[Source Code link]
https://github.com/iuliagroza/Formal-Languages-Compiler-Design-Mini-Language/tree/main

Parser Algorithm

Students: Iulia Groza
Nicholas Halmagyi

Grammar Class Documentation

Overview

The Grammar class is designed to encapsulate the grammar of a programming language or any formal language defined by a set of rules. It primarily deals with context-free grammars (CFGs) and provides functionalities to read grammar from a file, display various components of the grammar, and check if the grammar conforms to the CFG properties.

Data Structure Components

Nonterminals (self.nonterminals):
- **Type**: List (or Set)
- **Description**: This stores all the nonterminal symbols of the grammar. Nonterminals are symbols that can be replaced by a group of other symbols according to the production rules of the grammar. In most implementations, nonterminals are represented by uppercase letters or words enclosed in angle brackets (e.g., <expression>).

Terminals (self.terminals):
- **Type**: List (or Set)
- **Description**: This holds all the terminal symbols of the grammar. Terminals are the basic symbols from which strings (sentences) are formed in the language. They are the atomic elements that cannot be broken down further. Typically, terminals include characters, numbers, and special symbols.

Productions (self.productions):
- **Type**: Dictionary
- **Description**: This is a key-value pair representation of the production rules. The keys are nonterminal symbols, and the values are lists of possible replacements (right-hand side of the production rules). Each replacement is a string that represents a sequence of terminals and/or nonterminals.

Start Symbol (self.start_symbol):
- **Type**: String
- **Description**: This represents the start symbol of the grammar, which is the initial nonterminal from where the parsing or derivation starts. The start

symbol plays a critical role in defining the language generated by the grammar.

Methods

read_from_file(self, file_path):

- Reads the grammar from a file and populates the nonterminals, terminals, productions, and start symbol.

print_nonterminals(self):

- Displays the nonterminals of the grammar.

print_terminals(self):

- Displays the terminals of the grammar.

print_productions(self):

- Prints the production rules of the grammar.

print_productions_for_nonterminal(self, nonterminal):

- Displays the production rules for a specific nonterminal.

is_cfg(self):

- Checks and returns whether the grammar is a context-free grammar.