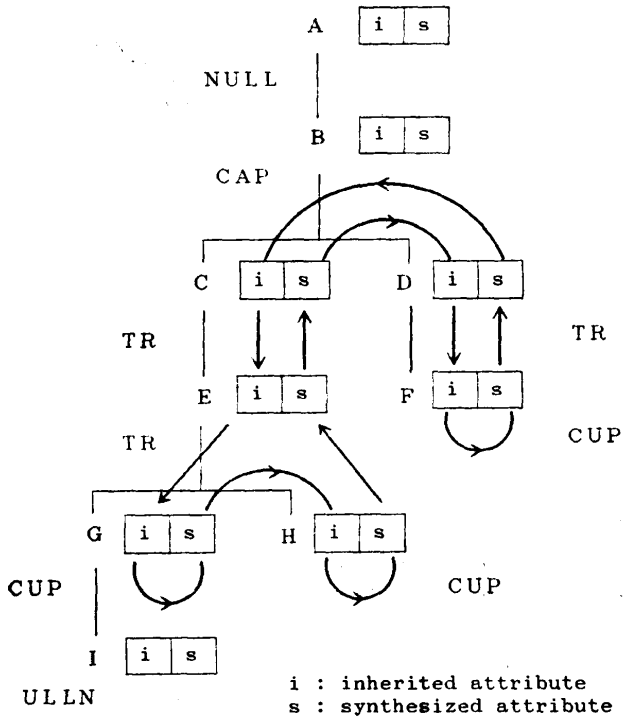

図 2 *DG(t)* 内のサイクルの例
Fig. 2 An example of a cycle in *DG(t)*.

成規則の分割および処理の順番付けの各前処理部を導入することによって，循環性検査の効率を上げる．

3.2 生成規則の分類

属性文法 $G=(Gu, A, F)$ の構文解析木 t について，$DG(t)$ 中に現れるサイクルは図 2 に示すように，上部，中部，下部を構成する $DG(p)$ に分割できる．これらの部分に相当する生成規則 $p: X_0 \rightarrow X_1 \cdots X_{n_p}$ を，それぞれ CAP, TR, CUP と呼ぶ．すなわち，CAP, TR, CUP を次のように定める．

(1) $FG(p)$ 中に節 X_0 を含まないサイクルが存在するとき，その生成規則 p は CAP であるという．

(2) $FG(p)$ 中に節 X_0 を含む長さ 2 以上のサイクルが存在するとき，その生成規則 p は TR であるという．

(3) $FG(p)$ 中に節 X_0 上の単位ループが存在するとき，生成規則 p は CUP であるという．

(4) CAP でも TR でも CUP でもないような生成規則 p は，NULL であるという．

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，それぞれを組み合わせて持つ場合もある．CAP, TR, CUP, NULL を生成規則の分類情報という．

分類情報の定義より，次の定理は明らかである．

[定理 1] $DG(t)$ 内の各サイクルについて，$DG(p)$ がそのサイクルの一部となりうる生成規則 p の中で，根に最も近い生成規則は CAP 生成規則，各枝の葉に最も近い生成規則は 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．

