SYNTAX ANALYSIS

LECTURE 3 - 4

CONTENT

- Top-Down Parsing
- Recursive Descent Parsing
- Concepts of FIRST and FOLLOW
- Examples

TOP DOWN PARSING

- Can be viewed as a problem of constructing a parse tree for the input string
- It starts from the root and create nodes of the parse tree in pre order (Depth First)
- Equivalently can be viewed as a leftmost derivation

TOP DOWN PARSING

Example:

E→TE'

E'→+TE'|ε

T→FT'

T'→*FT'|ε

F→(E)|**id**

Leftmost Derivation of string id + id * id

E → TE'

→ FT'E'

→ idT'E'

→ id e E'

→ id + TE'

→ id + FT'E'

→ id + id T'E'

→ id + id * F T' E'

→ id + id * id T' E'

→ id + id * id e E'

→ id + id * id e e

TOP DOWN PARSING

- At each of the step the key problem is that Determining the production to be applied for a nonterminal say A
- Once an A production is chosen the rest of the parsing process consists of matching the terminal symbols in the production body with the input string

RECURSIVE DESCENT PARSING

```
void A ( )
{
    Choose an A pro duction A → X1 X2 … Xk
     For ( i to k)
     {
          if (Xi is a nonterminal)
                call procedure Xi ( );
          else if (Xi equals the current input symbol a)
                advance the input to the next symbol
           else
                an error has occurred
     }
}
```

RECURSIVE DESCENT PARSING

- A recursive descent parsing program consists of a set of procedures one for each nonterminal
- Execution begins with the procedure for the start symbol
- The execution halts and announces success if its procedure body scans entire input string.
- Note that this pseudo code is nondeterministic since it begins by choosing the A production to apply in a manner that is not specified
- Backtracking is required

RECURSIVE DESCENT PARSING

Consider the grammar

S → c A d

A → a b | a

Derivation of string w = cad

If you go with S → cAd and then A → ab it leads to wrong string.

S → cabd

Backtracking is necessary

FIRST OF GRAMMAR

- If X is a terminal then FIRST (X) = X
- If X is a nonterminal and

  X → Y1 Y2 … Yk is a production for some k >=1

  Case 1:

  a. if for some i, 'a' is in FIRST(Yi) and

  b. epsilon is in all of FIRST Yj where j = 1, 2, … i –1

  Then add 'a' in the FIRST(X)

  Case 2:

  If epsilon is in FIRST(Yj) for all j = 1, 2, … , k

  then add epsilon in FIRST (X)

- If X → e is a production then add epsilon in FIRST(X)

FIRST OF GRAMMAR

Example 1:

E→TE'

E'→+TE'|ε

T→FT'

T'→*FT'|ε

F→(E)|**id**

Solution:

- FIRST(E) = FIRST(T) = FIRST(F) = { (, id}
- FIRST(E') = { +, ε }
- FIRST(T) = FIRST (F) = { (, id }
- FIRST (T') = { *, ε }
- FIRST (F) = { (, id }

FIRST OF GRAMMAR

Example 2

- A -> BC
- B -> Ax | x
- C -> yC | y

Solution

- In A-> BC

  FIRST(A)={FIRST(B) U FIRST (C)} if B -> Є is true
- FIRST(A)={FIRST(B)} if B -> Є is false
- FIRST(A) = {x}
- FIRST(B) = {x}
- FIRST(C) = {y}

FOLLOW OF GRAMMAR

Rule 1:

Place $ in FOLLOW (S) where S is the start symbol and $ is the input right endmarker

Rule 2:

If there is a production A -> αBβ then everything in FIRST(β) except ε is in FOLLOW (B)

Rule 3:

If there is a production A -> αB or

a production A -> αBβ where FIRST(β) contains ε

then everything in FOLLOW (A) is in FOLLOW (B)

APPLY ABOVE RULES UNTIL THERE IS NO UPDATION IN FOLLOW LIST

FOLLOW OF GRAMMAR

	FIRST	FOLLOW
E	(, id	$,)
E'	+ , ε	$,)
T	(, id	
T'	* , ε	
F	(, id	

Example 1:

E→TE'

E'→+TE'|ε

T→FT'

T'→*FT'|ε

F→(E)|**id**

Solution:

1. FOLLOW (E) = { $,) }

Since E is start symbol and Production Rule F → (E)

2. FOLLOW (E') = FOLLOW (E) = { $,) }

By Rule 3 of FOLLOW

FOLLOW OF GRAMMAR

	FIRST	FOLLOW
E	(, id	$,)
E'	+ , ε	$,)
T	(, id	+ , $,)
T'	* , ε	+ , $,)
F	(, id	

Example 1:

