B4 右順位文法に対する構文解析プログラムの発生

井上謙蔵・阿部　克・西村公一（富士通）

1. 緒

コンパイラ作製自動化の一つの焦点は，コンパイラ・コンパイラである．コンパイラ・コンパイラは，言語の文法の形式化の上になりたっている．文法のうち，意味づけの記述法については，あまり多くの仕事はなされていない．構文の記述と，構文解析法については，多くの提案がなされ，実現されているが，その方法は，大まかに未確定的なものと，確定的なものに分類される．前者は構文の記述能力は高いが，解析速度がおそい．後者はその逆であるから，速度のはやいコンパイラを作るのに使われるが，それでも実用的観点よりは不満がある．それは解析に使用される表の大きさと，やはり速度の問題である．

本報告では，右順位文法にしたがう構文解析プログラムとその発生を説明する．これは順位文法の表現能力を広げ，かつ表を小型とし，解析速度をあげるものである．

2. 右順位文法

文脈独立文法を

$$G=(V_N,\ V_T,\ P,\ S) \qquad (2.1)$$

V_N ：超変数の集合

V_T ：端記号の集合

P ：生成規則の集合

S ：出発記号

とするとき，右順位文法をつぎのように定める．[1)]

1) 出発記号は，いかなる生成規則の右辺にもあらわれない．

2) 任意の超変数 A に対して，必らず

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

なる導出がある．ただし $V_T^{\,*}$ は，V_T の要素の任意の列（文）の集合である．

3) 任意の生成規則

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

において，$A \neq B$ で，$\alpha = \beta$ であることはない．ただし，

$$A, B \in V_N,\quad \alpha, \beta \in V^* - \{\lambda\},\quad V = V_N \cup V_T$$

で，V^* は V の要素の任意の列（文形）の集合，λ は空列である．

4) $S \overset{*}{\Rightarrow} \alpha$

である任意の文形 α 中に，となり合う2記号（超変数と端記号）

$A \in V, \quad a \in V_T$

の間には，一義的な順位関係がある．

5) 生成規則

$A \to \alpha\beta, \quad B \to \beta \qquad A, B \in V_N, \quad \alpha, \beta \in V^*$

が存在するときは生成規則

$C \to \gamma H_i(\alpha) D \delta$

は存在しない．ただし

$C \in V_N, \quad \gamma, \delta \in V^*$

$D = T_{n-i}(\alpha) B$ 又は $D \overset{*}{\Longrightarrow} T_{n-i}(\alpha) B$

で，$H_i(\alpha)$，$T_{n-i}(\alpha)$ は記号列 α の頭の i 桁，尾の $n-i$ 桁の記号であり，n は α の長さ $|\alpha|$ とする．

条件 4) における右順位関係は，つぎのように定義する．集合

$$L(X) = \{Y \mid X \overset{*}{\Rightarrow} Y\xi, \quad Y \in V, \quad \xi \in V^*\}$$
$$R(X) = \{Y \mid X \overset{*}{\Rightarrow} \xi Y, \quad Y \in V, \quad \xi \in V^*\}$$
$$L_T(X) = \{X \mid X \in V_T\} \cup \{Y \mid X \Rightarrow Y\xi, \quad Y \in V_T, \quad \xi \in V^*\} \qquad (2.2)$$

とするとき，任意の生成規則

$A \to \alpha BC \beta \qquad \alpha, \beta \in V^*$

において，

i) $B \leqq C$

ii) $F \in L(C)$ なら $B \leqq F$

iii) $E \in R(B)$ なら $E \geqq C$

iv) $E \in R(B)$, $F \in L(B)$ なら $E \geqq F$

である．条件 4) は，上の関係で

$E \in R(B), \quad F \in L_T(C)$

であるとき，関係

$B \lessdot F, \quad E \lessdot C$

が，ことごとくの記号対に関して一義的になりたつことを要求するものとする．

上に定義された文法が，曖昧でないことおよび解析方法が確定的であることについての証明は省略する．

右順位文法の条件 2) は，任意の文脈独立文法を，そのように書き直すことができるので，実質的な制限条件にはならない．条件 3) は，確定的構文解析法に従う多くの文法の類にとって共通な制限である．条件 4) と 5) は，右順位文法独特のものである，これらの条件をゆるめるか，条件にかなうように文法を書き換える方法については，のちに説明する．