以下の三つの系は，上の定理より明らかである．

[系 1] NULL 生成規則を入力文法から取り除いても，循環性検査の結果は変わらない．

[系 2] CAP 生成規則を持たない属性文法は非循環である．

CAP 生成規則を持たない属性文法は，Filè[3] によって階層化された 1-VISIT の属性文法である．

[系 3] CUP 生成規則を持たない属性文法は非循環である．

系 1 より，循環性検査の効率を上げるために，NULL 生成規則を取り除いてもよいことがわかる．属性文法の各生成規則について，$FG(p)$ を調べることによって，p が CAP であるか，TR であるか，あるいは CUP であるかを判定することができる．高速化のためのこの前処理手続き FILT1 を図 3 に示す．

```
procedure FILT1(in P, DR; out CAP, TR, CUP);
  begin
    CUP := φ;
    TR  := φ;
    CAP := φ;
    for each p ∈ P do begin
      construct FG(p) from DR(p);
       /* the following three decisions
          can be done in the cubic order
          time of the grammar size */
      if p is CUP-production
              then CUP := CUP ∪ {p};
      if p is CAP-production
              then CAP := CAP ∪ {p};
      if p is TR-production
              then TR := TR ∪ {p}
      end;
    if (CUP = φ) or (CAP = φ)  then  begin
       determine G to be non-circular;
       stop
       end
end;
```

図 3 生成規則を分類する手続き
Fig. 3 A procedure for partitioning the set of productions.

3.3 生成規則の導出パターングラフ

ここでは，生成規則の縮小，および生成規則の分割と処理順番を行うのに用いる導出パターングラフ (derivation pattern graph) について述べる．

生成規則の集合 $Q \subseteq P$ について，Q 上の二項関係 $\alpha(Q)$ を次のように定める．Q 内の生成規則の組，$p: A \to \mu$, $p': B \to \nu$ について，B が系列 μ 内に生起しているとき，かつそのときに限り $(p', p) \in \alpha(Q)$ である．生成規則の集合 $Q \subseteq P$ について，節の集合が Q で，$(p', p) \in \alpha(Q)$ のとき，かつそのときに限り p から p' への枝を持つ有向グラフを $\alpha(Q)$ 導出パターングラフという．Q を明記しなくても誤解がないときは，このグラフを α 導出パターングラフ (α-DPG と略記) と書く．NULL 分類情報を持つ生成規則を除いた生成規則の集合の α-DPG の各節に，その生成規則の分類情報を付けたグラフを β 導出パターングラフ (β-DPG と略記) という (**図 4** 参照)．

β-DPG は循環性検査で用いるが，その効率を上げるために，まず，β-DPG の変形を考える．すべての節が TR である β-DPG 内のサイクルをこのグラフの TR サイクルという．β-DPG 内の各 TR サイクル上の節を一つの節にする変換を TR 変換という．TR 変換で一つの節になったものを COMP 節という (もとの節と同じ節は，COMP 節といわない)．各 COMP 節について，そのCOMP 節に圧縮された節の中に一つでも CAP (TR, CUP) である生成規則があったとき，その COMP 節は CAP (TR, CUP) であるという．TR 変換されたグラフのラベルを，次のように定める．圧縮されない節についてはもとの生成規則をその節のラベルとし，COMP 節については圧縮された節に相当する生成規則の集合をその COMP 節のラベルとする．この操作によって作られるグラフを，γ 導出パターングラフ (γ-DPG と略記) という (**図 5** 参照)．γ-DPG を求める前処理手続き GAMMA を**図 6** に示す．

```
1,2) A -> BC¦C
  3) B -> DE
  4) C -> E
5,6) D -> DF¦FGH
  7) E -> I
8,9) F -> BJ¦K
 10) G -> L
 11) H -> M
 12) I -> CO
 13) J -> P
 14) K ->
 15) L -> G
 17) M -> H
 19) O ->
```

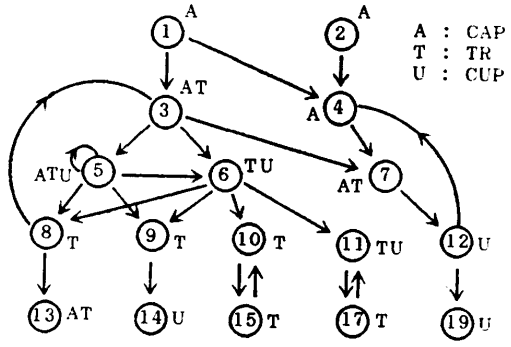

図 4 β 導出パターングラフ
Fig. 4 A β-derivation pattern graph.

