# 右　順　位　文　法*

井　上　謙　蔵**

**Abstract**

The definition on precedence grammers by Wirth and Weber has been revised by McKeeman to seperate precedence relations into right edge ones and left edge ones of phrases.

It is shown that we can further widen the definition through throwing up the left precedence relations. The new definition, which classifies right precedence grammers, could cover wider languages and its parsing speed may be expected to be faster than the previous one. A method to overcome restrictions for right precedence grammers is discussed.

## 1. まえがき

コンパイラの製作を自動化するのに重要な一つの手段は，プログラム用言語の文法の記述を形式化して，その形式で記述しうるすべての文法の類に対する一般的な構文解析の手段を与えることである．このような類として，しばしば文脈独立文法 (context free grammer) がとられる．その理由は，それがプログラム用言語を完全に記述しうるからではなく，その上位の類をとると，構文解析の手順が複雑となって，ほとんどものの役に立たないからである．

しかし，現実には文脈独立文法に対しては，解析の方法は不確定的であるから，速度がおそい．その上，通常のプログラム用言語では，文脈独立文法よりも下位のもので記述できる部分が多い．たとえば，変数や手続きなどの名前の同一性については，文脈独立文法でも表現できないが，名前そのものや数の生成規則 (production) は，正則文法 (regular grammer) の形に書き直すことができる．名前の同一性とは，一つの名前には同一ブロック内で宣言されたものと同じ変数や手続きなどが指定されるという意味である．その他にも，かなりの部分を正則文法の形に表現することが可能であるから，実際に，ALGOL 60の生成規則を適当に変形して，複数個の有機的に結合された有限オートマトン (finite state automaton) として，解析プログラムを作る考案がなされている[1]．

それゆえ，文脈依存的な部分は別として，文法を適当に制限して，最適の解析手段を見い出す試みがなされてきた．その一つは，文法に追加して，記号間に順位関係をつけることによって，文法を制限し，句を確定的に決定できるようにする方法である．これは最初に Floyd によって演算子順位文法 (operator precedence grammer) として定義され[2]，のち Wirth によって精密化され，順位文法となった[3]．もう一つは，直接に文脈を参照することによって，句を確定的に見い出す方法である．これに属するものとしては，遷移マトリクス法[4]，$(m, n)$ 位文脈文法 (bounded context grammer)，LR $(k)$ 文法などがある[6]．

順位文法の構文解析は，速度の点では文脈をみるものとかわらないが，解析に使われる表の大きさは，かなり小形になる．問題点は，順位文法の類にはいらない場合に，文法を書き直す方法が，必ずしも機械的でないことである．この点については，Presser によって解決の手段が見い出されているが，われわれは部分的に順位関係を廃止することによって，表を小形化し，速度を速め，かつ文法の類をひろげる方法があることを示そう．

4節までに，5節以降の説明に必要な概念を与え，5, 6, 7 節で，順位文法の拡張としての右順位文法の解析方法と定義を与え，制限の問題を論ずる．

## 2. 正準解析列[3]

一般に文脈独立 (context froe) 文法を

$$G=(V_N, V_T, P, S) \qquad (1)$$

と表わす．ただし，$V_N$, $V_T$, および $P$ は，それぞれ，変数，端記号 (terminal symbol) および生成規則 (production) の集合であるとする．また，$S\in V_N$ で，$S$ は出発 (start) 記号とする．

* Right Precedence Grammers, by Kenzo Inoue (Software Technology, Fujitsu Ltd.)

** 富士通株式会社・ソフトウェア技術部

文形 (sentential form), $\eta$ と $\xi$ との間に

$$\eta=\beta A\gamma,\ \xi=\beta\alpha\gamma,\ A\to\alpha\in P \qquad (2)$$

なる関係があるとき，$\eta$ は $\xi$ を直接導びく (directly derive) といい，$\xi$ は $\eta$ に直接還元される (directly reduce) という．この関係を

$$\eta\Rightarrow\xi \qquad (3)$$

と表わす．ただし

$$\left.\begin{aligned}&A\in V_N,\ \alpha,\beta,\gamma,\eta,\xi\in V^*\\&V=V_N\cup V_T\end{aligned}\right\} \qquad (4)$$

で，$V^*$ は，$V$ の要素の任意の列からなる集合である．また，文形 $\eta$, $\eta_1$, $\eta_2$, ……$\eta_n$ および $\xi$ があって

$$\eta\equiv\eta_0\Rightarrow\eta_1\Rightarrow\eta_2\Rightarrow\cdots\cdots\Rightarrow\eta_{n+1}\equiv\xi \qquad (5)$$

である関係を

$$\eta\overset{*}{\Rightarrow}\xi \qquad (6)$$

と表わし，$\eta$ は $\xi$ を導びく．または $\xi$ は $\eta$ に還元されるという．関係 (6) の一つの導出

$$\eta_i\Rightarrow\eta_{i+1}$$

に適用される生成規則を $p_i$ と名付けることにすれば，$\xi$ から $\eta$ への還元に対して

$$p_np_{n-1}\cdots\cdots p_0\equiv[p]_{\xi,\eta} \qquad (7)$$

なる解析列(parse) が得られる．同じ導出樹 (derivation tree) を作る場合でも，生成規則の適用の順序には任意性があるから，複数個の解析列ができる．これらの中で最右端導出を逆にたどる還元に対応するものを正準解析列 (canonical parse) と呼べば，正準解析列は一つの導出樹に一つだけ決まる．

正準解析列は，つぎのように定義される．文形 $\eta$ を

$$\eta\equiv A_1A_2\cdots\cdots A_l,\ A_i\in V,\ 1\leq i\leq l \qquad (8)$$

とすると

$$\xi\equiv\gamma_1\gamma_2\cdots\gamma_l,\ A_i\overset{*}{\equiv\!\!>}\gamma_i,\ \gamma_i\in V^*,\ 1\leq i\leq l \qquad (9)$$

