# Implementation Of LALR Parser Using Lex and Yacc

Aayush Nirwan
1601001

Abhinav Jha
1601002

## I. Introduction

Yacc (Yet Another Compiler-Compiler) is a computer program for the Unix operating system developed by Stephen C. Johnson. It is a Look Ahead Left-to-Right (LALR) parser generator, generating a parser, the part of a compiler that tries to make syntactic sense of the source code, specifically a LALR parser, based on an analytic grammar written in a notation similar to Backus Naur Form (BNF). Lex is a computer program that generates lexical analyzers ("scanners" or "lexers"). Lex is commonly used with the yacc parser generator. Lex, originally written by Mike Lesk and Eric Schmidt and described in 1975, is the standard lexical analyzer generator on many Unix systems, and an equivalent tool is specified as part of the POSIX standard. Lex and yacc help us write programs that transform structured input. Lex turns these regular expressions into a form that the lexer can use to scan the input text extremely fast.

## II. Implementation

Using lex file("subc.l"), we are generating tokens. These tokens are printed in a file "output.txt". Symbol table is also implemented in the lex file using structure. Symbol table is printed in the file "symbol.txt". Variables will only be inserted in the symbol table if they are present in the declaration statements.
For Type-check,
In each expression we store the data-type of each variable and literal in some temporary buffer.When the data-types of RHS doesn't match with LHS of expression, we throw the error.
eg: float = float + int; (Accepted)
int = int + int; (Accepted)
int = int + (float|long|double); (Error)
For Finding "Cost of Evaluation" of each expression, we have used Ad Hoc SDT in yacc file, depending on the type of operation we have added the corresponding cost to cost variable(LHS.cost | $$).We have used a variable "tcost" to keep track of the total cost. Results are also printed in file "cost.txt".
For Tackling multiple enteries problem,
We have done it using Symbol table. When a variable is declared and it is already present in Symbol table then a warning appears showing "Multiple Declaration".

### A. Robustness

Robustness describes how reliable our program is, especially under extreme conditions such as extreme workload or unpredictable user inputs. This program would never crash no matter what the user inputs. Even if the user enters invalid data, program will handle it properly.

### B. Correctness

Correctness of an algorithm is asserted when it is said that the algorithm is correct with respect to a specification. Correctness refers to the input-output behaviour of the algorithm. This program might render some of test cases but it will work properly in most of test cases.

## V. Error Handling

Errors are handled in a special manner. Along with printing "error" in the output, program also prints the type of the error. Parsing will halt once an error has occurred.

## VI. Prerequisite

Lex and yacc should be installed before hand.