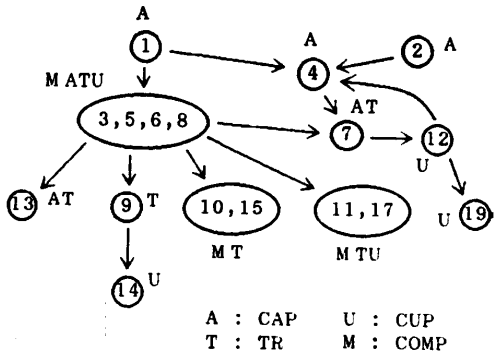

図 5 γ 導出パターングラフ
Fig. 5 A γ-derivation pattern graph.

```
procedure GAMMA
    (in CAP, TR, CUP; out γ-DPG);
 begin
  construct β-DPG;
  construct γ-DPG from β-DPG;
  /* the γ-DPG is obtained from
     TR-transformation of β-DPG */
 end;
/* the computing time of this procedure
   is at most the square order of the
   number of production rules */
```

図 6 γ 導出パターングラフを構成する手続き
Fig. 6 A procedure for constructing a γ-DPG.

3.4 生成規則の縮小

この節では，循環性判定に関係しない生成規則を γ-DPG から取り除く手続きを設計する．このような生成規則の集合の縮小によって，循環性検査の効率を上げることができる．

γ-DPG 内で CAP である節から出て，0 個以上の TR である節を経由して，CUP である節に至るパスを AT*U-パスと呼ぶ．γ-DPG 内の AT*U-パスを構成する節と，CAP-TR-CUP である COMP 節を有用な節と呼び，そうでない節を無用な節と呼ぶ．CAR-TR-CUP である COMP 節は，圧縮される前のグラフ (β-DPG) 内に AT*U-パスが存在することを表している．

[定理 2] 任意の構文解析木 t について，無用な節

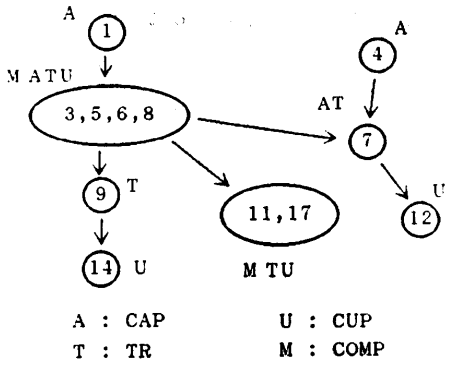

図 7 無用な節を取り除いた γ 導出パターングラフ
Fig. 7 A γ-derivation pattern graph without useless nodes.

```
procedure FILT2(in γ-DPG; out γ-DPG);
 begin
  find the AT*U-paths and
  the CAP-TR-CUP COMP-nodes in γ-DPG;
  choose all the useless nodes by using
  the AT*U-paths and CAP-TR-CUP COMP-nodes;
  remove all the useless nodes from γ-DPG;
  if γ-DPG has no nodes then begin
        determine G to be non-circular;
        stop
        end
 end;
```

図 8 無用な節を取り除く手続き
Fig. 8 A procedure for removing useless-nodes.

に属する生成規則 p の $DG(p)$ は，$DG(t)$ のサイクルの一部に成り得ない．

(証明) 定理1より，構文解析木 t の依存グラフ $DG(t)$ 内にサイクルがあれば，t の根から葉に向かうあるパス上に (パスの一部でもよい)，

CAP である生成規則，

0回以上の TR である生成規則，

CUP である生成規則

の順に，生成規則がその構文解析木 t に適用されている．生成規則の導出のこの順序は，γ-DPG 内の AT*U-パスに相当する (COMP 節については，圧縮される前の β-DPG 上で考える)．よって，このパスに関与しない節 (すなわち，無用な節) は，$DG(t)$ 内でサイクルの一部を構成することはない． □

