CPSC 323 Compilers and Languages Project Two
Syntax Analysis
Spring 2025

Overview

The second phase of a standard compiler is syntax analysis. The job of the syntax analyzer is to read in the token stream as it is produced by the lexical analyzer, parse the token stream by following the production rules of the grammar defining the language, and output a syntax tree. In essence, the parser is checking that the source code is composed of valid sentences. If it is, then it will be parsed and output as a syntax tree. If it is not, then it must reject the invalid sentence and report an error. This syntax tree output will be utilized by the semantic analyzer. The work of a parser is typically performed by a PDA, or something computationally equivalent.

In this project, you will be building a small, bottom-up parser based on the simple expression grammar from our textbook. You will then use it to parse and pass all provided test cases. You will be provided with a table that contains all of the shift, reduce, and state change information necessary to construct the parser. A concrete syntax tree is the expected output of your project.

Learning Outcomes

1. Develop familiarity working with and implementing theoretical concepts such as a CFG, and PDA in order to construct a functioning syntax analyzer.
2. Develop familiarity with converting specifications and requirements into a correct, working piece of software.
3. Develop skills in goal setting, communication, and time management in a group context.

Requirements

Your project must satisfy the following requirements in order to receive full marks:

1. The parser must be a bottom-up, shift-reduce parser.
2. The parser must use a stack and output a concrete syntax tree.
3. The parser must return an appropriate error for invalid sentences.
4. The parser must successfully parse (or correctly reject) all provided test cases.

Grading

The project will be graded based on how many of the above requirements your submission satisfies. If a project successfully meets all of the requirements above, then it will be awarded full marks. Points will be deducted for each requirement that is not met; partial credit is possible. The project will be worth 60 points. Per the course syllabus, there will be no late work accepted.

Submission Items

The following items must be submitted in a zipped file format to Canvas no later than the assigned due date of the project:

1. Source code.
2. Screenshot of input and successful output.
3. Your transformed grammar (from the given grammar below).
4. A very brief project report (pdf format) that outlines the following:
    a. Group members.
    b. Which member did what work.
    c. Programming language used.
    d. An explanation of your design choices and how your parser works. You may include a diagram or design sketch.
    e. Citation of any external resources utilized (stackoverflow, reddit, wikipedia, etc).
    f. A statement that no AI resources were utilized in the production of any work for this project.

Test Cases

1. id + id + id
2. (id - id * id)
3. id * id / id * id
4. (id - id) / (id + id)
5. (id + (id / id * id / id) / id)
6. (id + id
7. id id id
8. - id + id / id)
9. (id () id)
10. (id * id * id / id))

The Table (next page)

(1) E → E + T
(2) E → E – T
(3) E → T
(4) T → T * F
(5) T → T / F
(6) T → F
(7) F → (E)
(8) F → id

State	id	+	-	*	/	(	)	$	E	T	F
0	S5					S4			1	2	3
1		S6	S7					Acc			
2		R3	R3	S8	S9		R3	R3			
3		R6	R6	R6	R6		R6	R6			
4	S5					S4			10	2	3
5		R8	R8	R8	R8		R8	R8			
6	S5					S4				11	3
7	S5					S4				12	3
8	S5					S4					13
9	S5					S4					14
10		S6	S7				S15				
11		R1	R1	S8	S9		R1	R1			
12		R2	R2	S8	S9		R2	R2			
13		R4	R4	R4	R4		R4	R4			
14		R5	R5	R5	R5		R5	R5			
15		R7	R7	R7	R7		R7	R7			