Find Follows

- ## Introduction

In the field of compiler design, specifically in syntax analysis, the concept of "Follow Set" or "Follows" is essential for determining the terminal symbols that can immediately follow a non-terminal symbol in a grammar. The Follow Set is crucial for parsing algorithms to handle predictive parsing and error recovery efficiently.

- ## Purpose

The "Find Follows" algorithm aims to compute the Follow Set for each non-terminal symbol in the grammar. By calculating the Follow Sets, the parser can make decisions during parsing based on the expected terminal symbols following a particular non-terminal.

* Algorithm

1. Initialize the Follow Set for the start symbol of the grammar as { $ }, where $ is the end-of-input marker.

2. For each production rule $A \rightarrow \alpha B \beta$ where B is a non-terminal symbol:

- Add *First(β)* to *Follow(B)*, excluding ε.

- If β can derive ε or *β* is ε, add *Follow(A)* to *Follow(B)*.

3. Repeat step 2 for all non-terminal symbols until no more symbols can be added to any Follow Set.

- ## Example

Consider the following grammar:

```
S → AaB

A → a | ε

B → b | ε
```

The Follow Set calculations:

- Follow(S) = { $ }

- Follow(A) = { a, b, $ }

- Follow(B) = { a, $ }

- ## Conclusion

In conclusion, the "Find Follows" algorithm is a critical component in syntax analysis for compilers. By accurately computing the Follow Sets, the parser can navigate through the input code effectively and handle syntactic ambiguities and errors gracefully. Understanding and implementing this algorithm is vital for developing a robust and efficient parser in compiler design.

Producer: Elham Jafari

Computer Engineering