LR(0) Parser

- ## Introduction

In the field of **compiler design**, syntax analysis is a crucial step in processing programming languages. An **LR(0) parser** is a bottom-up parser that uses zero token lookahead to shift or reduce symbols in a parsing table. This parser is powerful and capable of parsing a wide range of **context-free grammars** efficiently.

- ## Purpose

The purpose of the LR(0) parser is to recognize the structure of the input program and determine if it conforms to the grammar rules provided. It aims to build a parse tree for the input program by shifting and reducing symbols based on the states and transitions defined in the parsing table.

* Algorithm

1. Construct *LR(0)* items representing the augmented grammar rules with a "dot" to mark the current position in the production.

2. Build a canonical collection of *LR(0)* items by applying closure and goto operations to determine the parser states.

3. Construct the *LR(0) parsing table* by filling in entries for shift, reduce, and goto actions based on the parser states and transitions.

4. Initialize the parsing stack with the start state and process the input token stream.

5. Repeat until the parsing stack is empty:

   a. Consult the parsing table to determine the action for the current state and input token.

   b. Perform the corresponding shift, reduce, or goto operation based on the table entry.

   c. Update the parsing stack accordingly.

- **Example**

Consider the augmented grammar:

S' → S

S → AA

A → a | b

Construct the LR(0) items, build the canonical collection of LR(0) items, and create the LR(0) parsing table for the provided grammar.

Parsing table entries:

- (0, a) -> Shift 2

- (0, b) -> Shift 3

- (1, $) -> Accept

- (2, a) -> Reduce A → a

- (2, b) -> Reduce A → b

- (3, a) -> Shift 4

- (3, b) -> Shift 5

- (4, $) -> Reduce A → a

- (5, $) -> Reduce A → b

• Conclusion

In conclusion, the *LR(0) parser* is a powerful tool in *syntax analysis* for compilers. By utilizing a bottom-up parsing approach and zero token lookahead, this parser can effectively process a wide range of context-free grammars and generate parse trees for input programs. Building and using a parsing table based on *LR(0) items* and states is essential for the successful implementation of the *LR(0) parser.*

Producer: Elham Jafari

Computer Engineering