条件 1) については，V_T 中に含まれない端記号 $\vdash$ および $\dashv$，超変数 S' を導入し，

$$G' = (V_T \cup \{\vdash, \dashv\}, \quad V_N \cup \{S'\}, \quad P \cup \{S' \to \vdash S \dashv\}, \quad S') \qquad (2.3)$$

をつくればよい．付加的な生成規則は，G' の中で特別に扱われるものとしよう．

文献1）と時を同じくして，殆んど同じ発想の下に弱順位文法が，他の著者によって定義された．[2] これは条件5）のかわりに

$$T_1(\alpha)\odot B$$

をおくものである．これが右順位文法よりはきびしい条件を与えていることは明らかであろう．実際に，次の文法は右順位文法であるが，弱順位文法ではない．

$$G=\{\{S_0, S, A, B\},\ \{a, b\},\ P,\ S_0\}$$
$$P=\{S_0 \to \vdash S \dashv,\ S \to A,\ S \to aA,\ A \to B,\ A \to AaB,\ B \to b\}$$

3. 順位表と構文表

集合Vの要素数をn，V_Tの要素数をn_Tとして，$n+1$行，n_T+1列のマトリックスMを作る．各行には，$V \cup \{\vdash\}$の要素の名前をつけ，各列には，$V_T \cup \{\dashv\}$の要素の名前をつける．ただし便宜上，最下端行を$\vdash$の行，最右端列を$\dashv$の行とする．マトリックスMの各要素は，行と列の番号によって$M_{i,j}$と記されるか，行と列の名前によって$M(N_A, N'_a)$と記されるものとしよう．

Mの要素には，つぎのような記入を行なう．

$$A \in V \cup \{\vdash\},\quad a \in V_T \cup \{\dashv\} \qquad (3.1)$$

に対して，

$$A \gtrdot a \text{ ならば} \quad M_{A,a} = \gtrdot,$$
$$A \doteqdot a \text{ ならば} \quad M_{A,a} = \doteqdot, \qquad (3.2)$$
$$A \gtrdot a \text{ でも } A \lessdot a \text{ でもなければ} \quad M_{A,a} = \odot$$

である．このマトリックスMを順位表という．

右順位文法であれば，その順位表の各要素には$\lessdot$，$\doteqdot$，又は$\odot$が一義的に記入される．

順位表の各要素は，3つの状態を区別しなければならないから，2ビットを必要とする．そこで，表全体では

$$2(n_T+1)(n+1) \quad \text{ビット} \qquad (3.3)$$

が必要である．しかし，実用上$\odot$と$\gtrdot$を区別しなくてもよいこと，したがって表の大きさを

$$(n_T+1)(n+1) \quad \text{ビット} \qquad (3.4)$$

にとどめることができる．

構文表は，生成規則を樹の形に並べた表であるが，完全に樹にしてしまうとメモリ効率が悪いし，構文解析の速度にも影響するので，ベクトル的に並べることにする．

まず

$$S' \to \vdash S \dashv \qquad (3.5)$$

を除く，すべての生成規則を，右辺の尾の1記号の同じものの部分集合にわける．その1組を，たとえば，

$$A \to \alpha_1 \alpha_2 \alpha_3 Q : A_S$$
$$B \to \alpha_3 Q : B_S \qquad (3.6)$$
$$C \to \alpha_1' \alpha_2 \alpha_3 Q : C_S$$

とする．ここで

$$A, B, C \in V_N, \quad Q \in V,$$
$$\alpha_1, \alpha_1', \alpha_2, \alpha_3 \in V^*$$

で，A_S，B_S，及びC_Sは，それぞれの規則に対応する意味づけ規則の名前とする．
これらの規則を，まとめてつぎのようにあらわす．

$$〔(C_S : C)\ \alpha_1' \,|\, (A_S : A)\ \alpha_1〕\ \alpha_2\ (B_S : B)\ \alpha_3 Q \qquad (3.7)$$

これを，右から左への順に構文表に記入する．表3.1に示されるように．構文表の各行はLH，HD，SYMBOL，およびBSの4欄よりなり，LHとHDには1（true）又は0（false）が入り，SYMBOLにはVの要素が入り，BSには意味づけ規則，構文表の行，又は順位表の行などの名前が入る．

まず構文表の（あいている部分の）第1行をL_Qと名付ける．その行のBS欄には順位表のQの行の名前N_Qを入れ，LHの欄は0，HDの欄は右辺がQだけの規則があれば1，なければ0とする．SYMBOL欄にはなにも入れない．

