Right Precedence Grammars

KENZO INOUE*

Abstract

The definition on precedence grammars 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 grammars, could cover wider languages and its parsing speed may be expected to be faster than the previous one.

1. *Introduction*

Although simple precedence grammars give the very practical analyzing method of languages in the speed and largeness of parsers, but their syntactical class is rather low to cover usual programming languages.

For example, consider the next syntax for simplified arithmetic expressions.

example 1:

$$\begin{aligned} &G=(V_N,\ V_T,\ P,\ S)\\ &V_N=\{S,\ E,\ T,\ F\}\\ &V_T=\{+,\ \times,\ (,\ ),\ i,\ \vdash,\ \dashv\}\\ &P=\{S\to\vdash E\dashv,\ E\to E+T,\ E\to T,\ T\to T\times F,\ T\to F,\ F\to i,\ F\to(E)\} \end{aligned} \tag{1}$$

The precedence relations for the above example are decided with several dualities using simple precedence rules (Fig. 1). Because these dualities are based on the left recursive productions which are frquently used in describing usual pro-

	E	T	F	i	$+$	$\times$	$($	$)$	$\dashv$
E					$\doteq$			$\doteq$	$\doteq$
T						$\doteq$		$\gtrdot$	$\gtrdot$
F								$\gtrdot$	$\gtrdot$
i					$\gtrdot$	$\gtrdot$		$\gtrdot$	$\gtrdot$
$+$		$\doteq$	$\lessdot$	$\lessdot$			$\lessdot$		
$\times$			$\doteq$	$\lessdot$			$\lessdot$		
$($	$\doteq$	$\lessdot$	$\lessdot$	$\lessdot$			$\lessdot$		
$)$					$\gtrdot$	$\gtrdot$		$\gtrdot$	$\gtrdot$
$\vdash$	$\doteq$	$\lessdot$	$\lessdot$	$\lessdot$			$\lessdot$		

Fig. 1. Simple precedence relations of Example 1.

This paper first appeared in Japanese in Joho Shori (the Journal of the Information Processing Society of Japan), Vol. 11, No. 8 (1970), pp. 449-456.

* Dept. of Information Processing Technology, Fujitsu Ltd.

gramming languages, we need wider classes of grammars giving practically reasonable parsing speed and largeness of parsers.

With the above purpose, an attempt to revise simple precedence grammars is shown in the following.

2. *Terminology and Symbolism*

The following terms and symbols are used in the meaning in the reference [1] and [2].

a) context-free grammar $G=(V_N, V_T, P, S)$, (2)
where V_N is a finite set of variables, V_T a finite set of terminal symbols (not including marginal symbols $\vdash$ and $\dashv$), P a finite set of productions, and S the start symbol of G.

b) $V=V_N \cup V_T$.

c) elements of V_N: $A, B, C, \ldots$
elements of V_T: $a, b, c, \ldots$

d) sentential forms: $\alpha, \beta, \gamma, \ldots$
sentences: $z, y, x, \ldots$
null string: Λ.
productions: $A \to \alpha, \ldots$

e) the set of all finite strings over V including Λ: V^*,
the set of all finite strings over V_T including Λ: V_T^*.

f) direct derivation or direct reduction: $\alpha \Rightarrow \beta$,
derivation or reduction: $\alpha \overset{*}{\Rightarrow} \beta$.

g) phrase, handle, canonical parse.

h) $L(A)=\{F \mid A \overset{*}{\Rightarrow} F\alpha,\ A \in V_N,\ F \in V,\ \alpha \in V^*\}$,
$R(A)=\{F \mid A \overset{*}{\Rightarrow} \alpha F,\ A \in V_N,\ F \in V,\ \alpha \in V^*\}$.

The next notations added.

i) $\alpha \Rrightarrow \beta$: $\alpha \Rightarrow \beta$ or $\alpha=\beta$,
$\alpha \overset{*}{\Rrightarrow} \beta$: $\alpha \overset{*}{\Rightarrow} \beta$ or $\alpha=\beta$.

j) $L'(A)=\{F \mid F \in L(A)$ or $F=A,\ A \in V\}$,

k) $H(\alpha)$: a head of a string α,
$H_i(\alpha)$: the head of length i of a string α,
$T(\alpha)$: a tail of a string α,
$T_i(\alpha)$: the tail of length i of a string α.

3. *Right Precedence Grammars and the Parsing Algorithm*

The right precedence relation between two symbols is determined as follows.

a) if $A \to \beta B E \gamma$ and $C \in L'(E)$ then $B \lessdot C$,

b) if $A \to \beta D E \gamma$ and $B \in R(D)$, $C \in L'(E)$ then $B \gtrdot C$,

where $\beta, \gamma \in V^*$.

