寄　　書

順位言語の右順位解析*

井　上　謙　蔵**

1. 順位文法

文脈独立 (context free) 言語の構文解析を形式的に実行しようとすると，その文法の生成規則 (production) を不確定的に適用することになる．その結果，解析の速度はたいへん遅い．これを改善するために，いろいろな方法が考えられているが，その一つに文法にある種の制限を加えて，生成規則の適用を確定的にする順位 (precedence) 文法がある[1]．ここでは順位言語の構文解析について，すでに知られているものよりも，さらに簡単な方法を示す．

順位文法を

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

と表わそう．

ただし，V_N は非端記号 (non terminal symbol) $A, B, \ldots$ などの集合，V_T は端記号 (terminal symbol) $a, b, \ldots$ などの集合，P は生成規則の集合，S は出発記号 (start symbol) である．なお，前もって記号の説明をしておくと，$V=V_T \cup V_N$，V_T^*，V_N^*，V^* などは，それぞれ V_T，V_N，V の記号からなる列の集合，それらの要素を $\alpha, \beta, \gamma, \ldots$ などで表わす．特に長さ 0 の要素を ϵ で表わす．順位文法は文脈独立文法に，次の制限を加えたものである．

(i)　$A \in V_N$ に対して，生成規則

$$A \to \alpha,\quad \alpha \in V^*,\quad \alpha \neq \epsilon$$

が少なくとも一つはあり，そのうち一つは，

$$A \overset{*}{\Rightarrow} \beta A \gamma,\quad \beta, \gamma \in V^* \ (\beta, \gamma = \epsilon \text{ でもよい})$$

にならないものである．ここで $\overset{*}{\Rightarrow}$ は，一般に左辺の記号列から，生成規則の逐次的な適用によって右辺の記号列を導く導出関係を表わす．

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

(iii)　V の各要素の間には，位数 (m, n) の一義的な順位関係がなりたつ．

位数 (1, 1) の順位文法を (1, 1) 順位文法，または単純順位文法というが，ここでの話は単純順位文法に制限する．さて，位数 (1, 1) の順位関係とは，次のようなものである．

(a)　$A \to \alpha BC\beta$ であれば，B と C は同じ順位にあるといい，$B \doteq C$ と表わす．

(c)　$A \to \alpha DC\beta$，または $A' \to \alpha' DE\beta'$ において，$D \overset{*}{\Rightarrow} \gamma B$，$E \overset{*}{\Rightarrow} C\delta$ であれば，B は C' より順位が高いといい，$B \gtrdot C$ と表わす．

(c)　$A \to \alpha BE\beta$ で，$E \overset{*}{\Rightarrow} C\delta$ であれば，C は B より順位が高いといい，$B \lessdot C$ と表わす．

(d)　(a)，(b)，(c) 以外の記号，すなわち，文 (sentence) または文形 (sentential form) の中で隣り合うことのない記号の間には，いかなる関係もない．ただし，$B, C \in V$ とし，$\alpha, \beta, \gamma, \delta$ は ϵ でもよい．

順位文法に従う言語に対して，構文解析を行なうためには，順位表が利用される．それは V の要素の名前をつけた行と列からなる $n \times n$ マトリクスである．ただし，$n=n_N+n_T$ で，n_N, n_T はそれぞれ，V_N V_T との要素数とする．順位表を M，その要素を $M_{A,B}(A, B \in V)$ とするとき，$M_{A,B}$ には，

$A \doteq B$ であれば　$\doteq$,

$A \gtrdot B$ であれば　$\gtrdot$,

$A \lessdot B$ であれば　$\lessdot$,

が記入され，A，B が隣接しないものであれば，なに

第 1 表　順位表の例

(a) Wirth, Weker の順位表

	S	A	a	b	c
S	⋗	⋗	⋗	⋗	⋗
A	≐	⋖	≐	⋖	≐
a	⋗	⋗	⋗	⋗	⋗
b	⋗	⋗	⋗	⋗	⋗
c	⋗	⋗	⋗	⋗	⋗

(b) McKeeman の順位表

右順位表

	a	b	c
S	⋗	⋗	⋗
A	≦	≦	≦
a	⋗	⋗	⋗
b	⋗	⋗	⋗
c	⋗	⋗	⋗

左順位表

	S	A	a	b	c
S					
A	≐	⋖	≐	⋖	≐
a					
b					
c					

文法 $G=(\{S, A\}, \{a, b, c\}, P, S)$
$P=\{S \to Ac,\ A \to a,\ A \to Ab,\ A \to AS\}$

* A Syntax Analysis Technique based on Right Precedences of Precedence Languages, by Kenzo Inoue (Software Technology, Fujitsu Ltd.

