Universitat Pompeu Fabra Barcelona

Aranich, Marcel - 251453
Quera, Clàudia - 231197
Villarino, Jorge - 231351
Solans, Arnau - 216530
Garcia, David - 251587
Prat, Ariadna - 251281

DESIGN DOCUMENT

Project Overview

In this lab, we aim to develop the bottom-up parsing algorithm using a shift/reduce automaton (sra) for a given grammar. The grammar consists of production rules for arithmetic expressions involving addition, multiplication, parenthesis, and numeric literals.

The sra will also create an abstract syntax tree (ast) and we will translate it to the corresponding instructions in assembly.

Project Goals and Objectives

The main goal of this project is to implement a parser that can read and evaluate simple arithmetic operations involving addition and multiplication (alongside with parenthesis).

- Use the bottom-up parsing algorithm to detect the correct strings that are accepted by the automaton.
- Use the AST to determinate how the operations should be performed.
- Designing a well-structured codebase that works correctly for all the specifications described in the lab statement.

Derive the shift/reduce automaton

Production rules:

1. s → e
2. e → e + t
3. | t
4. t → t * f
5. | f
6. f → (e)
7. | NUM
8. NUM → DIGIT
9. | NUM DIGIT
10. DIGIT → {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}

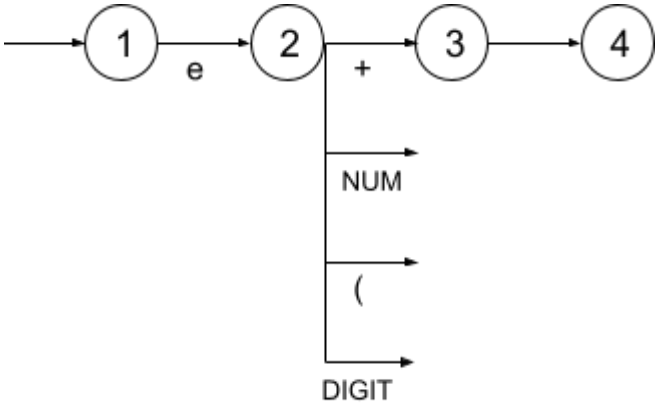
Universitat Pompeu Fabra Barcelona

Aranich, Marcel - 251453
Quera, Clàudia - 231197
Villarino, Jorge - 231351
Solans, Arnau - 216530
Garcia, David - 251587
Prat, Ariadna - 251281

These are the steps we followed to derive the shift/reduce automaton:

1. Construct the State Diagram:
   - Begin by identifying the states of the automaton.
   - For each state, determine the transitions based on the current symbol and the action to take (shift or reduce).
   - Indicate reductions at states where applicable.

State	e	+	*	(	)	NUM	DIGIT	$
0 (reject)	0	0	0	0	0	0	0	0
1 (start)	2	0	0	0	0	0	0	0
2	2	3	0	4	0	4	4	0
3	2	0	0	5	0	4	4	0
4	0	0	6	0	4	8	8	0
5	2	0	0	5	0	4	4	0
6	7	0	0	0	0	8	8	0
7	0	7	0	0	7	0	0	7
8	0	8	6	0	8	0	0	8
9	2	0	0	5	0	4	4	0



Universitat Pompeu Fabra Barcelona

Aranich, Marcel - 251453
Quera, Clàudia - 231197
Villarino, Jorge - 231351
Solans, Arnau - 216530
Garcia, David - 251587
Prat, Ariadna - 251281

2. Rename the States:

- Once we have the state diagram, rename the states using numerical values (e.g., 1, 2, 3,...).
- Ensure that each state is uniquely identified by its numerical value.

3. Generate the Table:

- Create a table representation of the shift/reduce automaton.
- Rows of the table correspond to states, and columns correspond to input symbols.
- Populate the table entries with actions (shift, reduce, or accept) based on the transitions in the state diagram.

4. Implement the Shift/Reduce Automaton Engine:

- Write a program that simulates the behavior of the shift/reduce automaton using the table generated in the previous step.
- The program should include functionalities for shifting symbols onto the stack and reducing the right-hand side of productions.
- Ensure that the implementation is general enough to handle different language specifications by allowing the table to be changed dynamically.

5. Testing:

- Test the implemented engine with example inputs to verify that it correctly parses and analyzes strings according to the language specifications.
- Test the engine with both provided language specifications to ensure its generality.

6. Documentation:

- Document the steps taken to derive the shift/reduce automaton, including the construction of the state diagram, renaming of states, table generation, and implementation of the engine.
- Provide explanations for each step and any considerations made during the process.

By following these steps, you'll be able to derive the shift/reduce automaton for the given language and implement a general engine that can handle different language specifications.

Design of the program

Universitat Pompeu Fabra Barcelona

Aranich, Marcel - 251453
Quera, Clàudia - 231197
Villarino, Jorge - 231351
Solans, Arnau - 216530
Garcia, David - 251587
Prat, Ariadna - 251281

Data structure

As we will use a Shift-Reduce Automata for the LR parsing, we will need to define a stack and a states diagram data structure, each with its necessary fields:

- Stack: A data structure that will represent the stack we will use during the parsing process, to store the symbols during the parsing. The fields that our stack will have are the items and the top:
    - Items → The items will be an array or linked list which will contain and store the symbols. These symbols will be pushed or popped into/from the stack, depending on the operation applied to them.
    - Top → The field top will contain a pointer to the top of the stack.
- State diagram→ This represents the state diagram of the Shift-Reduce Automata, including the transition and reduction rules. Its fields will be:
    - states → Array with the information of all the states and states transitions. Each state will contain a set of all the possible actions (accept, reduce or shift), depending on the input symbol.
    - Production rules → List of the rules that will be used to redude.