上記の定理より次の系が成立する．

[系4] 無用な節である生成規則を入力文法より取り除いても，循環性検査の結果は変わらない．

無用な節を取り除いた γ-DPG の例を図7に示す．γ-DPG から無用な節を取り除く前処理手続き FILT2 を図8に示す．

3.5 生成規則の分割と処理順番

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 weak stability 法[2]が知られている．前節で導入した γ-DPG を用いて，分割法[1] と weak stability 法[2] を組み合わせることができる．また CUP, TR, CAP の分類情報を用いることによってさらに改善することができる．

アルゴリズム KCTC 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 $D(X)$ の安定状態と呼び，生成規則 $p: X \to \mu$ 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 p の安定状態と呼ぶ．ここでは，安定順による生成規則のクラス分けと，クラスの処理順番を与える前処理手続きを設計する．

アルゴリズム KCTC では，分類情報は用いてはいないが，我々が導入した分類情報を考慮して，そのアルゴリズムの動作を検討すると，次のことがいえる．アルゴリズムの実行中，まず CAP 生成規則の処理によって得られる $A(X)$ 上の依存関係が $D(X)$ 内に求められ，ついで，この依存関係と TR 生成規則の処理から得られる $A(X)$ 上の依存関係が $D(X)$ 内に追加される．生成規則の CAP 分類情報は，構文解析木の下から順次定まる $A(X)$ 上の依存関係を用いて，サイクルがあるかどうかを検査するときのみ用いられる．以上のことより，生成規則の安定順による分割と処理が効率のよい順番付を，次の各クラス順に定めることができる．

1) TR でなく CUP である生成規則のクラス．
2) TR である生成規則のクラス．
3) TR でなく CAP である生成規則のクラス．

このクラス分けは重複を許している．すなわち，CAP-CUP 生成規則は，1) と 2) の両方のクラスに入る．次に 2) のクラスをさらに細分するが，この目的のために，γ-DPG 中の TR である節だけに注目したグラフを考える．

すべての TR である節と，両端が TR の節になっているすべての枝からなる γ-DPG の部分グラフを，δ 導出パターングラフ (δ-DPG と略記) という (図9

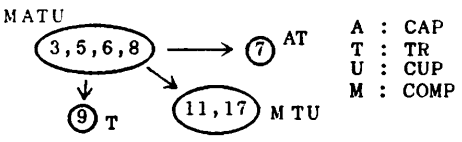

図 9 δ 導出パターングラフ
Fig. 9 A δ-derivation pattern graph.

```
procedure FILT3
    (in γ-DPG; out INIT, BLOCKS, FINAL);
begin
 let INIT be the set of productions
 associated with CUP but not TR nodes
 in γ-DPG;
 let FINAL be the set of productions
 associated with CAP but not TR nodes
 in γ-DPG;
 construct δ-DPG from γ-DPG;
 the nodes of δ-DPG are topologically
 sorted, and the sorted nodes are stored
 into array BLOCKS in nondecreasing order;
end;
```

図 10 生成規則の分割および順番付を行う手続き
Fig. 10 A procedure for partitioning and ordering productions.

参照). δ-DPG を用いた，TR 生成規則のクラス分けと処理順番の決定を，次のように行う．分割は，TR である生成規則を δ-DPG の節ごとに行う．δ-DPG 中の任意の節について，その節に属する生成規則が安定するための必要条件は，その節から到達できるすべての節に属する生成規則が安定していることである．このことにより，TR 生成規則の処理は δ-DPG の下の節から行えばよい．よって，δ-DPG の節をトポロジカルソートし，その順序を TR 生成規則のクラスの処理順番とする．COMP 節内の生成規則の安定化が指数関数時間かかる要因となる．

以上より，効率改善の第 3 段階の前処理手続き FILT 3 を設計することができる．この手続きを**図 10** に示す．

3.6 生成規則の各クラスに対する処理

前処理 FILT1 によって，生成規則の CAP, TR, CUP の情報を持たせてあるので，主検査で生成規則の各クラスに適した処理が可能である．すなわち，生成規則の各クラスについて，主検査部の一部を省略できる．