と表わすことができる．ただし，$A_i\overset{*}{\equiv\!\!>}\gamma_i$ は $A_i\overset{*}{\Rightarrow}\gamma_i$ または $A_i=\gamma_i$ である．こうすると $\eta\overset{*}{\Rightarrow}\xi$ に対する解析列の一つは

$$[p]_{\xi,\eta}=[p]_{\gamma_1,A_1}[p]_{\gamma_2,A_2}\cdots\cdots[p]_{\gamma_l,A_l} \qquad (10)$$

である．ことごとくの $[p]_{\gamma_i,A_i}$ が正準解析列であるとき，$[p]_{\xi,\eta}$ を $\xi$ から $\eta$ への正準解析列という．(9) において

$$\left.\begin{aligned}&A_i=\gamma_i\ \text{であれば}\ [p]_{\gamma_i,A_i}=\Lambda\\&A_i\Rightarrow\gamma_i\ \text{であれば}\ [p]_{\gamma_i,A_i}=A_i\to\gamma_i\end{aligned}\right\} \qquad (11)$$

で，いずれも一義的で，正準解析列である．(11) より出発して (10) を回帰的に適用することにより，正準解析列は一義的に決まる．ただし $\Lambda$ は空列を表わす．とくに，$\eta=S$ であるとき，正準解析列を

$$[p]_{\xi,S}\equiv[p]_{\xi} \qquad (12)$$

と表わすことにする．

文法があいまいであると，複数個の正準解析列が存在する．あいまいでない場合は，対応する導出樹は一義的に決まるから，正準解析列も一義的に決まる．

## 3. 単純順位文法[3)]

文形中に隣り合う記号間に，適当な順位関係をつけることによって，正準解析列が一義的に決まる文法の類，単純順位文法を定義する．

文脈独立文法 (1) に対して，文法

$$\left.\begin{aligned}&G'=(V_N',V_T',P',S'),\ V_N'=V_N\cup\{S'\}\\&V_T'=V_T\cup\{\vdash,\dashv\},\ V'=V_N'\cup V_T'\\&P'=P\cup\{S'\to\vdash S\dashv\}\end{aligned}\right\} \qquad (13)$$

がつぎの制限を持つとき，$G'$ を単純順位文法という．

(i) $A\in V_N'$ に対して，必ず

$$A\overset{*}{\Rightarrow}u,\ u\in V_T'^*$$

を作ることができる．

(ii) $A\to\alpha$, $B\to\alpha$, $A\neq B$ である生成規則はない．

(iii) $S'\overset{*}{\Rightarrow}\alpha AB\beta$, $A$, $B\in V'$

である記号 $A$ と $B$ の間に，非可換で一義的な順位関係が決まる．

非可換で一義的な順位関係は，つぎのように決められる．まず，変換 $A$ に対して，集合 $L(A)$ および $R(A)$ をつぎのように定義する．

$$\left.\begin{aligned}&L(A)=\{F\mid A\overset{*}{\Rightarrow}F\alpha,\ A\in V_N',F\in V',\alpha\in V'^*\}\\&R(A)=\{F\mid A\overset{*}{\Rightarrow}\alpha F,\ A\in V_N',F\in V',\alpha\in V'^*\}\end{aligned}\right\} \qquad (14)$$

これらの集合を用いて，順位関係がつぎのように決められる．$A,D,E\in V_N'$, $B,C\in V'$, $\beta,\gamma\in V'^*$ に対して

(a) $A\to\beta BC\gamma$ であれば，$B$ の順位は $C$ に等しい．すなわち $B\doteq C$.

(b) $A\to\beta BE\gamma$, $C\in L(E)$ であれば，$B$ の順位は $C$ より低い．すなわち $B\lessdot C$.

(c) $A\to\beta DC\gamma$ で $B\in R(D)$，または $A\to\beta DE\gamma$ で $B\in R(D)$, $C\in L(E)$ ならば，$B$ の順位は $C$ より高い．すなわち $B\gtrdot C$.

正準解析列を作ること，すなわち，構文解析のためには，この順位関係をつぎに述べるマトリクスの形に表わして使用する．$V$ の要素を全部1回ずつとって並べ，右端に $\dashv$ または $\vdash$ をつけ加えたものを

$Q_1$, $Q_2$, ……, $Q_N$, $Q_{N+1}$ $\quad N=V$ の要素数

とする．$Q_1,\cdots,Q_{N+1}$ を行及び列に対応させて

| | $A$ | $B$ | $a$ | $b_1$ | $b_2$ | $\vdash$ |
|---|---|---|---|---|---|---|
| $A$ | $\gtrdot$ | $\gtrdot$ | $\gtrdot$ | $\gtrdot$ | $\gtrdot$ | $\doteq$ |
| $B$ | $\doteq$ | $\lessdot$ | $\doteq$ | $\lessdot$ | $\doteq$ | ⊙ |
| $a$ | $\gtrdot$ | $\gtrdot$ | $\gtrdot$ | $\gtrdot$ | $\gtrdot$ | ⊙ |
| $b_1$ | $\gtrdot$ | $\gtrdot$ | $\gtrdot$ | $\gtrdot$ | $\gtrdot$ | ⊙ |
| $b_2$ | $\gtrdot$ | $\gtrdot$ | $\gtrdot$ | $\gtrdot$ | $\gtrdot$ | $\gtrdot$ |
| $\perp$ | $\doteq$ | $\lessdot$ | ⊙ | $\lessdot$ | ⊙ | ⊙ |

