LR(0) Parser Generator - README

Student Information

- **Name:** Kaushik Dey
- **Roll No.:** 20CS01043

Assignment 6 - LR(0) Parser Generator

Introduction

This project implements an LR(0) parser generator for context-free grammars. An LR(0) parser employs a bottom-up parsing technique to recognize and parse strings based on a provided grammar. This README serves as documentation for the code.

Grammar Representation

The code represents a context-free grammar using production rules. The grammar is depicted as a list of strings, with each string representing a production rule in the format `LHS -> RHS`. In this format, `LHS` signifies the left-hand side non-terminal symbol, and `RHS` denotes the right-hand side of the production.

Grammar Augmentation

Before constructing the LR(0) parser, the code performs grammar augmentation. This involves adding a new start symbol and a new rule to introduce the start symbol on the right-hand side of some rules. This augmentation ensures that the grammar becomes suitable for LR(0) parsing.

Closure and State Generation

The LR(0) parser generator constructs LR(0) states and calculates the closure for each state. Each state is represented as a set of LR(0) items, where an LR(0) item is a production rule with a dot (.) indicating the parsing cursor's position. State generation involves the following functions:

- `findClosure(input_state, dotSymbol)`: Computes the closure of a given state by identifying all possible transitions using the dot symbol.
- `computeGOTO(state)`: Computes the GOTO transitions for a given LR(0) state.
- `GOTO(state, charNextToDot)`: Computes a single GOTO transition for a state and a character adjacent to the dot.
- `generateStates(statesDict)`: Generates all LR(0) states using GOTO transitions.

FIRST and FOLLOW Sets

The code computes the FIRST and FOLLOW sets for non-terminal symbols. These sets are crucial for constructing the parsing table and handling reductions. The following functions are involved in FIRST and FOLLOW set computation:

- `first(rule)`: Computes the FIRST set for a given rule or symbol.
- `follow(nt)`: Computes the FOLLOW set for a given non-terminal symbol.

LR(0) Parsing Table

Once the LR(0) states, GOTO transitions, FIRST, and FOLLOW sets are computed, the code constructs the LR(0) parsing table. This parsing table is a 2D array that encodes the action and GOTO entries for each state and input symbol. The following function is responsible for parsing table construction:

- `createParseTable(statesDict, stateMap, T, NT)` : Constructs the LR(0) parsing table, incorporating SHIFT and REDUCE actions based on the computed states and transitions.