(1) TR でない CUP 生成規則のクラスの処理

このクラスの生成規則に対しては，アルゴリズムの主検査部で $D(X)$ 内に $A(X)$ 上の関係を代入するだけで十分である．よって，このクラスの生成規則の主検査部は，**図 11** の手続き PCUP で行える．

(2) TR である生成規則のクラスの処理

このクラスの生成規則に対しては，アルゴリズム KCTC で行うすべての処理が必要である．KCTC の 3-13 行目を一度実行するだけで，COMP でないクラスの生成規則はすべて安定する．このことにより，2-14 行目のループは，COMP であるクラスに対してのみ行えばよい．また，10 行目の d^+ のサイクル検

```
procedure PCUP(in INIT, DR; out D)
 begin
  for each p ∈ INIT
          (p:X0→X1X2···Xnp) do begin
   let d0 be the restriction of DR(p) to A(X0);
   if d0 is not covered by D(X0) then
     D(X0) := covering(D(X0) ∪ {d0});
  end
 end;

procedure PTR(in BLOCKS, DR, D; out D)
 procedure PBLOCK(in BLOCK, DR, D; out D);
 begin
  repeat
   escape := true;
   for each p ∈ BLOCK (p:X0→X1X2···Xnp) do
     for each (d1, ···, dnp)
              ∈ D(X1)×···×D(Xnp) do begin
       d := DR(p) ∪ U_{j=1}^{np} j·dj;
       compute d+;
       if p is CAP-production then
       if d+ is circular then begin
           determine G to be circular;
           stop
           end;
       let d0 be the restriction of d+ to A(X0);
       if d0 is not covered by D(X0) then begin
            D(X0) := covering(D(X0) ∪ {d0});
            escape := false
            end
       end
  until (escape) or not(COMP_node(BLOCK))
 end;
begin /* procedure PBLOCK main */
 for i:=1 to size of BLOCKS do
     PBLOCK(BLOCKS[i], DR, D, D)
end;

procedure PCAP(in FINAL, DR, D);
 begin
  for each p ∈ FINAL (p:X0→X1X2···Xnp) do
   for each (d1, ···, dnp)
              ∈ D(X1)×···×D(Xnp) do begin
     d := DR(p) ∪ U_{j=1}^{np} j·dj;
     compute d+;
     if d+ is circular then begin
         determine G to be circular;
         stop
         end
     end
 end;
```

図 11 手続き PCUP, PTR, PCAP
Fig. 11 Procedures PCUP, PTR and PCAP.

査は，CAP 生成規則だけ行えばよい．これらの改善を施した手続き PTR を**図 11** に示す．

(3) TR でない CAP 生成規則のクラスの処理

上の(1)と(2)の処理を行うと，すべての $D(X)$ は安定する．このクラスの生成規則の処理は，この安定した $D(X)$ を用いて d^+ 内のサイクルを検査するだけでよい．よって，図 11 の手続き PCAP で行える．

今まで述べた効率改善をすべて取り入れた高速循環性検査アルゴリズム F 123 KCT を**図 12** に示す．

```
algorithm F123KCT(in G: grammar);

procedure FILTER
    (in P, DR; out INIT, BLOCKS, FINAL);
  begin
    FILT1(P, DR, CAP, TR, CUP);
    GAMMA(CUP, TR, CAP, γ-DPG);
    FILT2(γ-DPG, γ-DPG);
    FILT3(γ-DPG, INIT, BLOCKS, FINAL)
  end;

procedure TEST
          (in DR, INIT, BLOCKS, FINAL);
  begin
    for each X∈N do  D(X) := {d_φ};
    PCUP(INIT, DR, D);
    PTR(BLOCKS, DR, D, D);
    PCAP(FINAL, DR, D)
  end;

begin /* main */
  FILTER(P, DR, INIT, BLOCKS, FINAL);
  TEST(DR, INIT, BLOCKS, FINAL);
  determine G to be non-circular
end.
```

図 12 高速循環性検査アルゴリズム
Fig. 12 A fast circularity test algorithm.