つぎの行から下へ順次SYMBOL欄に，T_1（α_3）から順次左へ並ぶ各記号を記入する．これらの各行のLHとHD欄は0とし，BS欄には$T(\alpha_3 Q)$を右辺とする規則があれば，その最長のものの左辺の行の名前，なければ誤りの行の名前L_ωが入る．

右の丸括弧に到達したとき，最後に記入された記号は，1つの規則の右辺の左端に位するから，その記号の行のHD欄を1とする．丸括弧の中は，左辺の超変数と意味づけ規則の名前であるから，それを左辺の行（LH＝1）のSYMBOLとBS欄に記入するが，この左辺の行は，対応する左丸括弧の左に｜か〔があれば，その規則の右辺に続く行に，そうでなければ，他の右辺の規則のうしろにおかれる．

α_2の記号については，α_3と同様である．

右の角括弧〕に達したときは，試すべき記号列が二手にわかれるから，次の記号T_1（α_1）の行のBS欄はT_1（α_1'）の行の名前を記入する．以後はα_1については，α_3，α_2と同様に記入される．α_1の表の部分につづいて，Aに関する左辺の行があり，それにBに関する左辺の行が続く．その下にα_1'の記入が続いて，最後にCに関する左辺の行が作られる．

各行のSYMBOL欄には，記号そのものでなく，その記号を右端にもつ生成規則の集合の，構文表における第1行の名前（例えばL_Q）が記入されるものとする．それがいかなる規則の右端にもあらわれなければ，その記号のために特別な1行（LH＝HD＝1，BS＝順位表におけるその記号の行の名）の名前が入る．

誤まりの行はBS欄のみが使用される．

表3.1　構文表の記入法

	LH	HD	SYMBOL	BS
L_Q :				N_Q
			$L_{T_1}(\alpha_3)$	L_ω
		1	$L_{H_1}(\alpha_3)$	L_ω
			$L_{T_1}(\alpha_2)$	L_{Q1}
			$L_{H_1}(\alpha_2)$	L_{Q1}
			$L_{T_1}(\alpha_1)$	L_1
		1	$L_{H_1}(\alpha_1)$	L_{Q1}
	1		L_A	A_S
L_{Q1} :	1		L_B	B_S
L_1 :			$L_{T_1}(\alpha_1')$	L_{Q1}
		1	$L_{H_1}(\alpha_1')$	L_{Q1}
	1		L_C	C_S

例3.1

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

$V_N=\{S, E, T, F\}$

$V_T=\{a, m, l, r, i, \vdash, \dashv\}$

$P=\{S\to\vdash E\dashv,\quad E\to EaT:R_1,\quad E\to T:R_2,\quad T\to TmF:R_3,$

$T\to F:R_4,\quad F\to i:R_5,\quad F\to lEr:R_6\}$

ただし，$R_1, R_2, \cdots, R_6$は意味づけの規則の名前とする．

例文法に対して，つぎの集合がつくられる．

$R(E)=\{T, F, i, r\}$,

$R(T)=\{F, i, r\}$,　　$R(F)=\{i, r\}$,

$L_T(S)=\{\vdash\}$,

$L_T(E)=L_T(T)=L_T(F)=\{i, l\}$,

$L_T(i)=\{i\}$,　$L_T(l)=\{l\}$,　$L_T(a)=\{a\}$,

$L_T(m)=\{m\}$,　$L_T(r)=\{r\}$

これらの集合と各生成規則から，表3.2(a)の順位表がつくられる．これは⊙を≫とし，重ね合せることにより表3.2(b)に変更される．

また生成規則を，その右端の記号によってわかれる部分集合ごとに，書き換えて，つぎの形

にする．

$(R_1: E)\ E\,a\,(R_2: E)\ T$

$(R_3: T)\ T\,m\,(R_4: T)\ F$

$(R_5: F)\ i$

$(R_6: F)\ l\,E\,r$

これらの規則から，表3.3の構文表がつくられる．

表3.2　例3.1の順位表

(a)

		a	m	i	l	r	$\dashv$
N_E:	E	⋖				⋖	⋖
N_T:	T		⋖			⋗	⋗
N_F:	F					⋗	⋗
N_a:	a			⋖	⋖		
N_m:	m			⋖	⋖		
N_i:	i					⋗	⋗
N_l:	l			⋖	⋖		
N_r:	r					⋗	⋗
$N_\vdash$:	$\vdash$			⋖	⋖		

