13 FIRST and FOLLOW Sets

Monday, October 16, 2023 11:04 AM

Still with top Down parsers.

Issue: Guiding the algorithm.

$A \to B \mid C \mid d$

which one to choose?

- if you know what terminals a B can begin with and what terminals a C can begin with.

then compare these two sets with the token to make a decision.

- of course you hope both dont begin with same terminal.

- **FIRST Sets:**

A set of terminals a non-terminal can begin with

Intuition.

$FIRST(\alpha) = \{$ terminals that begin sentences that can be derived from $\alpha \}$

$$FIRST(\alpha) = \{ a \mid \alpha \overset{*}{\Rightarrow} a\beta \}$$

Example:

$S \to AB \mid BA$
$A \to aB \mid cS$
$B \to bc$

$FIRST(a) = \{a\}$
$FIRST(c) = \{c\}$
$\vdots$
$FIRST(B) = \{b\}$ $\quad B \to \underline{b}c$
$FIRST(A) = \{a, c\}$
$A \to \underline{a}B \quad A \to \underline{c}S$
$FIRST(S) = \{a, b, c\}$
$S \to AB \to \underline{a}BB$
$S \to BA \to \underline{bc}A \quad S \to AB \to \underline{c}SB$

Algorithm:

1) • If X is a terminal then $FIRST(X) = \{X\}$
   else Repeat:
2) • If $X \to \lambda$ THEN add λ to $FIRST(X)$
3) • If $X \to Y_1 Y_2 Y_3 \ldots Y_k$
   a) ◦ If for some i, $\lambda \in FIRST(Y_j)$, for every $j < i$

5) • If $X \to Y_1 Y_2 Y_3 \ldots Y_k$

a) ○ If for some i, $\lambda \in FIRST(Y_j)$, for every $j < i$ and $a \in FIRST(Y_i)$ then add a to $FIRST(X)$

b) ○ If for all $i, 1 \le i \le k$, $\lambda \in FIRST(Y_i)$ then add λ to $FIRST(X)$.

$S \to ABCD$
$A \to \lambda \mid a$
$B \to \lambda \mid b$
$C \to \lambda \mid c$
$D \to \lambda \mid d.$

$FIRST(S) \to \{ c, \lambda,$

$A \to \lambda$
$B \to \lambda$

$S \to ABCD \to CD \to cD$
1 2 3 4
$i = 3$ $Y_1 \to \lambda$ $Y_2 \to \lambda$

$S \to ABCD \to \lambda$

	FIRST
a b c d	a b c d
S	a, b, c, d, λ
A	a, λ
B	b, λ
C	c, λ
D	d, λ

$D \to$

EXAMPLE #2

$A \to CB$
$B \to +CB \mid \lambda$
$C \to ED$
$D \to *ED \mid \lambda$
$E \to (A) \mid n$

	FIRST
A	(, n
B	+, λ
C	(, n
D	*, λ
E	n, (

USES:-

To guide the algorithm

$A \to B \mid C \mid d$

```
FUNCTION Parse_A()
    IF token ∈ FIRST(B) THEN
        parse_B()
    ELSIF token ∈ FIRST(C) THEN
        parse_C()
    ELSIF token = 'd' THEN
        get_token()
    ≡
```

Of course you strive for "pairwise disjointness"

$FIRST(B) \cap FIRST(C) = \emptyset$

- FOLLOW Sets

Intuition: the terminals that can appear immediately to the right of a non-terminal.

- FOLLOW sets

Intuition: the terminals that can appear immediately to the right of a non-terminal.

e.g. $S \Rightarrow^* ABC \rightarrow AbcC$

$b \in FOLLOW(A)$.

Definition: $FOLLOW(A) = \{a \mid S \Rightarrow^* A\beta \Rightarrow^* Aa\Gamma\}$

Note: $\$ \in FOLLOW(S)$ where S is the start symbol.

EXAMPLE:

$S \rightarrow AB \mid BA$
$A \rightarrow aB \mid cS$
$B \rightarrow bc$

$FOLLOW(S) = \{\$, b\}$
$FOLLOW(A) = \{b, \$\}$
$FOLLOW(B) = \{a, \$, c\}$

$S \rightarrow AB \rightarrow Abc$

$S \rightarrow BA \rightarrow BaB$ $\quad S \rightarrow BA \rightarrow BcS$

$S \rightarrow BA\$$ $\quad S \rightarrow AB \rightarrow cSB \rightarrow cSbc$

$S \rightarrow AB\$$

Algorithm:

1) Place \$ in FOLLOW(S), where S is the start symbol
REPEAT:
2) IF $A \rightarrow \alpha B \beta$ THEN place everything in $FIRST(\beta)$, except λ, into $FOLLOW(B)$
3) IF a) $A \rightarrow \alpha B$ or b) ($A \rightarrow \alpha B\Gamma$ and $\lambda \in FIRST(\Gamma)$) place everything in $FOLLOW(A)$ into $FOLLOW(B)$

Note: λ is never in a follow set.

3) $A \rightarrow \alpha B$ $\quad S \Rightarrow^* Ax \rightarrow \alpha Bx$

$A \rightarrow \alpha B\Gamma$ $\quad S \Rightarrow^* Ax \rightarrow \alpha B\Gamma x \Rightarrow^* \alpha Bx$

$\Gamma \Rightarrow^* \lambda$

Example #2

$A \rightarrow CB$
$B \rightarrow +CB \mid \lambda$
$C \rightarrow ED$
$D \rightarrow *ED \mid \lambda$

	FIRST	FOLLOW
A	(n	\$)
B	+ λ	\$)
C	(n	+ \$)
D	* λ	+ \$)

$Ax \rightarrow CB \rightarrow Cx$
V
λ

B → + C B | ε
C → E D
D → * E D | ε
E → (A) | n

	+	ε	$)
B			
C	(	n	+ $)
D	*	ε	+ $)
E	n	(	* + $)

ε

A → C B → C → E D → E → (A) → (C B) → (C) → (E D) → (E)

USES:

- Error Recovery on a recursive descent parser.

Intuition:

Suppose you are parse_A()

A → a ...

FIRST(A) = {a, b, c}

FOLLOW(A) = {x, y, z}

Algorithm Draft:

token = 🐱 🐱

- if token is not in the FIRST set
  - print error
  - read tokens until you find one in the FIRST set.
- parse_A as usual.
- if token is not in the FOLLOW set
  - print error
  - read tokens until you find one in the FOLLOW set

Example:

S → a B b

```
FUNCTION old_parse_S()
  IF token = 'a' THEN
    getToken()
    new_parse_B()
    IF token = 'b' THEN
      getToken()
```

```
FUNCTION new_parse_S()
  WHILE token not in FIRST(S) DO
    error_print(token)
    getToken()

  old_parse_S()

  WHILE token not in FOLLOW(S) DO
    error_print(token)
    getToken()
```

```
FUNCTION new_parse_S()
  WHILE token not in FIRST(S) ∪ FOLLOW(S) DO
    error_print(token)
    getToken()

  IF token in FIRST(S) THEN
    old_parse_S()
```

```
    WHILE token not in FOLLOW(S) DO
      error_print(token)
      getToken()

  ELSE
      error("expecting S")
```