CFG → Compute FIRST, FOLLOW

→ Build LL-Parse Table

→ LL-Parsing using LL-Parse Table

Definition: FIRST(w)

Given a CFG G = (V, T, S, P), $a \in T$, and w, v $\in (V \cup T)^*$,

FIRST(w) = { a | the first terminal a in w $\Rightarrow a$v}.

$\lambda \in$ FIRST(w) if w $\Rightarrow \lambda$.

Knowing the first terminal that starts a sentential form helps to choose the next rule to apply in a derivation.

Algorithm for FIRST

Given a grammar G = (V, T, S, P), calculate FIRST(w) for w in $(V \cup T)^*$,

1. For $a \in T$, FIRST(a) = {a}
2. FIRST(λ) = {λ}
3. For A $\in$ V, set FIRST(A) = { }
4. Repeat:

   For every production A $\rightarrow$ w

   FIRST(A) = FIRST(A) $\cup$ FIRST(w)

   Until no more terminals or λ can be added to any FIRST set for variables.
5. For w = $x_1 x_2 x_3 \ldots x_n$ where $x_i \in (V \cup T)$

   (a) FIRST(w) = FIRST(x_1) - {λ}

   (b) For i = 2 to n do:

   if $x_j \Rightarrow^* \lambda$ for all j from 1 to $i - 1$ then

   FIRST(w) = FIRST(w) $\cup$ FIRST(x_i) - {λ}

   (c) If $x_i \Rightarrow^* \lambda$ for all i from 1 to n then:

   FIRST(w) = FIRST(w) $\cup$ {λ}

Definition: FOLLOW(A)

Given a CFG G = (V, T, S, P), $a \in T$, A $\in$ V, and w, v $\in (V \cup T)^*$,

FOLLOW(A) = { a | the first terminal a that immediately follows A in a sentential form vAaw }.

\$ $\in$ FOLLOW(S) always.

Algorithm for FOLLOW

Given a grammar G = (V, T, S, P), A, B ∈V, and w, v ∈ (V ∪ T)*,

1. $ is in FOLLOW(S)
2. For A → vB, FOLLOW(A) is in FOLLOW(B)
3. For A → vBw:
   (a) FIRST(w) − {λ} is in FOLLOW(B)
   (b) If λ ∈ FIRST(w), then FOLLOW(A) is in FOLLOW(B)

Algorithm to Build LL(1) Parse Table

For a rule A → w where A is non-terminal, w ∈(V ∪ T)∗

1. For each production A → w
   (a) For each *a* in FIRST(w), add w to LL[A, *a*]
   (b) If λ ∈ FIRST(w), add w to LL[A, b] for each b in FOLLOW(A)
2. Each undefined entry is error.

L1(1) Parsing Algorithm

```
push(S)
lookahead = get()
while the stack is not empty do {
        symbol = top()
        if symbol is a Terminal then: {
                if symbol == lookahead then:
                        pop()
                        lookahead = get()
                 else;  error  }
        else if symbol is a Variable then: {                    // entry.action is r, i.e. Reduce
                if LL[symbol, lookahead] is not error then:  {
                        pop()
                        push(LL[symbol, lookahead])  }
                else:   error } }                               // end while
if  lookahead ≠ $ then
         error
else:
        accept
```

Example:

1) **CFG:** (1) S → aSb, (2) S → A, (3) A → bAa, (4) A → c

2) **FIRST** and **FOLLOW**

	FIRST	FOLLOW
S	*a*, b, c	b, $
A	b, c	*a*, b, $

3) **LL(1) Parse Table**

	a	***b***	***c***	***$***
S	*a*Sb	A	A	
A		bA*a*	c	

4) **Trace String '*abcab*'$ with LR(1) Parse Table**