Find Firsts

- ## Introduction

In syntax analysis, the concept of "First Set" or "Firsts" is crucial in determining the first terminal symbol(s) that can be derived from a given non-terminal symbol or sequence of symbols in a grammar. The First Set plays a fundamental role in parsing and building the parser tables in *compiler design*.

- ## Purpose

The purpose of the "Find Firsts" algorithm is to compute the First Set for each non-terminal symbol in the grammar. By calculating the First Sets, the parser can determine the viable options for expanding non-terminal symbols during parsing.

* Algorithm

1. Initialize the First Set for all terminal symbols as themselves.

2. For each non-terminal symbol A:

- If A → ε is a production rule, add ε to First(A).

- For each production rule *A → X1X2...Xk*:

- If ε belongs to First(X1), First(X2), ..., First(Xi) for 1 ≤ i < k, add First(Xi) to First(A).

- If ε does not belong to First(Xj) for any 1 ≤ j < k, add First(Xj) to First(A), and stop further expanding First(A).

3. Repeat step 2 until no more symbols can be added to any First Set.

- ## Example

Consider the following grammar:

```
S → AB

A → a | ε

B → b | ε
```

The First Set calculations:

- First(S) = { a }

- First(A) = { a, ε }

- First(B) = { b, ε }

• Conclusion

In conclusion, the "Find Firsts" algorithm is a key component in syntax analysis for compilers. By accurately computing the First Sets for non-terminal symbols, the parser can efficiently determine the next steps in parsing input code. Understanding and implementing this algorithm is essential for building a robust and efficient parser in compiler design.

Producer: Elham Jafari

Computer Engineering