

LL(1) Parsing

Compiler Construction, Lecture 8

K.R. Chowdhary, Professor of CS

August 16, 2019 | Fall 2019

FIRST and FOLLOW

FIRST(α)

Terminals that begin strings derived from α ; $\varepsilon \in \text{FIRST}(\alpha)$ if $\alpha \Rightarrow^* \varepsilon$.

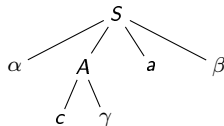
1. X terminal: $\text{FIRST}(X) = \{X\}$
2. $X \rightarrow Y_1 \cdots Y_k$: add $\text{FIRST}(Y_i) \setminus \{\varepsilon\}$ while $Y_1..Y_{i-1} \Rightarrow^* \varepsilon$; add ε if all Y_j derive ε
3. $X \rightarrow \varepsilon$: add ε

- Rules are applied until no set changes
- FIRST chooses the production from the next input symbol
- FOLLOW supplies ε -productions and synchronizing tokens in panic-mode recovery

FOLLOW(A)

Terminals that can appear immediately right of A in some sentential form; $\$$ if A can be last.

1. $\$$ is in $\text{FOLLOW}(S)$ for the start symbol S
2. $A \rightarrow \alpha B \beta$: $\text{FIRST}(\beta) \setminus \{\varepsilon\} \subseteq \text{FOLLOW}(B)$
3. $A \rightarrow \alpha B$, or $A \rightarrow \alpha B \beta$ with $\varepsilon \in \text{FIRST}(\beta)$: $\text{FOLLOW}(A) \subseteq \text{FOLLOW}(B)$



$c \in \text{FIRST}(A)$, $a \in \text{FOLLOW}(A)$

Example 8.1 and the LL(1) Condition

Grammar

$$\begin{aligned} E &\rightarrow T E' \\ E' &\rightarrow + T E' \mid \varepsilon \\ T &\rightarrow F T' \\ T' &\rightarrow \times F T' \mid \varepsilon \\ F &\rightarrow (E) \mid id \end{aligned}$$

	FIRST	FOLLOW
E	$\{(, id\}$	$\{), \$\}$
E'	$\{+, \varepsilon\}$	$\{), \$\}$
T	$\{(, id\}$	$\{+,), \$\}$
T'	$\{\times, \varepsilon\}$	$\{+,), \$\}$
F	$\{(, id\}$	$\{\times, +,), \$\}$

$\text{FIRST}(F) = \text{FIRST}(T) = \text{FIRST}(E)$, since F does not derive ε . E' appears only at the ends of E -bodies, so $\text{FOLLOW}(E') = \text{FOLLOW}(E)$.

G is LL(1) iff for every $A \rightarrow \alpha \mid \beta$

1. No terminal a starts strings derived from both α and β
2. At most one of α, β derives ε
3. If $\beta \Rightarrow^* \varepsilon$, $\text{FIRST}(\alpha) \cap \text{FOLLOW}(A) = \emptyset$ (and symmetrically)

Keyword-led constructs such as **if**, **while** and **{** satisfy this naturally.

Predictive Parsing Table $M[A, a]$

Construction

For each production $A \rightarrow \alpha$:

1. For each terminal $a \in \text{FIRST}(\alpha)$, put $A \rightarrow \alpha$ in $M[A, a]$
2. If $\epsilon \in \text{FIRST}(\alpha)$, put it in $M[A, b]$ for each $b \in \text{FOLLOW}(A)$, including $\$$

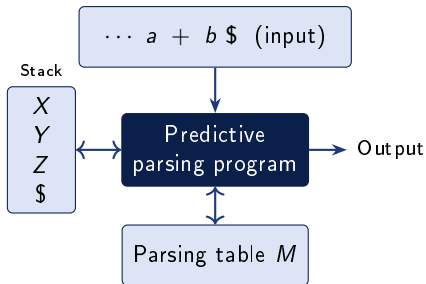
Empty entries are **errors**.

Example: $E' \rightarrow \epsilon$ goes in $M[E',)]$ and $M[E', \$]$ because $\text{FOLLOW}(E') = \{), \$\}$.

	id	$+$	$\times$	$($	$)$	$\$$
E	TE'			TE'		
E'		$+TE'$			ϵ	ϵ
T	FT'			FT'		
T'		ϵ	$\times FT'$		ϵ	ϵ
F	id			(E)		

Each entry shows the body of the production $A \rightarrow \cdot$ for the row's non-terminal.

Table-Driven Predictive Parser



The stack holds grammar symbols with the leftmost on top; if w has been matched, $S \Rightarrow^* w\alpha$.

Let X = top of stack, a = current input

- $X = a$ (terminal): pop, advance input
- X terminal, $X \neq a$: **error**
- $M[X, a]$ empty: **error**
- $M[X, a] = X \rightarrow Y_1 \dots Y_k$: output it, pop X , push $Y_k \dots Y_1$ with Y_1 on top

Stop when $X = \$$.

Example 8.3: Parsing $id + id \times id$

	Stack	Input	Action
1	$E\$$	$id + id \times id\$$	$E \rightarrow TE'$
2	$TE'\$$	$id + id \times id\$$	$T \rightarrow FT'$
3	$FT'E'\$$	$id + id \times id\$$	$F \rightarrow id$
4	$id T'E'\$$	$id + id \times id\$$	match id
5	$T'E'\$$	$+ id \times id\$$	$T' \rightarrow \epsilon$
6	$E'\$$	$+ id \times id\$$	$E' \rightarrow +TE'$
7	$+TE'\$$	$+ id \times id\$$	match $+$
8	$TE'\$$	$id \times id\$$	$T \rightarrow FT'$
9	$FT'E'\$$	$id \times id\$$	$F \rightarrow id$

	Stack	Input	Action
10	$id T'E'\$$	$id \times id\$$	match id
11	$T'E'\$$	$\times id\$$	$T' \rightarrow \times FT'$
12	$\times FT'E'\$$	$\times id\$$	match $\times$
13	$FT'E'\$$	$id\$$	$F \rightarrow id$
14	$id T'E'\$$	$id\$$	match id
15	$T'E'\$$	$\$$	$T' \rightarrow \epsilon$
16	$E'\$$	$\$$	$E' \rightarrow \epsilon$
17	$\$$	$\$$	accept

The outputs trace a leftmost derivation:
 $E \Rightarrow TE' \Rightarrow FT'E' \Rightarrow \dots$

Detection

A terminal on the stack top does not match the input, or $M[A, a]$ is empty.

Panic mode: skip to synchronizing tokens

- Put FOLLOW(A) in A 's synchronizing set; skip to one and pop A
- Adding FIRST(A) lets parsing resume with A
- A nullable non-terminal can default to its ϵ -production
- Blank entry: skip input a ; *synch* entry: pop the non-terminal; unmatched terminal on stack: pop it

Phrase level

Exercises

1. Predictive table for $S \rightarrow aSa \mid bSb \mid \epsilon$ and $S \rightarrow aSb \mid \epsilon$
2. Table for $S \rightarrow AS \mid a$, $A \rightarrow aA \mid b$; parse *aaaba*
3. Table for $S \rightarrow aBS$, $B \rightarrow ABb \mid \epsilon$, $A \rightarrow aA \mid \epsilon$
4. Error detection vs. recovery?
5. Case study of GCC, Turbo C, Fortran 90 and GPROLOG parsers