注：記入のない欄は⊙とする．

(b)

		a	m	(i, l)	$(r, \dashv)$
N_E:	E	⋖	⋗	⋗	⋖
N_T:	T	⋗	⋖	⋗	⋗
N_F: N_i: N_r:	F, i, r	⋗	⋗	⋗	⋗
N_a: N_m: N_l: $N_\vdash$:	a, m, l, $\vdash$	⋗	⋗	⋖	⋗

表3.3　例3.1の構文表

	LH	HD	SYMBOL	BS
L_T:		1		N_T
			L_a	L_{T1}
		1	L_E	L_{T1}
	1		L_E	R_1
L_{T1}:	1		L_E	R_2

	LH	HD	SYMBOL	BS
L_r:				N_r
			L_E	L_ω
		1	L_l	L_ω
		1	L_F	R_6
L_E:	1	1		N_E

	LH	HD	SYMBOL	BS
L_F:		1		N_F
			L_m	L_{F1}
		1	L_T	L_{F1}
	1		L_T	R_3
L_{F1}:	1		L_T	R_4
L_i :		1		N_i
		1	L_F	R_5

	LH	HD	SYMBOL	BS
L_a :	1	1		N_a
L_m :	1	1		N_m
L_l :	1	1		N_l
$L_{\vdash}$:	1	1		$N_{\vdash}$
L_ω:				ω

4. 構文解析

解析中の文形を

$$S'_1 \cdots S'_{k'}\ S_k\ S_{k+1} \cdots S_n \qquad (4.1)$$

とする．ここで

$$S'_1,\ S'_2,\ \cdots,\ S'_{k'} \in V,\quad S_k,\ S_{k+1},\ \cdots,\ S_n \in V_T,$$

であり，$k=0$ であるときは（4.1）は与えられた文とする．これに

$$S' \rightarrow \vdash S \dashv$$

に対応して，

$$S'_0 = \vdash \text{ および } S_{n+1} = \dashv$$

をつけ加えて考える．また S_k，S_{k+1}，…，S_{n+1} の任意のものを X とするとき，それはそれによって名付けられた構文表の行の名前 L_X と順位表の列の名前 N'_X の対（L_X，N'_X）として与えられるものとする．

S'_0，S'_1，…，$S'_{k'}$ はスタックに保存され，$S'_{k'}$ がその尾であるが，任意の記号 X がスタックに入れられるときは，（L_X，N'_X）から，（L_X，N_X）に変換されるものとする． N_X の値は構文表の L_X 行の BS 欄に記入されているから，この変換は容易である．

以上の注意の下に，右順位文法に対する構文解析手順が，図 4.1 に示される．図中 ω は誤りの処理プログラムの名前で，それは表 3.3 中のものと同じである．

ω のために，スタック 1 要素の記入項目に 1 ビットからなる欄 ER を加えて，

（ER，L_X，N_X）

とする．$S'_{k'}$ の ER を $ER_{k'}$，S_k の ER を S_k とすれば，通常これらは 0 の値をもつ．これが 1 をもつのは，誤りが認識されたときである．その手順は図 4.2 に示される．この方法によってエラー・アバランシュを，ある程度防止できるであろう．

