← • HOME • UP • →

Top/Top-down parsing/FIRST(x)

8.2. FIRST(α) (Dragon Book p. 220)

On the previous page, I constructed the parsing table by hand. But we need an algorithm that constructs it. That is the subject of this and the next page.

Fix a grammar *G* to talk about. The tokens, nonterminals and productions in what follows belong to *G*. *S* is the start symbol.

Definition of FIRST(*X*)

Let α be a string of tokens and nonterminals.

Definition. FIRST(α) is the set of all tokens *t* so that α ⇒* *t*β for some string β.

Additionally, FIRST(α) contains ε if α ⇒* ε.

That is, FIRST(α) contains all tokens that can begin α, plus ε if α can be erased.

For example, let's see what FIRST(*N*) is for each nonterminal *N* in the following grammar.

$E \to T\,R$
$R \to \varepsilon$
$R \to +\,E$
$T \to F\,S$
$S \to \varepsilon$
$S \to *\,T$
$F \to \mathbf{n}$
$F \to (\,E\,)$

E -> FSR
-> nSR
-> (E)SR
T -> FS
-> nS
-> (E)S
F -> n
-> (E)
R -> ε
-> +
S -> ε
-> * T

N	*E*	*T*	*F*	*R*	*S*
FIRST(*N*)	{**n**, (}	{**n**, (}	{**n**, (}	{ε, +}	{ε, *}

Computing FIRST(*S*)

There is a simple and natural algorithm to compute FIRST(α). To begin with, we compute FIRST(α) where α is exactly one symbol long.

For each nonterminal *N*, start with FIRST(*N*) = {} and add members as necessary until no more members can be added.

To compute first(*t*) and first (*N*)

1. If *t* is a token then FIRST(*t*) = {*t*}.
2. If *N* is a nonterminal, find all productions with *N* on the left-hand side. For each such production $N \longrightarrow Y_1 Y_2 \ldots Y_n$ ($n \geq 0$), do the following.
    a. Add all tokens in FIRST(Y_1) to FIRST(*N*).

6. $U \longrightarrow$ `if` C `then` S
7. $U \longrightarrow$ `if` C `then` B `else` U
8. $C \longrightarrow$ $\mathtt{c}_i, i \geq 1$
9. $O \longrightarrow$ $\mathtt{s}_i, i \geq 1$

else B

RST(*N*).

nals: `if`, `then`, `else`, $\mathtt{c}_i$, $\mathtt{s}_i$, `$$`.

id nonterminals α.

To compute FIRST(α) for strings α

a. FIRST(ε) = {ε}

b. If α begins with token *t* then FIRST(α) = {*t*}.

c. If α = *N*β starts with nonterminal *N*, then

a. If FIRST(*N*) does not contain ε then FIRST(α) = FIRST(*N*).

b. If FIRST(*N*) contains ε then FIRST(α) = FIRST(*N*) ∪ FIRST(β).

The interesting part of the computation is FIRST(*N*) for a nonterminal *N*. Here is the computation for the expression grammar above.

An example

The computation goes in phases. Start with FIRST(*N*) = {} for every nonterminal *N*. For the sample grammar above, that looks like this.

N	*E*	*T*	*F*	*R*	*S*
FIRST(*N*)	{}	{}	{}	{}	{}

First, the productions whose right-hand side begin with a token and the erasing productions come into play. Rule 2 above tell you that

- FIRST(*R*) contains + and ε, due to productions *R* → + *E* and *R* → ε.
- FIRST(*S*) contains * and ε, due to productions *S* → * *T* and *S* → ε.
- FIRST(*F*) contains **n** and **(** due to productions *F* → **n** and *F* → **(** *E* **)**.

N	*E*	*T*	*F*	*R*	*S*

FIRST(*N*)	{}	{}	{**n**, (}	{ε, +}	{ε, *}

Now, since $T \rightarrow F\ S$ is a production, all tokens in FIRST(F) are added to FIRST(T).

N	*E*	*T*	*F*	*R*	*S*
FIRST(*N*)	{}	{**n**, (}	{**n**, (}	{ε, +}	{ε, *}

Then, since $E \rightarrow T\ R$ is a production, all tokens in FIRST(T) are added to FIRST(E).

N	*E*	*T*	*F*	*R*	*S*
FIRST(*N*)	{**n**, (}	{**n**, (}	{**n**, (}	{ε, +}	{ε, *}

Finally, no new members can be added to any of the sets, so the algorithm is done.

Using FIRST(*S*) in constructing the LL(1) parsing table.

The FIRST sets tell how to make entries in the LL(1) parsing table. Look at each production. If a production is

$$i.\ N \rightarrow \alpha$$

then add production number i to the table at row N, column t for every token t in FIRST(α).

← • HOME • UP • →

Top/Top-down parsing/FIRST(x)