文法
$G'=(\{S', A, B\}, \{a, b_1, b_2, \vdash, \dashv\}, P', S')$
$P'=\{S'\to\vdash A\dashv, A\to Bb_2, B\to b_1, B\to Ba, B\to BA\}$

図1　Wirth の順位表

$(N+1)$ 行，$(N+1)$ 列

のマトリクスを作る．このマトリクスの要素を $M_{l,m}$ または $(Q_l, Q_m)$ と表わす．ただし，第$(N+1)$ 列は $\dashv$ に対応し，第 $(N+1)$ 行は $\vdash$ に対応するものとする．これらの要素に

$Q_l \gtrdot Q_m$ ならば $M_{l,m}=\gtrdot$
$Q_l \doteq Q_m$ ならば $M_{l,m}=\doteq$
$Q_l \lessdot Q_m$ ならば $M_{l,m}=\lessdot$

を書き込む．順位関係のない記号 $Q_l$ と $Q_m$ については，$M_{l,m}$ には⊙を書きこむ（**図1** 参照）．

正準解析列を作るには，文形中に最左端の句，すなわち，把手 (handle) を見い出し，これを対応する生成規則に従って，変数に置換する過程を繰り返せばよい．この手順を示すために，まず文形を

$$A_0'A_1'\cdots A_{k'}'A_kA_{k+1}\cdots A_{n+1},\quad A_0'=\vdash,\quad A_{n+1}=\dashv \tag{15}$$

とする．$A_0'A_1'\cdots A_{k'}'$ は，その尻を除いて，可能な還元のなされた部分で，$A_kA_{k+1}\cdots A_{n+1}$ はまだ走査されない部分である．それゆえ

$A_0', A_1', \cdots, A_{k'}'\in V', A_k, A_{k+1}, \cdots, A_{n+1}\in V_T'$

である．手順：

$(p_1)$　最初に，$k'=0$, $k=1$, $A_0'=\vdash$ とする．
$(p_2)$　$A_{k'}'=Q_l$, $A_k=Q_m$ なる $M_{l,m}$ をみる．

$M_{l,m}$ が⊙であれば，隣り合ってはいけない記号が隣り合っているので，文形はこの文法に属するものでないから，手順は中止．

$M_{l,m}=\doteq$ または $M_{l,m}=\lessdot$ の場合は $A_k=\dashv$ なら解析完了で停止，それ以外では，$k':=k'+1$, $A_{k'}':=A_k$, $k:=k+1$ として $(p_2)$ にもどる．$:=$ は置換を意味する．$M_{l,m}=\gtrdot$ ならば $j:=k'$ として $(p_3)$ に進む．

$(p_3)$　$A_{k'-1}'=Q_l$, $A_{k'}'=Q_m$ なる $M_{l,m}$ をみる．

$M_{l,m}=\doteq$ なら $k':=k'-1$ を行ない $(p_3)$ を繰り返す．$M_{l,m}=\lessdot$ なら $i:=k'$ として，$(p_4)$ に進む．$M_{l,m}=$⊙ なら誤りとして中止．

$(p_4)$　$A_i'\cdots A_j'$ は把手であるから，生成規則 $A\to A_i'\cdots A_j'$ を正準解析列の右端に加え

$k':=i$, $A_{k'}':=A$ を行なってのち $(p_2)$ にもどる．

ここに述べた手順では，$A_{k'}$ の左に関係$\gtrdot$をもつ隣接記号はないから，句の右端が $A_0'\cdots A_{k'}'$ の中に存在することはない．また，順位関係は一義的であるから，二つの句が完全に一致することなしに，部分を共有することはない．そこで，$A_i'\cdots A_j'$ は把手であり，かつ，文法はあいまいでないことがわかる．

## 4. 右順位表と右順位表

すでに述べたことから，(15) の $A_k$ は端記号であって，把手の右端を調べるさいには，$A_{k'}'\gtrdot A_k$ の関係さえ求めればよい．そこで，$N_T$ を $V_T'$ の要素の数とすれば，把手の右端を捜す順位表としては，$N_T+1$ 列，$N+1$ 行のもので充分であるし，その要素 $M_{l,m}$ は$\leqq$か$\gtrdot$が記入されるか，または⊙であればよい．左端を捜す順位表は前と同様に，$(N+1)^2$ マトリクスであるが，その要素は$\doteq$か，$\lessdot$か，または⊙である[8]（**図2** 参照）．

| | $a$ | $b_1$ | $b_2$ | $\vdash$ |
|---|---|---|---|---|
| $A$ | $\gtrdot$ | $\gtrdot$ | $\gtrdot$ | $\lessdot$ |
| $B$ | $\leqq$ | $\leqq$ | $\leqq$ | ⊙ |
| $a$ | $\gtrdot$ | $\gtrdot$ | $\gtrdot$ | ⊙ |
| $b_1$ | $\gtrdot$ | $\gtrdot$ | $\gtrdot$ | ⊙ |
| $b_2$ | $\gtrdot$ | $\gtrdot$ | $\gtrdot$ | $\gtrdot$ |
| $\perp$ | ⊙ | $\leqq$ | ⊙ | ⊙ |

右順位表
文法は図1のものと同じ

| | $A$ | $B$ | $a$ | $b_1$ | $b_2$ | $\dashv$ |
|---|---|---|---|---|---|---|
| $A$ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | $\doteq$ |
| $B$ | $\doteq$ | $\lessdot$ | $\doteq$ | $\lessdot$ | $\doteq$ | ⊙ |
| $a$ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ |
| $b_1$ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ |
| $b_2$ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ |
| $\perp$ | $\doteq$ | $\lessdot$ | ⊙ | $\lessdot$ | ⊙ | ⊙ |