ω' は，ω とメッセージが異なるだけである．

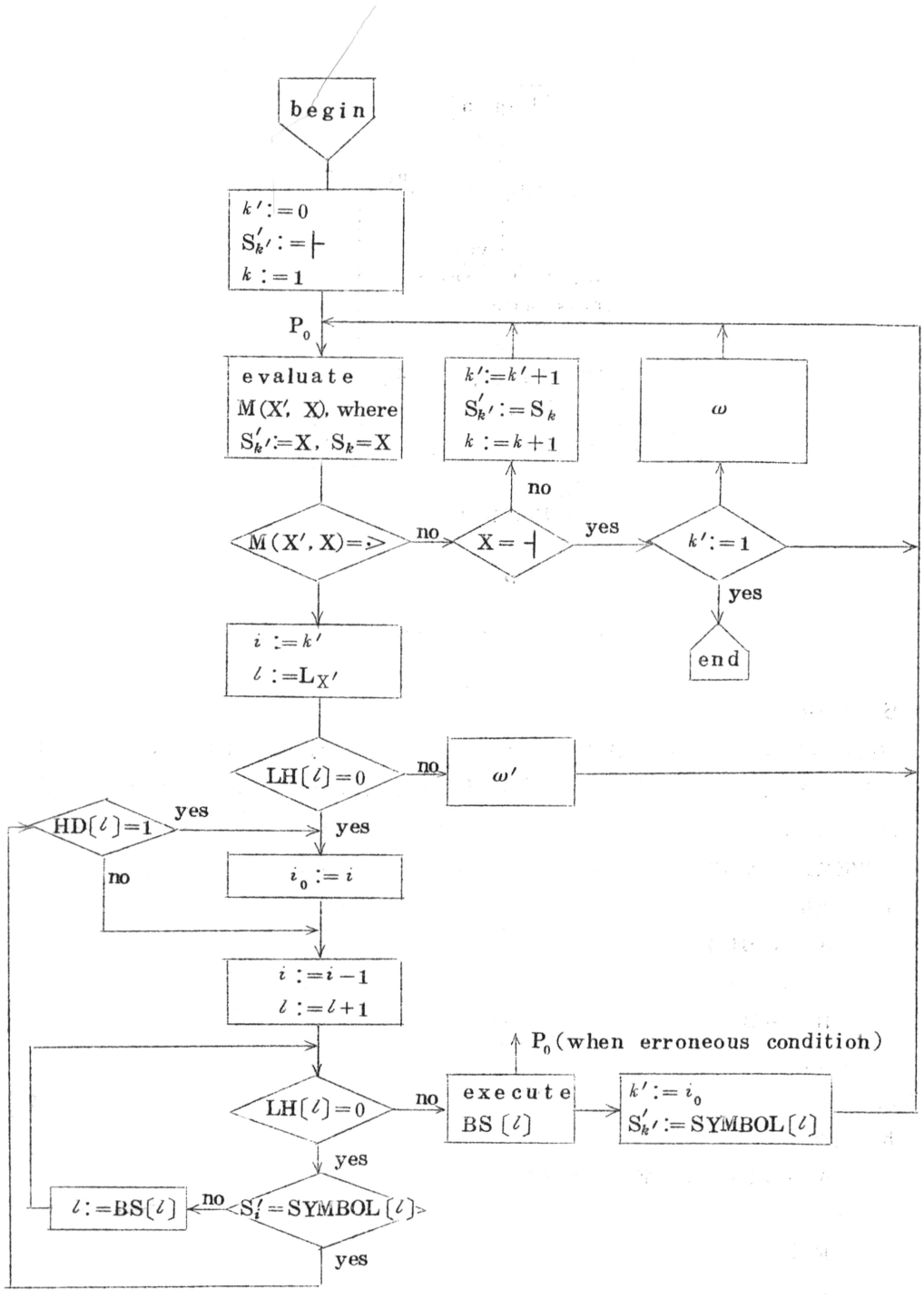

図 4.1　右順位文法の構文解析手順

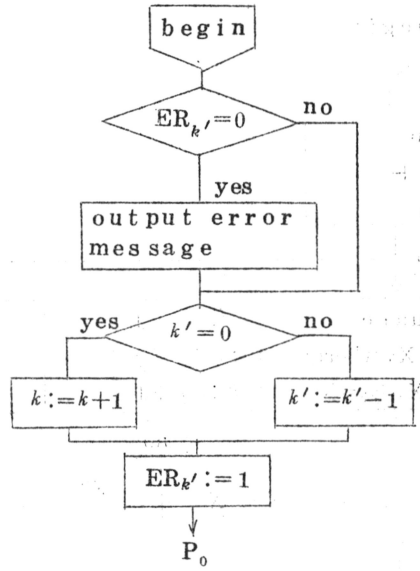

図 4.2　誤まりの処理

5. 拡張

右順位文法は，左順位が一義的でなくてもよいから，左回帰的な規則はすべて，その中に含むことができる．しかし右順位の一義性が破られる場合には，それを回避する手段を考えなければならない．

右順位文法の一義性が破られるのは，つぎのような場合である．

a) 生成規則

$$A \to \beta B C \gamma$$

があって，かつ

$$B \overset{*}{\Rightarrow} \delta B$$

である．

又は

b) 生成規則

$$A \to \beta B C \gamma, \quad E \to \mu F C \nu$$

があって，かつ

$$F \overset{*}{\Rightarrow} \delta B$$

である．

このような場合には，新らしい超変数 B′ を導入して，

$$A \to \beta B C \gamma$$

の代りに

$A \to \beta B'C\gamma$,　$B' \to B$

とすればよい．このようにしてもB′があらわれるのは，この2規則の中だけであるから他の記号間の順位関係には影響を与えないし，節2の条件5）にも影響を与えない．ただし，節2の条件3）には影響を与える可能性がある．そこで次に，節2，条件3）に対する拡張の問題を議論する．