E→TE'

E'→+TE'|ε

T→FT'

T'→*FT'|ε

F→(E)|**id**

Solution:

3. FOLLOW (T) = { FIRST (E') - ε } U { FOLLOW (E') }

= { + } U { $,) }

= { + , $,) }

4. FOLLOW (T') = FOLLOW (T)

= { + , $,) }

FOLLOW OF GRAMMAR

	FIRST	FOLLOW
E	(, id	$,)
E'	+ , ε	$,)
T	(, id	+ , $,)
T'	* , ε	+ , $,)
F	(, id	* , + , $,)

Example 1:

E→TE'

E'→+TE'|ε

T→FT'

T'→*FT'|ε

F→(E)|**id**

Solution:

5. FOLLOW (F) = { FIRST (T') - ε } U { FOLLOW (T') }

= { * } U { + , $,) }

= { * , + , $,) }

FOLLOW OF GRAMMAR

	FIRST	FOLLOW
A	x	$, x
B	x	y
C	y	$, x

Example 2:

A -> BC

B -> Ax | x

C -> yC | y

Solution:

1. FOLLOW (A) = { $ } U FIRST (x) … As A is the start symbol

   = { $, x }

2. FOLLOW (B) = FIRST (C)

   = { y }

3. FOLLOW (C) = FOLLOW (A)

   = { $, x }

FOLLOW OF GRAMMAR

	FIRST	FOLLOW
S	d, g, h, b, a, ε	
A	d , g , h , ε	
B	g, ε	
C	h , ε	

Example 3:

S → ACB | Cbb | Ba

A → da | BC

B → g | ε

C → h | ε

Solution:

1. FIRST (S) = FIRST (A) U FIRST (C) U FIRST (B) U FIRST (b) U FIRST (a)

   = { d, g, h } U { h , ε } U { g , ε } U { b , a }

   = { d , g , h , b, a, ε }

2. FIRST (A) = FIRST (d) U FIRST (B) U FIRST (C)

   = { d , g , h , ε }

3. FIRST (B) = { g , ε }

4. FIRST (C) = { h , ε }

FOLLOW OF GRAMMAR

	FIRST	FOLLOW
S	d, g, h, b, a, ε	$
A	d , g , h , ε	h , g , $
B	g, ε	a , h , g , $
C	h , ε	b , h , g , $

Example 3:

S → ACB | Cbb | Ba

A → da | BC

B → g | ε

C → h | ε

Solution:

1. FOLLOW (S) = { $ } … Since S is start symbol
2. FOLLOW (A) = { FIRST (C) - ε } U { FIRST (B) - ε } U FOLLOW (S)
   = { h , g , $ }
3. FOLLOW (B) = FIRST (a) U { FIRST (C) - ε } U FOLLOW (A) U FOLLOW (S)
   = { a , h , g , $ }
3. FOLLOW (C) = FIRST (b) U { FIRST(B) - ε } U FOLLOW (A)
   = { b , g , h, $ }

FOLLOW OF GRAMMAR

	FIRST	FOLLOW
S	a , b , d , ε	
A	a , b , d , ε	
B	b , d , ε	
D	d , ε	

Example 4:

S → ABD

A → a | BSB

B → b | D

D → d | ε

Solution:

1. FIRST (S) = FIRST (A)
   = { a } U FIRST (B)
   = { a } U { b } U FIRST (D)
   = { a , b , d , ε }
2. FIRST (A) = { a } U FIRST (B)
   = { a } U { b } U FIRST (D)
   = { a , b , d , ε }
3. FIRST (B) = { b } U FIRST (D)
   = { b , d , ε }
4. FIRST (D) = { d , ε }

FOLLOW OF GRAMMAR

	FIRST	FOLLOW
S	a , b , d , ε	b , d , $
A	a , b , d , ε	b , d , $
B	b , d , ε	a , b , d , $
D	d , ε	a , b , d , $

Example 4:

S → ABD

A → a | BSB

B → b | D

D → d | ε

Solution:

1. FOLLOW (S) = { $ } U { FIRST (B) - ε } U FOLLOW (A)

   = { $, b , d } U { FIRST (B) - ε } U { FIRST (D) - ε }

   = { b , d , $ }

2. FOLLOW (A) = { FIRST (B) - ε } U { FIRST (D) - ε } U FOLLOW (S)

   = { b , d , $ }

3. FOLLOW (B) = {FIRST (D) - ε } U FOLLOW (S) U {FIRST (S) - ε } U FOLLOW (A)

   = { d, b, a, $ }

3. FOLLOW (D) = FOLLOW (B) U FOLLOW (S)

   = { a , b , d , $ }