4. 実行効率の比較

本章では，我々のアルゴリズムとこれまでに知られているアルゴリズムを，四つの属性文法の記述例に対して適用し，それらの実行時間から効率を比較する．

測定は以下の九つの循環性検査アルゴリズムに対して行った．

- KCT：Knuth のアルゴリズム[8]
- KCTW：KCT＋Weak stability 法[2]
- KCTC：KCT＋Covering 法[10]
- KCTCW：KCT＋Weak stability 法＋Covering 法
- PKCTCW：分割法[1]→KCTCW
- F 1 KCT：FILT1→KCTCW
- F 12 KCT：FILT 1→FILT 2→KCTCW
- F 13 KCT：FILT 1→FILT 3→TEST
- F 123 KCT：FILT 1→FILT 2→FILT 3→TEST

検査した属性文法は以下の四つである．

- D 1：Kastens[6] の例 (ordered AG) を Bochmann 正規形に変換した属性文法．
- D 2：Knuth[8] の 2 進小数の値を計算する属性文法．
- D 3：Filè[3] の変数の宣言と参照からなる言語の属性文法 (1-VISIT AG)．
- D 4：簡単な自然言語処理を行う G 20 属性文法[13] (1-VISIT AG)．

各アルゴリズムは，同一の基本データ構造を用いて，PASCAL 言語で記述した．使用計算機システムは FACOM M-340 U/UTS で，実行時間は CPU 秒で測定した．実行時間の比較を表 1 に示す．

表 1 循環性検査の実行効率の比較
Table 1 A comparison of several circularity test algorithms.

	D1	D2	D3	D4
KCT	107.0	13.7	7.1	47.2
KCTW	102.6	13.7	7.1	39.7
KCTC	35.1	5.0	6.0	30.5
KCTCW	30.4	4.9	6.0	25.9
PKCTCW	34.3	2.7	3.8	12.9
F1KCT	29.1	3.9	0.6	0.9
F12KCT	29.1	4.0	0.6	1.0
F13KCT	20.9	2.4	0.6	1.0
F123KCT	20.3	2.4	0.6	1.0

5. むすび

本論文に用いた前処理部のアルゴリズムの実行時間は，たかだか多項式時間である．この前処理の採用は，一般に主検査 (KCT または KCTC) の実行時間を短縮する．我々の改善アルゴリズムは，Chebotar, Räihä, Deransart らの改善アルゴリズムを包含し，さらに改善したものである．表 1 より，我々の設計した F 123 KCT は，いままでに知られている循環性検査アルゴリズムより速いことがわかる．

前処理手続き FILT 1 は，1-VISIT の属性文法を非循環と決定する．このことは，属性文法 D 3, D 4 (1-VISIT のクラス) に対する結果に現れている．また FILT 1 は，NULL 生成規則を取り除き入力文法を縮小する．この効果は，F 1 KCT に対する結果に現れる．

前処理手続き FILT 2 を用いたアルゴリズムは，この手続きを用いて入力文法を縮小できる属性文法に対して効果がある．今回の比較に用いた属性文法に対しては有効な結果を得てないが，より大きな属性文法に対しては効果があることが期待できる．

前処理手続き FILT 3 は，生成規則の集合を幾つかのクラスに分割できる場合には，大きな効果を発揮する．我々の実験結果でも FILT 3 による効果がよく現れている．

文法の大きさの多項式時間内で，強循環性[6),7)] (絶

対循環性とも呼ぶ) を検査できるアルゴリズムが存在する．属性文法のあるクラスに属するものは，強循環性検査アルゴリズムでその循環性を判定できることが知られている．本論文で用いた効率改善法は，強循環性検査アルゴリズムにも適用できる[9]．