左順位表

図2　McKeeman の順位表

このように順位表を作ると，順位が一義的に決まらなくなった場合に，右端で二重になったのか，左端で二重になったのかが，よくわかる．このことを利用して，Presser は，順位関係を一義的にするために，文法を書き換える機械的な方向を示した[9]．

右順位関係は，つぎのようにして作られる．まず，前節で定義した集合 $L(A)$, $R(A)$ のほかに，集合 $L_T(A)$ を定義する．

$A\in V_T'$ ならば $L_T(A)=\{A\}$
$A\in V_N'$ ならば $L_T(A)=\{F \mid A\overset{*}{\Rightarrow}F\alpha, F\in V_T'\}$,
$B\in V'$, $\beta$, $\gamma\in V'^*$ として
（i）　$A\to\beta BE\gamma$ で $C\in L_T(E)$ ならば $B\leqq C$
（ii）　$A\to\beta DE\gamma$ で $B\in R(D)$, $C\in L_T(E)$ ならば $B\gtrdot C$

左順位関係は，$B$, $C\in V'$ として
（i'）　$A\to\beta BC\gamma$ ならば $B\doteq C$

(ii′)　$A\to\beta BD\gamma$, $C\in L(D)$ ならば $B\lessdot C$ である.

順位表を右と左に分けた場合の構文解析の手順は,把手の右側を捜すのに右順位表,左端を捜すのに左順位表を使用する以外は,前節のものと同じである.

右順位が一義的でなくなるのは,つぎのような生成規則がある場合である.$C\in V_T$ として

($r_1$)　$A\to\beta BE\gamma$ かつ $B\in R(B)$, $C\in L_T(E)$

($r_2$)　$A\to\beta BE\gamma$, $A'\to\beta' DE'\gamma'$ に対して

$B\in R(D)$, $C\in L_T(E)$, $C\in L_T(E')$

いずれの場合でも, $B\leqq C$ で,かつ $B\gtrdot C$ となる.このような場合には,新しい変数 $B'$ と,その生成規則 $B'\to B$ を導入し

$A\to\beta BE\gamma$ を

$A\to\beta B'E\gamma$ および $B'\to B$

に書き換える必要がある.

左順位に対しても同様であるが,ここでは必要ないので省略する.

## 5. 右順位解析

順位文法に基づく解析方法は,文脈を調べるものに比べて表が小さい.それでも3節の方法では,大きさが $(N+1)^2$ のマトリクスを必要とするし,4節の方法では $(N+1)^2$ および $(N+1)(N_T+1)$ のマトリクスを必要とする.計算機上に,これを実現すると,一要素2ビットでよいけれども,実用的なプログラム用言語に対して,かなり大きくなる.たとえば,ALGOL 60 に対して,$N\sim150$, $N_T\sim50$ であるから,後者の方法で 60,000 ビットを必要とする.そこで,マトリクスの代わりに,$2N$ 個のデータですむ順位関数を用いる方法があるが,順位関係が一義的であるのに,順位関数が定義できない場合がおこるので,必ずしも一般的な方法ではない[3].

順位表を使用する解析では,把手の左右両端が発見されてから,それがいずれの生成規則の右辺と一致するかを,もう1回調べなければならない.それゆえ,順位表のほかに各句を識別するための構文表を必要とする.これはいわば二重の手続きである.

われわれは,すでに順位関係が一義的であるならば,左順位表を不用とする解析方法があることを示した.この方法は,左順位表が不用であるばかりでなく,把手の左端が見い出されたときは,対応する生成規則が同時に認識されるような手順である[10].

ここでは,さらに,左順位関係が一義的でない場合にも,同様な方法が適用できることを示そう[11].この方法は,順位文法をはみでる言語にも適用できるので,新しい文法の類,右順位文法を定義するものである.この拡張は,$(m,1)$ 位文脈文法 (bounded context grammer) から,$LR(1)$ 文法への拡張に対応する(**図3**参照).

| | + | × | ( | ) | $i$ | ⊣ |
|---|---|---|---|---|---|---|
| $E$ | ≦ | ⊙ | ⊙ | ≦ | ⊙ | ≦ |
| $A$ | ≫ | ≦ | ⊙ | ≫ | ⊙ | ≫ |
| $B$ | ≫ | ≫ | ⊙ | ≫ | ⊙ | ≫ |
| + | ⊙ | ⊙ | ≦ | ⊙ | ≦ | ⊙ |
| × | ⊙ | ⊙ | ≦ | ⊙ | ≦ | ⊙ |
| ( | ⊙ | ⊙ | ≦ | ⊙ | ≦ | ⊙ |
| ) | ≫ | ≫ | ⊙ | ≫ | ⊙ | ≫ |
| $i$ | ≫ | ≫ | ⊙ | ≫ | ⊙ | ≫ |
| ⊥ | ⊙ | ⊙ | ≦ | ⊙ | ≦ | ⊙ |

右順位表

| | $E$ | $A$ | $B$ | + | × | ( | ) | $i$ | ⊣ |
|---|---|---|---|---|---|---|---|---|---|
| $E$ | ⊙ | ⊙ | ⊙ | ≒ | ⊙ | ⊙ | ≒ | ⊙ | ≒ |
| $A$ | ⊙ | ⊙ | ⊙ | ⊙ | ≒ | ⊙ | ⊙ | ⊙ | ⊙ |
| $B$ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ |
| + | ⊙ | ≒ | ≪ | ⊙ | ⊙ | ≪ | ⊙ | ≪ | ⊙ |
| × | ⊙ | ⊙ | ≒ | ⊙ | ⊙ | ≪ | ⊙ | ≪ | ⊙ |
| ( | ≒ | ≪ | ≪ | ⊙ | ⊙ | ≪ | ⊙ | ≪ | ⊙ |
| ) | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ |
| $i$ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ | ⊙ |
| ⊥ | ≒ | ≪ | ≪ | ⊙ | ⊙ | ≪ | ⊙ | ≪ | ⊙ |

左順位表

文法　$G'=(\{S',E,A,B\},\{+,*,(,),i,\vdash,\dashv\},P',S')$

$P'=\{S'\to\vdash E\dashv, E\to E+A, E\to A, A\to A\times B, A\to B, B\to(E), B\to i\}$

左順位表は参考のために示された.この中で $(\vdash,E)$, $((E)$ および $(+,A)$ の順位が二重となるので, $G'$ は単純順位文法ではないが,単純右順位文法である.

