FIRST( )

In parsers and compilers, the **FIRST()** set of a variable (non-terminal) in a grammar is used in syntax analysis, particularly in **LL(1) parsing** and **predictive parsing**. The **FIRST()** set of a non-terminal contains all the terminal symbols that can appear at the beginning of some string derived from that non-terminal. It helps in constructing parsing tables.

Rules to Compute FIRST() Set

To compute **FIRST(A)** for a non-terminal A, apply the following rules:

1. **If A is a terminal, then:**

   $$FIRST(A) = \{A\}$$

   (A terminal's FIRST set is the terminal itself.)

2. **If $A \to \epsilon$ (A can derive an empty string):**

   $$\epsilon \in FIRST(A)$$

   (If A can directly derive ϵ, then ϵ is in FIRST(A).)

3. **If A has a production $A \to X_1X_2X_3...X_n$, then:**
   - Add **FIRST(X_1)** to **FIRST(A)** (excluding ϵ).
   - If X_1 can derive ϵ, then add **FIRST(X_2)** to **FIRST(A)**.
   - Continue this process until:
     - A non-terminal that **cannot** derive ϵ is found.
     - If all X_i (for all i) can derive ϵ, then add ϵ to **FIRST(A)**.

4. **Handling multiple productions:**
   If A has multiple productions:

   $$A \to X_1X_2|Y_1Y_2$$

   Compute **FIRST()** for each right-hand side and take the union.

Example

Given the grammar:

```
less

S → A B
A → a | ε
B → b | ε
```

Compute **FIRST()**:

- **FIRST(a)** = { **a** } (Rule 1)
- **FIRST(b)** = { **b** } (Rule 1)
- **FIRST(A)** = { **a, ε** } (Rule 2 & 3)
- **FIRST(B)** = { **b, ε** } (Rule 2 & 3)
- **FIRST(S)**:
  - A can derive a or ϵ, so add a to **FIRST(S)**.
  - A can be ϵ, so check B. B can derive b or ϵ, so add b.
  - Since both A and B can derive ϵ, add ϵ to **FIRST(S)**.
  - **FIRST(S)** = { **a, b, ε** }.

These rules help in constructing **predictive parsing tables** and determining whether a grammar is **LL(1)**.

Here's a summarized table of the **FIRST()** rules:

Rule	Condition	Effect
1	A is a **terminal**	**FIRST(A)** = { **A** }
2	$A \to \epsilon$	Add **ε** to **FIRST(A)**
3	$A \to X_1X_2X_3...X_n$	Add **FIRST(X_1)** - {**ε**} to **FIRST(A)**
4	X_1 can derive **ε**	Add **FIRST(X_2)** - {**ε**} to **FIRST(A)**, continue until a non-ε-producing symbol is found
5	If all X_i can derive **ε**	Add **ε** to **FIRST(A)**

Follow()

Rules to Compute FOLLOW() Set in Parsers and Compilers

The **FOLLOW()** set of a non-terminal A contains all the terminals that can appear immediately after A in some derivation. It is useful for constructing **LL(1) parsing tables** and determining where to apply productions.

Rules for Computing FOLLOW(A)

To compute **FOLLOW(A)**, apply these rules:

1. **Start Symbol Rule**
   - If S is the start symbol of the grammar, then add **$** (end-of-input marker) to **FOLLOW(S)**:

$$FOLLOW(S) = \{\$\}$$

2. **Rule for A → αBβ** (B is followed by β)
    - If a production is of the form:

      $$A \to \alpha B \beta$$

      Then:

      $$FIRST(\beta) - \{\epsilon\} \subseteq FOLLOW(B)$$

      (Add everything in **FIRST(β)** except ϵ to **FOLLOW(B)**.)
3. **Rule for A → αB (B is at the end of a production)**
    - If a production is of the form:

      $$A \to \alpha B$$

      Then:

      $$FOLLOW(A) \subseteq FOLLOW(B)$$

      (Everything in **FOLLOW(A)** is added to **FOLLOW(B)**.)
4. **Rule for A → αBβ where β can derive ε**
    - If B is followed by β and β can derive ϵ:

      $$FOLLOW(A) \subseteq FOLLOW(B)$$

      (Because B can be the last symbol due to β being nullable.)

Example

Given the grammar:

```less
S → A B
A → a | ε
B → b | ε
```

Step 1: Compute FIRST Sets

We already computed these earlier:

- **FIRST(A) = { a, ε }**
- **FIRST(B) = { b, ε }**
- **FIRST(S) = { a, b, ε }**

Step 2: Compute FOLLOW Sets

1. **Start Symbol Rule:**
   - **FOLLOW(S) = { $ }** (Rule 1)
2. **From S → A B:**
   - **FOLLOW(A) includes FIRST(B) - { ε } → FOLLOW(A) = { b }**
   - **B is at the end, so FOLLOW(S) ⊆ FOLLOW(B) → FOLLOW(B) = { $ }**
3. **From A → ε:**
   - **A can be ε, so FOLLOW(S) ⊆ FOLLOW(A) → FOLLOW(A) = { b, $ }**
4. **From B → ε:**
   - **B can be ε, so FOLLOW(S) ⊆ FOLLOW(B) → FOLLOW(B) = { $ }** (already included)

Final FOLLOW Sets

- **FOLLOW(S) = { $ }**
- **FOLLOW(A) = { b, $ }**
- **FOLLOW(B) = { $ }**

Summary of FOLLOW() Rules

Rule	Condition	Effect
1	Start symbol	Add **$** to FOLLOW(S)
2	$A \to \alpha B \beta$	Add **FIRST(β)** - {ε} to FOLLOW(B)
3	$A \to \alpha B$ (B is at the end)	Add **FOLLOW(A)** to FOLLOW(B)
4	$A \to \alpha B \beta$ where β → ε	Add **FOLLOW(A)** to FOLLOW(B)

These rules help in **LL(1) parsing table construction** and ensure correct handling of **nullable** productions.