CPSC-323
Project 2
Documentation File

Contributors:

Guido Asbun	https://github.com/guidoasbun
Cade Duncan	https://github.com/Stingrayss
Briyana Verdugo	https://github.com/B-T-V

The purpose of this project is to analyze and implement a Parser (Syntax Analyzer) using the Java 21 programming language. This program implemented the Parser using LR parsing for syntactic analysis.

The parameter of this program is that it traces input strings over 6 characters:

id	+	*	)	(	$

And it ends with a $.

This program uses the provided CFG:

CFG	FIRST	FOLLOW
E → E + T	FIRST(E) = { (, id }	FOLLOW(E) = { + ,) , $ }
E → T	FIRST(E) = { (, id }	FOLLOW(E) = { + , * ,) , $ }
T → F		
F → (E)		
F → id		

And the provided LR Parsing Table:

State	id	+	*	(	)	$	E	T	F
0	S5			S4			1	2	3
1		S6				ACC			
2		R2	S7		R2	R2			
3		R4	R4		R4	R4			
4	S5			S4			8	2	3
5		R6	R6		R6	R6			
6	S5			S4				9	3
7	S5			S4					10
8		S6			S11				
9		R1	S7		R1	R1			
10		R3	R3		R3	R3			
11		R5	R5		R5	R5			

As a java program, 3 classes were used to implement the logic:

App.java:
The main class where the program is launched. This class is responsible for the interface with the user, as it displays the 3-stack implementation for the test words and then asks the user to input another word for testing. If the word is of the same characters and ends with $, the program will return the stack implementation of the user input word.

ParserState:
This class stores the possible actions and state of the transitions.

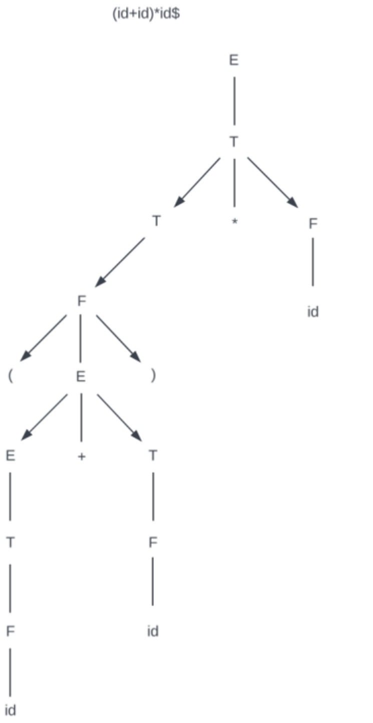
LRParser:
This class does the heavy lifting as it contains the main logic. Both tables were hard coded in the Java program.

- The CFG logic is coded in the LRParser class as a method named handleCFG().
- The LR Parsing table is coded in the the LRParser class as a method named initializeLRTable.
- And the tokenize() method converts the strings into tokens.

This program has 3 test words it tested:

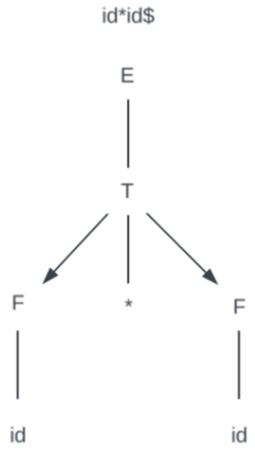
(id+id)*id$

Accepted word



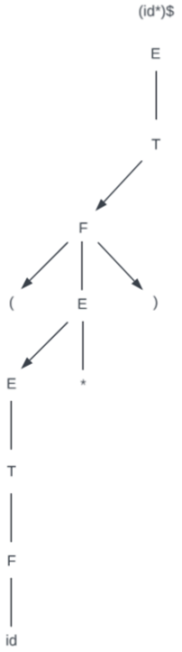
id*id$

Accepted word



(id*)$

Not accepted word


(id*)$
E
T
F
(
E
)
E
*
T
F
id