If, for a context free grammar (2), the grammar

$$\begin{aligned} &G'=(V_N', V_T', P', S'),\\ &V_N'=V_N \cup \{S'\},\\ &V_T'=V_T \cup \{\vdash, \dashv\},\\ &P'=P \cup \{S' \to \vdash S \dashv\} \end{aligned} \tag{3}$$

obeys the following restrictions, we say G' is a simple right precedence grammar. That is:

1) for $A \in V_N'$, $A \overset{*}{\Rightarrow} u$ and $u \in V_T'^*$ must hold,
2) no productions such that $A \to \alpha$, $B \to \alpha$, $A \neq B$ exist,
3) between symbols A and b which are
   $S' \Rightarrow \alpha A b \beta$, $A \in V'$ $(= V_T' \cup V_N')$ and $b \in V_T'$,
   there holds a unique right precedence relation,

and

4) if there exist productions such that
   $A \to \alpha\beta$ and $B \to \beta$
   then $C \to \gamma H_i(\alpha) D \delta$ with $D \overset{*}{\Rightarrow} T_{n-i}(\alpha) B \delta'$ does not exist, where n is the length of α and $n \geqq i > 0$.

According to a) and b), for example 1, the precedence relations of symbols are decided as shown in Fig. 2. After this, let $M_{l,m}$ be the (Q_l, q_m)-element of precedence table, Q_1 the symbol naming the 1-th line and q_m the symbol naming the m-th column. The symbol $\odot$ stands for $\not\lessdot$ and $\not\gtrdot$.

	i	$+$	$\times$	$($	$)$	$\dashv$
E	$\odot$	$\lessdot$	$\odot$	$\odot$	$\lessdot$	$\lessdot$
T	$\odot$	$\odot$	$\lessdot$	$\odot$	$\gtrdot$	$\gtrdot$
F	$\odot$	$\odot$	$\odot$	$\odot$	$\gtrdot$	$\gtrdot$
i	$\odot$	$\gtrdot$	$\gtrdot$	$\odot$	$\gtrdot$	$\gtrdot$
$+$	$\lessdot$	$\odot$	$\odot$	$\lessdot$	$\odot$	$\odot$
$\times$	$\lessdot$	$\odot$	$\odot$	$\lessdot$	$\odot$	$\odot$
$($	$\lessdot$	$\odot$	$\odot$	$\lessdot$	$\odot$	$\odot$
$)$	$\odot$	$\gtrdot$	$\gtrdot$	$\odot$	$\gtrdot$	$\gtrdot$
$\vdash$	$\lessdot$	$\odot$	$\odot$	$\lessdot$	$\odot$	$\odot$

Fig. 2. Right precedence relations of Example 1.

Let a sentential form be

$$A_0 A_1 A_2 \cdots A_{k'} a_k a_{k+1} \cdots a_{n+1} \tag{4}$$

which is being analyzed by the following parsing procedures on the right precedence grammars (2), where

$A_0, A_1, \ldots, A_{k'} \in V'$,

$a_k, a_{k+1}, \ldots, a_{n+1} \in V_T'$,

and

$$A_0=\vdash,\ a_{n+1}=\dashv.$$

A syntax table which consists of productions of grammar (2) must be prepared.

Parsing procedures:

(p1) Let $k'=0$, $k=1$ and $A_0'=\vdash$.

(p2) Search for $M_{l,m}$ with $Q_l=A_{k'}$ and $q_m=a_k$.

If $M_{l,m}=\odot$ then the procedures terminate since an erroneous condition has occurred.

If $M_{l,m}=\leqq$ with $a_k=\dashv$, then the analysis has been completed and the procedures terminate.

If $M_{l,m}=\lessdot$ with $a_k\neq\dashv$ then go back to (p2) with

$k':=k'+1$, $A_{k'}:=a_k$ and $k:=k+1$.

If $M_{l,m}=\gtrdot$ then $j:=k'$ and proceed to (p3).

(p3) From the syntax table pick a production of the form

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

and, for $k'=1, 2, \ldots, b-1$, check to see if $C_{b-k'}$ and $A_{j-k'}$ coincide one by one.

If more than one matching production is found, for the largest b, we assume $i:=j-b+1$.

If no such production is found, then the procedures are terminated with an erroneous condition.

(p4) Since $A_i\ A_{i+1}\ldots A_j$ is the handle, add

$B\to C_1C_2\ldots C_b$

to the right of the canonical parse, compute

$k':=i$, $A_k:=B$

and go back to (p2).

Let there be t productions whose right hand coincide with the tails of $A_0\ldots A_j$,

$$B_s\to\alpha_s\alpha_{s-1}\ldots\alpha_1,\ 1\leqq s\leqq t,\ \alpha_s\in V^* \tag{5}$$

In the syntax table, the right hand of each production of (5) is collectively expressed by

$\alpha_t\alpha_{t-1}\ldots\alpha_1$.

Collation is made to $A_{j-k'}$ ($k'=1, 2, 3, \ldots$) leftwards starting with $T_1(\alpha_1)$, and if perfect coincidence is established up to $H_1(\alpha_1)$, the value of k' is recorded. Then the procedure continues on leftwards. Generally if perfect coincidence is established up to $H_1(\alpha_s)$, the value of k' is updated to reflect $H_1(\alpha_s)$ rather than $H_1(\alpha_{s-1})$.