まず

$$A_1 \to \alpha,\ \cdots,\ A_n \to \alpha \qquad (5.1)$$

でA_j($1 \leqq j \leqq n$)は相互に異なるものとしよう．このような規則の組は，つぎの2種類にわけられる．

まず最初は，αが名前（identifier）であるような場合である．名前はいろいろな文法上の概念をあらわすのに用いられるが，それを名前そのものから類別するのは，通常の文脈独立文法の範囲をこえている．たとえば，ある一つの名前が単純な変数をあらわすのか，ラベルや手続きの名前をあらわすのかは，以前になされた定義を参照しなければならない．

このような場合には，それらの名前についての宣言がなされたとき，名前の表に，その名前と性質を記入しておき，のちにその表を参照するのが普通である（スタック・オートマトン）．われわれも，勿論このような方法をとることにする．

ここで問題とするのは，上記の場合を除いた部分である．下向き（top down）型の直構文解析法（syntax directed）であれば，まず文全体に適応する規則についての大まかな予想をたて，ついでその条件の中で部分列について適用可能な規則の予想をたてる．このような手順を繰返すのであるから，（5.1）のいずれの規則を試みるかということは自動的に定まってくる．しかし，上向き（bottom up）型であれば，いわば大枠がきめられていないために，いずれの規則を適用すればよいかは自動的には定まらない．このような場合は，αの両側の文脈（context）を参照すればよい．

われわれの構文解析法は左より右へ進み，かつS_kは順位を比較するためにのみ用いられるのであるから，左側の文脈だけを参照することにする．そのようにすれば，構文解析法は図4.1のものを全然変更しなくて済ますことができるからである．

式（5.1）のA_1，A_2，…等を含む，その左の，最右端導出（rightmost derivation）に従って考えうることごとくの文脈の，お互いに区別しうる最短の長さのものをとり出して，

$$\begin{matrix} \alpha_1 & A_1 \\ \alpha_2 & A_2 \\ \vdots & \\ \alpha_{n'} & A_{n'} \end{matrix} \qquad (5.2)$$

とする．一般に1つの超変数に，複数個の文脈が考えられるから，n'はnより大である．また（5.2）ではA_iは，（5.1）の超変数の添字をつけ変えたもので

$$A_i = A_j \qquad i \neq j$$

であってもよい．このとき $\alpha_i \neq \alpha_j$ である．

簡単のため，一例をとってみる．

$$\alpha_1 A_1 \ , \ \alpha_3 \alpha_2 A_3 \ ,$$
$$\alpha_4 \alpha_2 A_4 \ , \qquad (5.3)$$
$$A_i \rightarrow \alpha : A_{Si} \quad (1 \leqq i \leqq 4)$$

とする．これは次のようにあらわされる．

$$[[(A_{S3}, A_3)\alpha_4 \mid (A_{S2}, A_2)\alpha_3]\alpha_2 \mid (A_{S1}, A_1)\alpha_1]\,\|\,\alpha \qquad (5.4)$$

縦棒 $\|$ は，これより左が文脈を示すことを意味する．このとき，構文表の $H_1(\alpha)$ の HD 欄には 1 が記入され，これに応じて，$H_1(\alpha_1)$，$H_1(\alpha_3)$，$H_1(\alpha_4)$ 等の行の HD 欄は 0 とされる．それ以外は，表 3.1 の構成法と全く同様である（表5.1参照）．

除外された，文脈独立文法の枠をこえるものに関しては，その意味づけ規則の中で，採用すべき構文の規則の選択がなされなければならない．これには特別な標準手続きが使用される．またこのような構文の規則は，あらかじめ印をつけておき，（5.4）にあらわれる諸規則と混同しないような工夫が必要である．