謝辞 実験に必要な属性文法を提供していただいた，東京工業大学片山卓也教授，および筑波大学佐々政孝講師に感謝する．

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) Deransart, P., Jourdan, M. and Lorho, B.: Speeding up Circularity Tests for Attribute Grammars, *Acta Inf.*, Vol. 21, pp. 375-391 (1984).
3) Filè, G.: Theory of Attribute Grammars, Ph. D. Dissertation, Twente University of Technology, Enschede, The Netherlands (1983).
4) Jazayeri, M., Ogden, W. and Rounds, W.: The Intrinsically Exponential Complexity of the Circularity Problem for Attribute Grammars, *Comm. ACM*, Vol. 18, No. 12, pp. 697-706 (1975).
5) Jazayeri, M.: A Simpler Construction for Showing the Intrinsically Exponential Complexity of the Cicularity Problem for Attribute Grammars, *J. ACM*, Vol. 28, No. 4, pp. 715-720 (1981).
6) Kastens, U.: Ordered Attributed Grammars, *Acta Inf.*, Vol. 13, pp. 229-256 (1980).
7) Kennedy, K. and Warren, S. K.: Automatic Generation of Efficient Evaluators for Attribute Grammars, *Proc. 3rd ACM Symp. on Principles of Programming Languages*, pp. 32-49 (1976).
8) Knuth, D. E.: Semantics of Context-free Languages, *Mathematical Systems Theory*, Vol. 2, No. 2, pp. 127-145 (1968); *Correction to Article in Mathematical Systems Theory*, Vol. 5, No. 1, pp. 95-96 (1971).
9) 小池 博，五十嵐善英：属性文法の高速循環性検査アルゴリズムと実験結果，電子通信学会技報，Vol. 86, No. 60, COMP86-15, pp. 45-52 (1986).
10) Lorho, B. and Pair, C.: Algorithms for Checking Consistency of Attribute Grammars, in Huent, G. and Kahn, G. (eds.), *Proving and Improving Programs*, Colloque IRIA, Arc-et-Senans, France, pp. 29-54 (1975).
11) Räihä, K. J. and Saarinen, M.: Testing Attribute Grammars for Circularity, *Acta Inf.*, Vol. 17, pp. 185-192 (1982).
12) 佐々政孝：属性文法—チュートリアル—，情報処理学会研究報告，SF86-16, pp. 1-10 (1986).
13) 田村直良：上昇解析型属性文法評価器とその自然言語処理への応用，東京工業大学博士論文 (1985).

(昭和61年8月19日受付)
(昭和61年11月5日採録)

小池 博 (学生会員)
昭和37年生．昭和60年群馬大学工学部情報工学科卒業．現在，同大学院工学研究科情報工学専攻修士課程在学中．属性文法の研究に従事．

五十嵐善英 (正会員)
昭和13年生．昭和37年東北大学電気工学科卒業．昭和46年同大学院工学研究科電気及通信工学専攻博士課程修了．工学博士．東北大学助手，エジンバラ大学客員研究員，リーズ大学講師，ザ・シティ大学講師，群馬大学助教授，ケンタッキー大学客員準教授を経て群馬大学工学部情報工学科教授．主に計算の理論の研究に従事．電気通信学会，ACM, BCS, EATCS, IEEE Computer Society 各会員．

佐渡 一広 (正会員)
昭和30年生．昭和52年東京工業大学理学部情報科学科卒業．昭和54年同大学院情報科学専攻修士課程修了．昭和58年同大学院博士後期課程単位修得退学．同年4月より群馬大学工学部情報工学科助手，現在に至る．理学博士．プログラミング言語，処理系，支援環境，方法論，文書処理に興味を持つ．ACM, IEEE, ソフトウェア科学会各会員．