** 富士通株式会社・ソフトウェア技術部

も記入されない［**第1表**（a）参照］.

この順位表によって，文の左端の最短の句，すなわち把手（handle）をみつけるには，次のようにする．与えられた文を

$$S_0S_1S_2\cdots S_{n+1}$$

とするとき，これを左より走査していく．そのさい隣り合せた記号 S_k, S_{k+1} について，

$$S_k=A,\ S_{k+1}=B\quad(k=0,\ 1,\ 2\cdots)$$

とするとき，$M_{A,B}$ をながめて，$\doteq$ か $\lessdot$ であれば，走査を続ける．$M_{A,B}=\gtrdot$ であれば，そのときの k を j とし，次に j より左へ走査する．右への走査のさいと同様に，

$$S_{k-1}=A,\ S_k=B\quad(k=j,\ j-1,\ldots)$$

とするとき，$M_{A,B}$ を調べ，$M_{A,B}=\doteq$ なら走査を続け，$M_{A,B}=\lessdot$ なら走査を止めて，そのときの k の値を i とする．最初の走査の間に，$M_{A,B}$ が空白な場合はないとすれば

$$S_{i-1}\lessdot S_i\doteq S_{i+1}\doteq\cdots S_j\gtrdot S_{j+1}$$

であって，これは，順位規則（a），（b），（c），（d）より，ある生成規則の右辺であることを示す．しかも，この中に他の句がないことは明らかであるし，走査の方法から，この句の左に，他の句はないから，これが把手である．把手は，

$$C\to S_iS_{i+1}\cdots S_j$$

によって C に置き換えられた後，次の把手をみつけるには，文形の頭にもどる必要はない．なぜなら，走査の仕方から

$$S_{k-1}\lessdot S_k\ \text{または}\ S_{k-1}\doteq S_k\ (k=i-1,\ i-2,\cdots)$$

であり，$S_{i-1}\gtrdot C$ が起こるとすれば，（b）より，

$$S_{i-1}\gtrdot S_j$$

でなければならず，これは走査の方法から，生じないからである．そこで，次に，

$$S_{i-1}=A,\ C=B$$

として，$M_{A,B}$ をながめ，$M_{A,B}=\lessdot$ か $\doteq$ なら右へ走査し，$\gtrdot$ なら左へ走査する段階から，上記の手続きを繰り返せばよい．

ここで，注目すべきことは，最初に文，すなわち端記号の列が与えられるのであるから，走査の手順上，右への走査では，右側の記号（$B=S_{k+1}$）は，必ず端記号であるということである．

2. McKeeman の拡張

McKeeman[2] は，右への走査のさい，右側の記号が常に端記号である点に着目して，順位表を，把手の右端をみつけるものと，左端をみつけるものに分けた．したがって，右順位表は $n\times n_T$ マトリクスで，左順位表は，$n\times n$ マトリクスである．右端のしるしは，$\gtrdot$ なる順位関係がみつけられる所であるから，順位表の記入項目で $\lessdot$ と $\doteq$ を区別する必要はない．そこで，$\leqq$，$\gtrdot$ または空白とする．左端のほうは，$\lessdot$ と $\doteq$ を区別しなければならない．さて，順位規則は，次のように改められる．

右順位：

(Ra) $A\to\alpha BC\beta$ か $A'\to\alpha'BE\beta'$ で $E\overset{*}{\Rightarrow}C\delta$ なら $B\leqq C$
ただし，$B\in V$, $C\in V_T$

(Rb) $A\to\alpha DC\beta$ か $A'\to\alpha'D'E\beta'$ で $D\overset{*}{\Rightarrow}\gamma B$, $D'\overset{*}{\Rightarrow}\gamma'B$, $E\overset{*}{\Rightarrow}C\delta$ なら $B\gtrdot C$
ただし，$B\in V$, $C\in V_T$.