図3　単純右順位文法の例

まず,構文解析の方法を述べよう.文形は (15) にあげられたものとする.生成規則を並べた構文表および右順位表を用意する.

手順:

($p_1$)　$k'=0$, $k=1$, $A_0'=\vdash$ とする.

($p_2$)　$Q_l=A_{k'}'$, $Q_m=A_k$ である $M_{l,m}$ をみる.

$M_{l,m}=\odot$ なら,誤りとして停止.

$M_{l,m}=\leqq$ で $A_k=\dashv$ なら,解析完了で停止.

$M_{l,m}=\leqq$ で $A_k\neq\dashv$ なら, $k':=k'+1$,

$A_{k'}':=A_k$, $k:=k+1$ とし ($p_2$) にもどる.

$M_{l,m}=\gtrdot$ なら, $j:=k'$, ($p_3$) へ進む.

($p_3$)　構文表にある生成規則で

$B\to C_1C_2\cdots C_{b-1}C_b$, $C_b=A_j'$

であるものを取り上げ, $k'=1, 2, \cdots, b-1$ に対し,順次 $C_{b-k'}$ と $A_{j-k'}'$ を照合して,その一致を調べる.$C_1$に至るまで一致する生成規則が2個以上あれば,その最長のものに対して,

$i:=j-b'+1$

とおく.このような規則がなければ誤りとして中止.

($p_4$)　$A_i'A_{i+1}'\cdots A_j'$ は把手であるから

$B\to C_1C_2\cdots C_b$

を正順解析列の右端に加えて

$k':=i$, $A_{k'}':=B$

を行ない，($p_2$) にもどる．

手順 ($p_3$) で，右辺が$\cdots A_j'$ と一致する生成規則が $t$ 個あるとして，それを短い順に書き並べて

$$B_s\to\alpha_s\alpha_{s-1}\cdots\alpha_1,\quad 1\leq s\leq t,\quad \alpha_s\in V^* \qquad (16)$$

とする．また，$\alpha_s$ の頭と尾を $H(\alpha)$ および $T(\alpha)$ と表わし，とくに長さを指定したいときは，$H_i(\alpha)$，$T_i(\alpha)$ と表わすことにする．もし，$\alpha$ の長さが $n$ であれば

$$H_i(\alpha)\,T_{n-i}(\alpha)=\alpha,\quad n=|\alpha| \qquad (17)$$

である．また $H_0(\alpha)=\Lambda$ とする．構文表には，各生成規則の右辺はまとめられ

$$\alpha_t\alpha_{t-1}\cdots\alpha_1 \qquad (18)$$

の形で記録されている．$T_1(\alpha_1)$ から左へ，$A_{j-k'}'(k'=1,2,3,\cdots)$ との照合が行なわれ，$H_1(\alpha_1)$ まで一致したとき，$k'$ の値が記録される．照合はさらに左へ続けられる．一般に $H_1(\alpha_s)$ まで一致したとき，$k'$ の値の記録は，$H_1(\alpha_{s-1})$ に対するものから，$H_1(\alpha_s)$ に対するものに更新される．

$\alpha_{s+1}$ の中で，一致しない記号があれば

$$\alpha_s\alpha_{s-1}\cdots\alpha_1$$

が，最長の句である．$\alpha_t$ まで一致すれば

$$\alpha_t\alpha_{t-1}\cdots\alpha_1$$

が最長の句である．

尾が一致して，頭の異なる右辺を持つ生成規則は，共通部分を共有する形に記録される．

これを

$$B_1\to\alpha\beta,\quad B_2\to\alpha'\beta$$

とすれば，いずれの適用が可能であるかは，$n=|\beta|$ とするとき $A_{j-n}'$ を調べればわかる．

このようにして，照合の手順は，順位文法に対して，右順位だけを使用する方法に比べ，ある時点で $k'$ の値を一つだけ記録することを除いては，とくに複雑にならない．

## 6. 右順位文法

前節の方法は，二つの生成規則

$$A\to\alpha\beta,\quad B\to\beta$$

があって，$T(\alpha)\beta$ が $A_{j-k'}'\cdots A_j'$ と一致し，$T(\alpha)\neq\Lambda$，$T(\alpha)\neq\alpha$ であるときも，$\beta$ を句とする文法である．順位文法であれば，このようなことは許されない．なぜなら $\beta$ が何であるためには $T_1(\alpha)<H_1(\beta)$ で，$T(\alpha)\beta$ が $A_{j-k'}'\cdots A_j'$ と一致するためには，$T_1(\alpha)\doteq H_1(\beta)$ であるから，単純順位文法の要求3節の (iii) と矛盾するからである．

このような解析方法を許す文法は，文法 (13) に対しつぎの制限を設けたものである．

(i)，(ii)　3節の (i)，(ii) と同じ．

(iii)　$S'\overset{*}{\Rightarrow}\alpha AB\beta,\ A,B\in V$

である記号 $A$ と $B$ の間に，非可換で一義的な右順位関係が存在する．

(iv)　生成規則　$A\to\alpha\beta$，$B\to\beta$ があるときは，

$$C\to\gamma H_i(\alpha)\,D\delta,\quad D\overset{*}{\equiv\!\rangle}T_{n-i}(\alpha)\,B\delta'$$

