

Lecture 18

Computing FIRST Sets

Here is the algorithm for computing the FIRST sets.

1. For all terminal symbols b ,

$$\text{FIRST}(b) = \{b\}$$

2. For all productions: $X \rightarrow A_1 \dots A_n$

Add $\text{FIRST}(A_1) - \{\epsilon\}$ to $\text{FIRST}(X)$, stop if $\epsilon \in \text{FIRST}(A_1)$

Add $\text{FIRST}(A_2) - \{\epsilon\}$ to $\text{FIRST}(X)$, stop if $\epsilon \in \text{FIRST}(A_2)$

.....

Add $\text{FIRST}(A_n) - \{\epsilon\}$ to $\text{FIRST}(X)$, stop if $\epsilon \in \text{FIRST}(A_n)$

Add ϵ to $\text{FIRST}(X)$

This strategy is encoded in the following procedure

for each $a \in (T \cup \epsilon)$

$\text{FIRST}(a) \leftarrow \{a\}$

for each $A \in NT$

$\text{FIRST}(A) \leftarrow \emptyset$

while (FIRST sets are still changing)

for each $A \rightarrow b_1 b_2 \dots b_k \in P$

$\text{FIRST}(A) \leftarrow \text{FIRST}(A) \cup (\text{FIRST}(b_1) - \{\epsilon\})$

$i \leftarrow 1$

while ($\epsilon \in \text{FIRST}(b_i)$ and $i = k-1$)

$\text{FIRST}(A) \leftarrow \text{FIRST}(A) \cup (\text{FIRST}(b_{i+1}) - \{\epsilon\})$

$i \leftarrow i+1$

if ($i == k$ and $\epsilon \in \text{FIRST}(b_k)$)

$\text{FIRST}(A) \leftarrow \text{FIRST}(A) \cup \{\epsilon\}$

Example: consider the expression grammar again

1	E	$\rightarrow$	$T E'$
2	E'	$\rightarrow$	$+ T E'$
3		$ $	e
4	T	$\rightarrow$	$F T'$
5	T'	$\rightarrow$	$* F T'$
6		$ $	e
7	F	$\rightarrow$	(E)
8		$ $	<u>Id</u>

$$\text{FIRST}(\underline{\text{id}}) = \{ \underline{\text{id}} \}$$

$$\text{FIRST}('(') = \{ (\}$$

$$\text{FIRST}('+') = \{ + \}$$

$$\text{FIRST}(E) = \{ \text{FIRST}(T) - \{e\} \}$$

$$\text{FIRST}(T) = \{ \text{FIRST}(F) - \{e\} \}$$

$$\text{FIRST}(F) = \{ \text{FIRST}('(') - \{e\} \} = \{ (\}$$

$$\text{FIRST}(F) = \{ '(' \} + \{ \text{FIRST}(\underline{\text{id}}) - \{e\} \} = \{ (, \underline{\text{id}} \}$$

$$\text{FIRST}(E') = \{ +, e \}$$

$$\text{FIRST}(T') = \{ *, e \}$$

Thus,

$$\text{FIRST}(E) = \text{FIRST}(T) = \text{FIRST}(F) = \{ (, \underline{\text{id}} \}$$

$$\text{FIRST}(E') = \{ +, e \}$$

$$\text{FIRST}(T') = \{ *, e \}$$

FOLLOW Sets

Definition:

$$\text{FOLLOW}(X) = \{ b \mid S \Rightarrow^* b X b_w \}$$

Computing FOLLOW Sets:

1. Add \$ to FOLLOW(S) where S is the start non-terminal.
2. If there is a production $A \Rightarrow a B b$, then everything in $\text{FIRST}(\beta) - \{\epsilon\}$ is in FOLLOW(B).
3. If there is a production $A \Rightarrow a B$, or $A \Rightarrow a B b$, where $\epsilon \in \text{FIRST}(\beta)$ (i.e., $\beta \Rightarrow^* \epsilon$), then everything in FOLLOW(A) is in FOLLOW(B)

The following procedure encodes this strategy.

```

for each  $A \in NT$ 
    FOLLOW(A)  $\leftarrow \emptyset$ 
FOLLOW(S)  $\leftarrow \{\$ \}$ 
while ( FOLLOW sets are still changing)
    for each  $A \rightarrow b_1 b_2 \dots b_k \in P$ 
        FOLLOW( $b_k$ )  $\leftarrow$  FOLLOW( $b_k$ )  $\cup$  FOLLOW(A)
        T  $\leftarrow$  FOLLOW(A)
        for  $i \leftarrow k$  downto 2
            if ( $\epsilon \in \text{FIRST}(b_i)$ )
                FOLLOW( $b_{i-1}$ )  $\leftarrow$  FOLLOW( $b_{i-1}$ )  $\cup$  (FIRST( $b_i$ ) -  $\{\epsilon\}$ )  $\cup$  T
            else
                FOLLOW( $b_{i-1}$ )  $\leftarrow$  FOLLOW( $b_{i-1}$ )  $\cup$  FIRST( $b_i$ )
        T  $\leftarrow \emptyset$ 

```

Let's apply the algorithm to the expression grammar.

Put \$ in FOLLOW(E). By rule (2) applied to production $E \rightarrow (E)$, ')' is also in FOLLOW(E). Thus, FOLLOW(E) = {), \$ }. By rule (3) applied to production $E \rightarrow T E'$, \$ and ')' are in FOLLOW(E'). Thus, FOLLOW(E) = FOLLOW(E') = {), \$ }. Similarly, FOLLOW(T) = FOLLOW(T') = { +,), \$ } and FOLLOW(F) = { +, *,), \$ }