左順位：

(La) $A\to\alpha BC\beta$ なら $B\doteq C$

(Lb) $A\to\alpha BE\beta$, $E\overset{*}{\Rightarrow}C\delta$ なら $B\lessdot C$

順位表の作り方は，前と同様である［第1表（b）］．また把手をみつけるのに，右端に対して右順位表，左端に対して左順位表を使用する以外は，前章で述べた手続きをそのまま使用する．

単純順位文法でないと，これらの順位関係が一義的に決まらないため，$M_{A,B}$ に，右なら $\leqq$ と $\gtrdot$，左なら $\lessdot$ と $\doteq$ というように，二重の記入が起こる．Presser[3] は，左右に分けられた順位関係を利用して，一般に $(m,\ n)$ 順位文法を，単純順位文法に引き下げる方法を調べた．

3. 右順位解析法

さて，拡張された順位関係の欠点は，なんといっても2個の順位表をもたなければならないということである．そこで，単純順位文法について，左順位表を廃止することを考えてみる．まず把手をみつける方法を述べる．

把手の右端をみつける手順には，前と同じく，右順位表を使用する．いま S_j がみつかったとする．次に

$$A\to\alpha\qquad T_1(\alpha)=S_j$$

なる生成規則をとりあげる．ここで $T_1(\alpha)$ は α の最右端の記号を意味する．

$$A\to B_pB_{p-1}\cdots B_1S_j$$

とする．B_1 と S_{j-1}，B_2 と S_{j-2}，… が一致するかどうかを調べる．B_p まで一致するものが一つだけあれば，そのときの $S_{j-p}S_{j-p+1}\cdots S_j$ が把手である．

このような生成規則が多数であれば，その最大の p に対する $S_{j-p}S_{j-p+1}\cdots S_j$ が把手である．このことを次に証明しよう．

いま，単純順位文法に限っているのであるから，$S_{k-1}\leqq S_k$ $(k=j,\ j-1,\cdots)$ は保証されている．したがって，S_j より左に他の句はない．そこで，

$$S_{j-p}S_{j-p+1}\cdots S_j$$

が最小の句であることを示せばよい．

$$A\rightarrow B_pB_{p-1}\cdots B_1S_j$$

が一つしかなければ $S_{i-p-1}\lessdot S_{j-p}$ であることは確かであるから問題はない．そこで

$$C\rightarrow B_qB_{q-1}\cdots B_1S_j \qquad (q<p)$$

があったとする．右辺が，このような形のものは A と C の二つだけの場合について証明すれば十分である．C の右辺が把手であるためには

(Lb) より $S_{j-q-1}\lessdot S_{j-q}$

である．$q<p$ であるから，$j-p\leqq j-q-1<j$ であって，S_{j-q-1} は S_{j-p}, S_{j-p+1}, S_{j-p+2}, $\cdots$, S_{j-1} のいずれかである．ところが，Aの右辺と (La) から，

$$S_{j-q-1}\doteq S_{j-q}$$

でなければならない．そこで，$S_{j-q}S_{j-q+1}\cdots$, S_j は句にならない．それゆえ，$S_{j-p}S_{j-p+1}\cdots S_j$ が句でなければならない．したがって，左右ともに (1, 1) 順位であることが保証されている場合には，把手を決定するのに右順位表だけあれば十分である．

左端を順位表にしたがって求める方法では，把手の両端がわかったにすぎないから，もう1回，それがどの生成規則の右辺になるかを調べなければならない．このような二重の手続きは排除される．また右端の一致する生成規則，たとえば，

$$A\rightarrow\alpha\beta,\quad B\rightarrow\gamma\beta$$

があったとき，$T_1(\alpha)$ の一致がとれなかった場合に，$T_1(\beta)$ にもどる必要はなく，$T_1(\gamma)$ から調べるようにプログラムを作ることは容易である．したがって，この方法では，左右順位表を用いる手段より，解析速度が速いことが期待される．

参考文献

1) N. Wirth and H. Weber: "EULER-a generalization of ALGOL and its formal definition," Pt. 1 and 2, Comm. ACM 9, 1-2 p. 12, 99 (1966).
2) W. M. McKeeman: "An approach to computer language design", Tech. Rep., CS48, Computer Science Dept., Stanford U., Stanford, Calif., (Aug. 1966).
3) L. Presser: "The structure, specification, and evaluation of translators and translator writing systems" UCLA-10, P. 14—52, Rep. No. 68—51, USLA, Los Angeles, Calif. (Oct. 1968)