は存在しない．ここで $n=|\alpha|$，$n\geq i>0$ である．また，任意の $\xi$，$\eta$ について $\xi\overset{*}{\equiv\!\rangle}\eta$ は $\xi\overset{*}{\Rightarrow}\eta$，または $\xi=\eta$ を表わす．

このような，文法を単純右順位文法と呼ぶことにする．

単純右順位文法に対して，生成規則

$$A\to\alpha\beta,\quad B\to\beta$$

がともに $\cdots A_j'$ の部分に一致したときは，$\alpha\beta$ が把手であることを示す．

まず，$B\to\beta$ を適用して，文形が

$$\cdots\alpha BA_k\cdots$$

に還元されたとする．$T_1(\alpha)\geqslant B$ であるとすれば，$T_1(\alpha)\geqslant H_1(\beta)$ でなければならないから，そのようなことはない．そこで $B$ の左に何はない．つぎに

$$B\leq A_k \text{ か } B\geqslant A_k$$

である．$B\leq A_k$ であれば，走査が右に進められ，$B$ を含んだ句か，あるいは含まない句が右に作られ，還元される．後者がおこった場合を考えて，文形を

$$\cdots\gamma\alpha B\delta_0u_0\cdots,\quad \gamma\in V'^*,\quad \delta_0\overset{*}{\equiv\!\rangle}u_0',\quad u_0',u_0\in V_T'^*$$

($\gamma,u_0'$ は $\Lambda$ でもよい)

と表わす．そうすると，$B$ を含む還元は

$$C_1\to T_{i_1}(\alpha)\,B\delta_0 \text{ または，} C_1\to\gamma\alpha B\delta_0$$

の形の生成規則によって行なわれる．

$T_{i_1}(\alpha)$ の長さは任意で，0 でもよい．ただし，$T_{i_1}(\alpha)\neq\alpha$ とする．このうち後者は (iv) の規則で禁止されているから，前者によって，文形は

$$\cdots\gamma H_{n-i_1}(\alpha)\,C_1u_0\cdots$$

となる．再び $T_1(H_{n-i_1}(\alpha))\leqslant C_1$ であることはたしかであるから，$C_1$ を含む還元は

$$C_2\to T_{i_2}(H_{n-i_1}(\alpha))\,C_1\delta_1,\quad C_2\to\gamma H_{n-i_1}(\alpha)\,C_1\delta_1$$

の形の生成規則によることになる．ここで後者は (iv) で禁止されているから，前者によって文形は

$$\cdots\gamma H_{n-i_1-i_2}(\alpha)\,C_2u_1\cdots$$

となる．ただし

$$\delta_1\overset{*}{\equiv\!\rangle}u_1',\quad u_1'u_1-u_0,\quad H_{n-i_1-i_2}(\alpha)\neq\Lambda$$

で $u_1'=\Lambda$ でもよい．このように続けて

$$C_j \to T_{ij}(H_{n-i_1-\cdots-i_{j-1}}(\alpha))C_{j-1}\delta_{j-1}$$

によって，文形が

$$\cdots\gamma H_{n-i_1-\cdots-i_j}(\alpha)C_j u_j\cdots,\quad H_{n-i_1-\cdots-i_j}(\alpha)\neq\Lambda$$

になったとする．還元の手順によって

$$C_j \overset{*}{\Longrightarrow} T_{i_1+i_2+\cdots+i_j}(\alpha)Bu_0'u_1'\cdots u_{j-1}'$$

であるから，規則（iv）によって，$\alpha$ の頭を還元しつくすことはできない．すなわち，$\beta$ を句と判定したのは間違いである．他に適用可能な生成規則は

$$A\to\alpha\beta$$

だけであるから，$\alpha\beta$ が句である．$\alpha\beta$の左に他の句はないから，$\alpha\beta$ が把手である．$\alpha\beta$ も句にならない場合は，そのような文は，この文法に従わないものである．

## 7. 制限の除去

右順位文法の制限のうち，とくに問題となるのは，6節の（ii）と（iii）である．（iv）は実用のプログラム用言語に対しては，あまり問題とならないであろう．

右順位関係が一義的にならない場合に対する Presser の解法は4節の終わりに述べた．6節の（iv）の2行目に示された生成規則があるときは，新しく

$$C\to\gamma A\delta$$

を追加すればAの規則の適用によってもCに還元できるが，そうすると文法があいまいになるから，結局文法全体を再編成しなければならない．6節の（iii）と（iv）にぶつかる文法は，それぞれの規則に従い機械的に決定できる．実際，このような文法を矛盾のないもので置き換えることは，新しい変数を導入することになるから，人間の判断によったほうが好ましい．

（ii）を破る場合，すなわち，右辺の一致する生成規則が複数個ある場合は，文脈をみないと，いずれが適用可能であるかは，判断できない．この場合，見るべき文脈は機械的に作り出すことができる．まず

$$B_i\to\beta \qquad i=1,2,\cdots \tag{19}$$

とする．最右端導出

$$C_{ji}\overset{*}{\Longrightarrow}\cdots X_{ji1}X_{ji(l-1)}\cdots X_{ji0},\quad X_{ji0}=B_i \tag{20}$$
$$j_i=1,2,\cdots,\quad i=1,2,\cdots$$