If in α_{s+1} there is a symbol which does not have a perfect counterpart, then

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

is the longest phrase. If the correspondence is established up to α_t, then

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

is the longest phrase.

Productions whose right hand sides have the identical tails but not heads may be denoted by

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

in which the common part is shared. If expressed as above, it is easy to know which production is rejected by checking A_{j-n} with $n=|\beta|$.

Described above, the class of right precedence grammars, with

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

deals β as a phrase even when $T(\alpha)\beta$ coincides with $A_{j-k'}\ldots A_j$, $T(\alpha) \neq A$ and $T(\alpha) \neq \alpha$. However, this is not true for simple precedence grammars, because in order for β to be a phrase, $T_1(\alpha) < H_1(\beta)$ must hold, and in order for $T_1(\alpha)\beta$ to correspond with $A_{j-k'}\ldots A_j$, $T_1(\alpha) \doteq H_1(\beta)$ must hold, which is not consistent with the requirement for simple precedence grammars [4].

4. *Determinism*

For a simple right precedence grammar, if both productions

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

coincide with the tails of $A_0\ldots A_j$, it implies that $\alpha\beta$ is the handle.

To prove this statement, first of all, by applying $B \to \beta$, let us assume that a sentential form was reduced to

$$\ldots\alpha B a_k\ldots$$

If $T_1(\alpha) \gtrdot B$ then $T_1(\alpha) \gtrdot H_1(\beta)$, which, however, is not possible. Therefore there is no phrase at the left of B. Secondly,

$$B \lessdot a_k \text{ or } B \gtrdot a_k$$

holds. If $B \lessdot a_k$, the scan proceeds rightwards, and a phrase which does not contain B being formed on to the right of α, reduction may take place. Taking into consideration this possibility, let us denote the sentential form after the above reduction by

$$\ldots\gamma\alpha B\delta_0 u_0\ldots,\ \gamma \in V'^*,\ \delta_0 \overset{*}{\Rightarrow} u_0',\ u_0', u_0 \in V_T'^*$$

where γ and u_0' may be Λ. Then the reduction that involves B is taken place using productions of the form

$$C_1 \to T_{i_1}(\alpha)B\delta_0 \text{ or } C_1 \to \gamma\alpha B\delta_0.$$

The length of $T_{i_1}(\alpha)$ is arbitrary: it may be 0. However, $T_{i_1}(\alpha) \neq \alpha$. Since the latter form is prohibited by 4), the sentential form becomes

$$\ldots H_{n-i_1}(\alpha)C_1 u_0\ldots$$

using the former form. Since $T_1(H_{n-i_1}(\alpha)) \lessdot C_1$ holds certainly again, the reduction involving C_1 is taken place using productions of the form

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

Here since the latter form is prohibited by 4), the sentential form becomes

$$\ldots H_{n-i_1-i_2}(\alpha)C_2 u_1\ldots$$

using the former. Here $\delta_1 \overset{*}{\Rightarrow} u_1'$, $u_1'u_1=u_0$, $H_{n-i_1-i_2}(\alpha)\neq\Lambda$ and u_1' may equal to Λ. Let us assume that by repeated application of this, the sentential form becomes

$$\ldots H_{n-i_1\ldots-i_j}(\alpha)C_j\boldsymbol{u_j}\ldots,\ H_{n-i_1\ldots-i_j}(\alpha)\neq\Lambda$$

using

$$C_j\rightarrow T_{ij}(H_{n-i_1\ldots-i_{j11}}(\alpha))C_{j-1}\delta_{j-1}.$$

According to the reduction procedures,

$$C_j \overset{*}{\Rightarrow} T_{i_1+i_2+\ldots+i_j}(\alpha)B_{u_0'u_1'\ldots u_j'-1}$$

and, hence, the head of α cannot be exhausted by reduction because of 4). That is, we were wrong in assuming that β was a phrase. Since the only other available production is $A\rightarrow\alpha\beta$, $\alpha\beta$ is the phrase. Moreover, $\alpha\beta$ is the handle because there is no other phrase at the left of $\alpha\beta$.

5. *Conclusion abstracted*

The next conclusion is reached from the discussion in applicability of simple precedence grammars.

(a) the restriction 4) offers little problem, as far as practical programming languages are concerned.

(b) for the dualities of right precedence relations, presser's solution [3] is applied.

(c) the restriction 2) are lighten by the reference to the left context of handles and the reference does not disturb the parsing algorithm described. This is a distinguished characteristic of simple precedence grammars.

A method of error detection and contraction of precedence matrices were discussed.

References

[1] Wirth, N. and H. Weber : Euler-a generalization of ALGOL and its formal definition Pt. 1 and 2, *Comm. ACM* 9, 2-3 (1966), 12-23 and 89-99.

[2] Mckeeman, W. M.: An approach to computer language design, Tech. Rep. CS 48, Computer Science Dept., Stanford U., Stanford, Calif. (Aug. 1966).

[3] Presser, L.: The structure, specification and evaluation of translators and translator writing Systems, UCLA-10P14-52, Rep. No. 98-51, UCLA, Calif. (Oct. 1968).