表5.1　文脈の構文表

	LH	HD	SYMBOL	
$L_{T_1}(\alpha)$:				$N_{T_1}(\alpha)$
		1	$L_{H_1}(\alpha)$	L_ω
			$L_{T_1}(\alpha_1)$	L_1
			$L_{H_1}(\alpha_1)$	L_ω
	1		L_{A_1}	A_{S1}
L_1 :			$L_{T_1}(\alpha_2)$	L_ω
			$L_{H_1}(\alpha_2)$	L_ω
			$L_{T_1}(\alpha_3)$	L_2
			$L_{H_1}(\alpha_3)$	L_ω
	1		L_{A_3}	A_{S3}
L_2 :			$L_{T_1}(\alpha_4)$	L_ω
			$L_{H_1}(\alpha_4)$	L_ω
	1		L_{A_4}	A_{S4}

6. 構文解析プログラムの発生

構文解析プログラムの発生プログラムへの入力，すなわちある言語の規則の表現形式としては，

$$A \to \alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_l, \qquad 1 \leqq l \tag{6.1}$$

のものが許される．ただし

$$\alpha_{l'} = \beta_{l'}(\gamma_{l'1} \mid \gamma_{l'2} \mid \cdots \mid \gamma_{l'm})\delta_{l'} \qquad 1 \leqq l' \leqq l, \quad 0 \leqq m, \quad \beta_{l'}, \delta_{l'} \in V \tag{6.2}$$

で $\gamma_{l'm'}$ $(0 \leqq m' \leqq m)$ の形としては，$\alpha_{l'}$ の形のものが回帰的に適用される．(6.1)に対してはそこにあらわれる右辺の数だけの意味づけの規則の名前を，左から右への順に対応させて，

$$A \to \alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_l : A_1, \ A_2, \ \cdots, \ A_n \tag{6.3}$$

とされる．文脈独立文法の範囲をこえて右辺を等しくする生成規則があれば，それに対応する意味づけ規則の名前の前に＊印をつける．

(6.3)の列を入力として，それより順位表と構文表を作り，それと図4.1のごとき解析プログラムを結合して出力とする発生プログラムは，順位表と構文表の作製のほかに，

節2の条件 2)

節2の条件 3)

節2の条件 5)

についてしらべなければならない．条件4)は順位表を作ることによってなされる．条件3)に違反するものがあれば，その文脈の記号列を作らねばならない．また構文表を経済的なものとするために

$$S \overset{*}{\Rightarrow} \alpha A \beta, \qquad \alpha, \beta \in V \tag{6.4}$$

でないような超変数についてもしらべなければならない．

紙数の都合上，その構造の詳説はさける．

7. あとがき

右順位文法による構文解析法は，順位文法にくらべ表が小さく，速度がやや早い．更に文法の記述範囲が広いので，文脈独立文法からの書き換えが少なくてすむのが特徴である．問題点としては，ある文法の節2の条件への適否をしらべる手順が，かなり時間を要するものであることであろう．しかし，これは発生プログラムの上での問題であるから実用上は問題でない．

ALGOL 60に対して表の大きさの上限を概算してみる．ALGOL 60では

超変数　約 110

端記号　約　56

ただし英字と数字は，それぞれ1個とみなす(語彙解析で細かに判別する)．

生成規則の数　約 213

ただし有効なもののみをとった．これらの右辺の長さの総和は，約373．

そこで順位表の大きさは，約

（110＋56）×56＝9296 ビット以下

である．通常順位表は，全く同じ構成の行又は列を含むから，これを重ね合せることにより更に小型とされる．この傾向は順位表の縮約によって助長される．構文表は，約40個の記号がどの生成規則の右端にもあらわれないから，これを追加して，

213＋373＋40＝728 行

である．これも尾の一致するものが数多くあるから，実際にそれらを考慮に入れると，かなり減少するであろう．

36 ビットを1語，構文表の1行を1語とすると，合計

986 語以下

となる．実際には2／3位であろう．

構文解析プログラム自身の大きさは

129 語

にすぎない．

順位文法では，ALGOL 60に対して，上限が，約

$2\times(110+56)^2=55112$ ビット

の順位表を必要とする．この表も同じ構成の行と列を重ね合せることができるが，その機会は右順位文法より少ない．構文表は右順位文法と同じであるから，この場合は全体で，上限が約

2256 語

の表を必要とする．

引用文献

1) 井上謙蔵，"右順位文法"，情報処理11，No.8，p.449（Aug.1970）．

2) J.D.Ichbiah and S.P.Morse，"A Technique for Generating Almost Optimal Floyd-Evans Productions for Precedence Grammars"，C.ACM 13，No.8，p.501（Aug.1970）．