に対して，集合 $\{B_i\}$ の部分集合

$$N(Y_1)=\{X_{j0}\mid X_{j1}=Y_1\} \tag{21}$$

を可能なすべての $Y_1$ について作る．$N(Y_1)$ の要素数が2またはそれ以上のものに対しては，その部分集合

$$N_{Y_1}(Y_2)=\{X_{j0}\mid X_{j2}=Y_2,\ X_{j0}\in N(Y_1)\} \tag{22}$$

を可能なすべての $Y_2$ について作る．このように繰り返していくと，すべて要素数1の部分集合に分解することができるから，それを

$$\bar{N}_{Y_{l-1}Y_{l-2}\cdots Y_1}(Y_l)=\{X_{j0}\mid X_{jl}=Y_l,\ X_{j0}\in N_{Y_{l-2}\cdots Y_1}(Y_{l-1})\} \tag{23}$$

と表わす．

$$S\overset{*}{\Rightarrow}\beta_i B_i\cdots,\quad S\overset{*}{\Rightarrow}\beta_{i'}B_{i'}\cdots,\quad \beta_i\neq\beta_{i'}\quad(i\neq i')$$

であれば，このような分解は可能であるし，通常のプログラム用言語は，これを満たしている．(23)のように分解されたときには，記号列

$$Y_lY_{l-1}\cdots Y_1 \tag{24}$$

に対して，一つの $B_i=X_{j0}$ が対応する．それゆえ，ここに述べた分解の手順に従って，文形が

$$\cdots A_{k'-1}'A_{k'}'A_kA_{k+1}\cdots$$

で

$$A_{k'-m+1}'\cdots A_{k'}'=\beta$$

の場合に，$A_{k'-m}'=Y_1$ である集合 $N(Y_1)$ の要素が複数であれば，$A_{k'-m-1}'$ を調べ，$A_{k'-m-1}'=Y_2$ である集合 $N(Y_1)$ の部分集合 $N_{Y_1}(Y_2)$ の要素が複数であれば，さらに $A_{k'-m-2}'$ を調べる．このようにして，そのときの $\beta$ が，いずれの $B_i$ になるかを決定できる．この過程は，実際には必要な $Y_l\cdots Y_1$ を並べておき，$Y_1$ より出発して，いずれの $Y_l\cdots Y_1$ が $\cdots A_{k'-m}'$ と一致するかを調べればよい．実用のプログラム用言語では $l=1$ ないし2になるように文法を書き改めるのは非常に容易である．

例として ALGOL 60 の算術式の場合をあげる．算術式によって，定義される変数にはつぎのものがある[12]．

〈下限〉→〈算術式〉 〈上限〉→〈算術式〉
〈繰り返し要素〉→〈算術式〉 〈式〉→〈算術式〉
〈添字式〉→〈算術式〉

このうち〈下限〉と〈上限〉については

〈上下限〉→〈算術式〉：〈算術式〉

として対照からはずすことにする．これは $l=2$ とするための考慮である．残りの3規則と，〈算術式〉を含む他の規則との関連は，ここでは考えずに，これらを区別する文脈を作り出すことだけを問題とする．まず

$N(,)=\{$〈添字式〉,〈式〉,〈繰り返し要素〉$\}$
$\bar{N}([)=\{$〈添字式〉$\}$
$\bar{N}(()=\{$〈式〉$\}$

である．$N(,)$ は3要素を含むから

$\bar{N}$,(〈添字の並び〉)$=\{$〈添字式〉$\}$
$\bar{N}$,(〈実パラメタの並び〉)$=\{$〈式〉$\}$
$\bar{N}$,(〈繰り返し要素の並び〉)$=\{$〈繰り返し要素〉$\}$

表1　同一右辺を持つ生成規則に対する文脈表の例

| $Y_2$ | $Y_1$ | $N$ |
|---|---|---|
| 〈添字の並び〉 | | 〈添字式〉 |
| 〈実パラメタの並び〉 | , | 〈式〉 |
| 〈繰り返し要素の並び〉 | | 〈繰り返し要素〉 |
| | [ | 〈添字式〉 |
| | ( | 〈式〉 |

〈添字式〉→〈算術式〉，〈式〉→〈算術式〉，〈繰り返し要素〉→〈算術式〉

が作られる．$Y_2Y_1$ を並べると，**表1**のようになる．

## 7. 誤まりの検出と順位マトリクスの縮約

文の誤まりにはつぎの2種類がある．まず，生成規則

$$B\to\beta,\quad B'\to\beta'\qquad B\fallingdotseq B',\quad \beta\fallingdotseq\beta'$$

があり

$$S'\overset{*}{\Rightarrow}\alpha\beta\gamma,\quad S'\overset{*}{\Rightarrow}\alpha'\beta'\gamma',\quad \gamma,\gamma'\in V_T^*$$

であるものとする．

（a）　$T_1(\alpha')\leq H_1(\beta)$ であるが，生成規則

$$C\to T(\alpha')D,\quad D\overset{*}{\equiv\!\rangle}B\gamma_0,\quad \gamma_0=H(\gamma)$$

がないため

$$\alpha'\beta\gamma$$

は正しい文にならない．

（b）　$\xi\gamma'$ に対し，$T_1(\xi)\odot H_1(\gamma')$ である．ただし $\xi\in V^*$ とする．あるいは

$$\alpha\beta\gamma'\ \text{で}\ T_1(\beta)\gtrdot H_1(\gamma')$$

であるが，生成規則

$$C\to\alpha_2D,\quad D\overset{*}{\equiv\!\rangle}\alpha_3B$$

を使用するところまで還元が進み $\alpha\beta\gamma'$ が

$$\alpha_1C\gamma',\quad \alpha=\alpha_1\alpha_2\alpha_3$$

となったところで，$C\odot H_1(\gamma')$ になった．

後者は，前者により一般的に表されている．

（b）型の場合には，$T(\xi)$ と $H(\gamma)$ を含む生成規則は，いかなる意味においても存在しない．そこで（a）でも（b）でも，適用すべき生成規則が存在しない段階があるから，構文解析の手順は乱されることなく，文の誤まりが指適される．左側の順位マトリクスがないために，（a）型の誤りの発見は，順位文法の場合に比べ，数段階遅れる場合がある．

