Predictive Parser

Dr. Meera Thapar Khanna

CSE Department

Pandit Deendayal Energy University

Step for constructing Predictive Parser

- Write the Context Free grammar for given input String
- Check for Ambiguity. If ambiguous remove ambiguity from the grammar
- Check for Left Recursion. Remove left recursion if it exists.
- Check For Left Factoring. Perform left factoring if it contains common prefixes in more than one alternates.
- Compute FIRST and FOLLOW sets
- Construct Parser Table
- Using Parsing Algorithm generate Parse tree as the Output

First and Follow

To construct predictive parser following two functions are used:

- FIRST and FOLLOW, associated with a grammar G.
- During top down parsing, FIRST and FOLLOW allow us to choose which production to apply, based on the next input (look a head) symbol.

Computation of First

FIRST function computes the set of terminal symbols with which the right hand side of the productions begin.

To compute FIRST (A) for all grammar symbols, the following rules are used until no more terminals or ε can be added to any FIRST set.

1. If A is a terminal, then FIRST {A} = {A}.

2. If A is a Non terminal and A->X1X2…Xi

    FIRST(A)=FIRST(X1) if X1 is not null,

    if X1 is a non terminal and X1-> ε, add FIRST(X2) to FIRST(A),

    if X2-> ε add FIRST(X3) to FIRST(A), …

    if Xi-> ε, i.e., all Xi's for i=1..n are null, add ε FIRST(A).

3. If A -> ε is a production, then add ε to FIRST (A).

Example

Compute the FIRST values of the grammar

1. E -> TE’
2. E’ -> +TE’ | ε
3. T -> FT’
4. T’ -> *FT’ | ε
5. F -> (E) | id

FIRST Values

FIRST (E) = FIRST (T) = FIRST (F) = {(, id}

FIRST (E’) = {+, ε}

FIRST (T’) = {*, ε}

Computation of Follow

Follow (A) is the set of terminal symbols of the grammar that are immediately following the Non terminal A.

If a is to the immediate right of non terminal A, then

Follow(A)= {a}

To compute FOLLOW (A) for all non terminals A, apply the following rules until no more symbols can be added to any FOLLOW set.

1. Place $ in FOLLOW(S), where S is the start symbol, and $ is the input right end marker.
2. If there is a production A-> αBβ, then everything in FIRST (β) except ε is in FOLLOW(B).
3. If there is a production A->αB then Follow(B)=Follow(A) or a production A-> αBβ with FIRST(β) contains ε, then

   FOLLOW (B) = (First(β)- ε) U FOLLOW (A).

Example

Compute the FOLLOW values of the grammar

1. E -> TE’
2. E’ -> +TE’ | ε
3. T -> FT’
4. T’ -> *FT’ | ε
5. F -> (E) | id

FOLLOW Values

FOLLOW (E) = { $,), } Because it is the start symbol of the grammar.

FOLLOW (E’) = {FOLLOW (E)} satisfying the 3rd rule of FOLLOW()= { $,) }

FOLLOW (T) = { FIRST E’} It is Satisfying the 2nd rule U { FOLLOW(E’) }= {+, FOLLOW (E’)}= { +, $,) }

FOLLOW (T’) = { FOLLOW(T)} Satisfying the 3rd rule = { +, $,) }

FOLLOW (F) = { FIRST (T’) } It is Satisfying the 2nd rule U { FOLLOW(E’) }= {*, FOLLOW (T)}= { *, +, $,)}

First and Follow

NON TERMINAL	**FIRST**	**FOLLOW**
E	{ (, id }	{ $,) }
E’	{ +, ε }	{ $,) }
T	{ (, id}	{ +, $,) }
T’	{*, ε}	{ +, $,) }
F	{ (, id}	{ *, +, $,) }

Example

- Grammar

  S -> ABCDE

  A -> a|ε

  B -> b|ε

  C -> c

  D -> d|ε

  E -> e|ε

NON TERMINAL	**FIRST**	**FOLLOW**
S	{a, b, c}	{$}
A	{a, ε}	{b, c}
B	{b, ε}	{c}
C	{c}	{d, e, $}
D	{d, ε}	{e, $}
E	{e, ε}	{$}

Example

- Grammar

  S -> ACB|CbB|Ba

  A -> da|BC

  B -> g|ε

  C -> h|ε

NON TERMINAL	FIRST	FOLLOW
S	{d, g, h, b, a}	{$}
A	{d, g, h, ε}	{h, g, $}
B	{g, ε}	{$, a, h, g}
C	{h, ε}	{g, b, h, $}