Asian Institute of Technology

AT70.07

Programming Language and Compilers

Professor: Phan Minh Dung

Parser Analysis: Calculator Project

Minn Banya (st124145)

An assignment submitted in partial fulfillment of the

requirements for AT70.07

March 2024

Grammar Description:

The grammar for our calculator project is defined as follows:

Rule 0 E' → E

Rule 1 E → E * T

Rule 2 E → T

Rule 3 T → T + F

Rule 4 T → F

Rule 5 F → N

Where:

- E, T, and F represent expressions, terms, and factors respectively.
- N represents numeric values.

Canonical LR(0) items

state 0

(0) E' →. E

(1) E → . E * T

(2) E → . T

(3) T → . T + F

(4) T → . F

(5) F → . N

state 1

(0) E' → E .

(1) E → E . * T

state 2

(2) E → T .

(3) T → T . + F

state 3

(4) T → F .

state 4

(5) F → N .

state 5

(1) E → E * . T

(3) T → . T + F

(4) T → . F

(5) F → . N

state 6

(3) T → T + . F

(5) F → . N

state 7

(1) E → E * T .

(3) T → T . + F

state 8

(3) T → T + F .

Parser Type:

For our calculator project, we chose to implement a Bottom-up SLR(1) parser. This choice was made due to the efficiency and scalability of SLR(1) parsing for handling context-free grammars. The SLR(1) parser provides strong parsing capabilities by efficiently building a parse tree from the input token stream.

Parsing Table:

The parsing table is constructed based on the LR(1) parsing algorithm. It consists of entries representing shift (s), reduce (r), and accept actions. These actions determine the next step in the parsing process based on the current state and input token.

Parsing Table

	Action				goto		
State	*	+	N	$	E	T	F
0			s4		1	2	3
1	s5			acc			
2	r2	s6		r2			
3	r4	r4		r4			
4	r5	r5		r5			
5			s4			7	3
6			s4				8
7	r1	s6		r1			
8	r3	r3		r3			

Method of Translation:

The translation process involves evaluating the value, prefix notation, and postfix notation of the input token stream simultaneously during parsing. Semantic rules are applied based on the grammar productions to compute these properties:

- **Value Evaluation**: The value of each expression is computed recursively based on the values of its constituent parts.
- **Prefix Notation**: Prefix notation is generated by traversing the parse tree and concatenating operators and operands in the correct order.
- **Postfix Notation**: Similar to prefix notation, postfix notation is derived by traversing the parse tree and arranging operators and operands accordingly.

Rules for value evaluation

Production	Semantic Rules
$E \rightarrow E_1 * T$	$E.val := E_1.val * T.val$
$E \rightarrow T$	$E.val := T.val$
$T \rightarrow T_1 + F$	$T.val := T_1.val + F.val$
$T \rightarrow F$	$T.val := F.val$
$F \rightarrow N$	$F.val := N.lexval$

Rules for prefix notation

Production	Semantic Rules
$E \rightarrow E_1 * T$	$E.pf := '*' \| \| E_1.pf \| \| T.pf$
$E \rightarrow T$	$E.pf := T.pf$
$T \rightarrow T_1 + F$	$T.pf := '+' \| \| T_1.pf \| \| F.pf$
$T \rightarrow F$	$T.pf := F.pf$
$F \rightarrow N$	$F.pf := N.pf$

Rules for postfix notation

Production	Semantic Rules
$E \rightarrow E * T$	$E.pf := E_1.pf \| \| T.pf \| \| '*'$
$E \rightarrow T$	$E.pf := T.pf$
$T \rightarrow T + F$	$T.pf := T_1.pf \| \| F.pf \| \| '+'$
$T \rightarrow F$	$T.pf := F.pf$
$F \rightarrow N$	$F.pf := N.pf$

Integration of Parser and Translation:

The parser and translation components are tightly integrated, with semantic actions embedded within the grammar rules. As the parser constructs the parse tree, semantic actions are triggered to compute the value, prefix notation, and postfix notation of each expression. This seamless integration ensures efficient and accurate evaluation of input expressions.

How to run:

Please change the working directory to the project folder and run command 'python3 main.py' in the terminal to run the calculator.

Conclusion:

In conclusion, our calculator project implements a Bottom-up LR(1) parser coupled with semantic actions to translate input expressions into their corresponding values, prefix notation, and postfix notation. By leveraging LR(1) parsing and semantic analysis, our calculator provides robust parsing capabilities and accurate expression evaluation.