つぎに順位マトリクスの縮約を考える．マトリクスの各要素に，順位関係として，$\leq$と$\gtrdot$のほかに，$\odot$のいずれかを記入しておかなければならない．そのために記憶装置としては，2ビットを必要とする．そこで$\odot$の代わりに$\gtrdot$を記入することにすれば，1ビットですみ，記憶装置は節約される．このようにしても，（a）型の誤りに対する，上記の議論は影響をうけない．（b）型の場合には，つぎのようになる．まず，$T_1(\xi)\odot H_1(\gamma')$ は，$T_1(\xi)\gtrdot H_1(\gamma')$ として認識されるから，生成規則が適用されようとする．この場合，つぎのことがおこる．

($b_1$)　$T_1(\xi)$ を右辺の尾に持つ生成規則がない．

($b_2$)　$T_1(\xi)$ を右辺の尾に持つ生成規則はあるが，右辺全体は $T(\xi)$ に一致しない．

($b_3$)　右辺全体が $T(\xi)$に一致する生成規則がある．

($b_1$) と ($b_2$) の場合は，直ちに誤まりとして認識される．($b_3$) の場合は．その生成規則によって $\xi$ を還元した結果を，改めて $\xi$ と書き表わすと，記号列 $\xi\gamma'$ に対して

$$T_1(\xi)\gtrdot H_1(\gamma')\qquad(25)$$

である．なぜなら還元の行なわれる前の $\xi$ に対して，本来 $T_1(\xi)\odot H_1(\gamma')$であるから，新しい $\xi$ に対しても $T_1(\xi)\odot H_1(\gamma')$ である．これは縮約された順位マトリクス上は，(25)の関係で表わされる．かくして走査は右へ進行しないから，還元のある時点で ($b_1$) または ($b_2$) の場合に到達する．

順位マトリクスの縮約の結果，誤りの発見が数段階遅れる場合があるが，解析の手順は乱されない．

誤りが認識されたならば

$$A_{k'}\leq A_k$$

になるまで，$k':=k'-1$ を行なうと，患部を過剰に切り捨る可能性があるが残部の解析に影響が少ない．ただし，記号は (15) 式の意味で使用した．

## 8. あとがき

単純右順位文法は，変数の左回帰的定義を完全に含みうる文法である．単純順位文法では，

$$A\to\beta BC\gamma,\quad C\to C\delta$$

であれば，右順位関係に矛盾をひきおこさない形で，$C\to C\delta$ を右回帰的定義で置き換えるか，前式の代わりに

$$A\to\beta BC'\gamma,\quad C'\to C$$

を用いなければならない．

右順位文法はあいまいでない．6節の制限（ⅲ）と（ⅳ）は機械的に調べることができるから，ある文法が右順位文法であること，したがって，あいまいでないことは自動的に調べることができる．もし，これらの条件にかなわない場合は，問題となる生成規則の摘発，その変更の方法の示唆を自動化することができる．

制限（ⅱ）に対しても，右辺の等しい規則が複数個

あれば，それを判別する文脈を自動的に作ることができる．これも場合の数が多すぎたり，文脈が長すぎたりするような場合を示すことによって，生成規則の変更を示唆するよう自動化できよう．ただし，文脈独立文法の外にある場合は問題外である．

ここに述べた方法について，藤崎 湛†，林 達也†，白木 誠†，阿部 克†の諸君が批判を加え，考察を精密化するきっかけを与えてくれた．この点，論文中には示されなかったが，林君の提供してくれた例題が特に役に立った。深謝する次第である。

## 参 考 文 献

1) Tixier, V.: Recursive functions of regular expressions in language analysis, Tech. Rpt. CS 58, Computer Science Dept., Stanford U., Stanford, Calif., March 1967.
2) Floyd, R. W.: Syntactic analysis and operator precedence, J. ACM 10 (July 1963), 316-333.
3) Wirth, N. and Weber, H.: Euler-a generalization of ALGOL and its formal definition Pt. 1 and 2, Comm. ACM 9, 2-3 (1966), 12-23 and 89-99.
4) Gries, D.: The use of transition matrices in Compiling. Comm. ACM 11 (Jan. 1968), 26-34.
5) Eickel, J., Paul, M., Bauer, F. L., and Samelson, K.: A syntax controlled generator of formal language procersors, Comm. ACM 6 (Aug. 1963), 451-455.
6) Knuth, D. E.: On the translation of languages from left to right, Inform. Contr. 8 (Dec. 1965), 607-639.
7) Korenjak, A. J.: A practical Method for constructing $LR(k)$ processors, Comm. ACM 12 (Nov. 1969), 613-623.
8) McKeeman.: An approach to computer language design, Tech. Rep. CS 48, Computer Science Dept., Stanford U., Stanford, Calif. Aug. 1966.
9) Presser, L.: The structure, specification and evaluation of translators and translator writing Systems, UCLA-10P14-52, Rep. No. 98-51, UCLA, Calif. Oct. 1968.
10) 井上謙蔵：順位文法の右順位解析．情報処理 11(April 1970), 231—233.
11) 井上謙蔵：右順位文法について．オートマトン研究会資料，昭和 42, 2，電子通信学会.
12) 日本工業標準調査会：電子計算機プログラム用言語 ALGOL (水準 7000) JIS C 6210，昭和 42. 5. 1.

† 富士通株式会社ソフトウェア技術部

(昭和 45 年 2 月 3 日受付)