CS301P Compiler Design Laboratory Exercises Week-5

Date: Sep 01 2024

Objectives

- To implement a parser to verify arithmetic expressions.
- To deal with error handling and error reporting mechanisms using *error* token.
- To learn construction of syntax tree (one of the intermediate representations).

Exercise Problems

1. Design a CFG to generate arithmetic expressions that involve binary operators $+, -, *, /, \%$, unary minus $-$, and assignment operator $=$. The expressions may contain parenthesis $(,)$ as well.

2. Construct a parser to verify the given arithmetic expressions and accept all valid arithmetic expressions, and reject the invalid expressions by providing appropriate error diagnostic messages.

   Sample Input and expected Output:

rem = a % 6;	→	Accepted
r0 = (a * b) + c - (b / a);	→	Accepted
r1 = (x / y) * z - (x % y);	→	Accepted
r2 = ((a - b) * c) / (a + b);	→	Accepted
rem = a % 2.5;	→	Rejected - Invalid operand
r3 = (x + y * z;	→	Rejected - Close parenthesis missing
diff = x y;	→	Rejected - missing operator
r4 = -x * y + (x - y) * 2;	→	Accepted

3. Extend the parser to build a syntax tree for each valid binary arithmetic expression and print the expression in postfix form by performing post-order travel on the syntax tree.

Note

- In case you are finding difficulty, consider a limited set of operators, then come up with a CFG and the corresponding parser.
- Error handing is important.
- All other details are same as the previous lab.