CS320
Lab 5
Semantic analysis

Srinibas Swain (srinibas@iiitg.ac.in)
Submission date: 11:59 PM, 27^{th} March 2022

In this lab we will use *syntax directed definition* and *synthesized attributes* to create a semantic analyser which performs type checking. We will used the expression grammar (discussed in the class) for this purpose.

1 EXPRESSION

The language **EXPRESSION** consists of arithmetic expressions as defined in the class. The grammar for EXPRESSION is given below.

1. $E -> E + T$
2. $E -> E - T$
3. $E -> T$
4. $T -> T * F$
5. $T -> T/F$
6. $T -> F$
8. $F -> num$
9. $F -> id$
10. $F -> (E)$

2 Type checking

We restrict our domain to the set of objects (can be numbers/strings) on which the binary operations are performed. The operations (addition, subtraction, division, multiplication, etc.) can be generalised as a binary operation, which is performed on two elements from set of objects.

A binary operation can be defined as an operation $\oplus$ which is performed on a set A. The function is given by $\oplus : A \oplus A \rightarrow A$. So the operation $\oplus$ performed on operands a and b is denoted by $a \oplus b$.

Rules for type checking are applicable on the type of the type of operands before performing an operation. Information regarding the rules come from:

- a symbol table, where we store the type of variable, a list of parameters for a function etc.
- synthesized attributes defined over the production rules of the expression grammar.

The rules of type checking are the following:

- If both operands are an operation are of the same time the result should also be of the same type.
- If you are allowing operators of different types then define clearly what should be the type of the result.

3 Code flow

Input: an expression E
output: type of the result produced by E

You have to follow the following steps:

- Create the lexical analyser for expression.
- Use the output of the lexical analyser to create a symbol table.
- Create a parser (LALR(1)) for the input E.
- Implement the type checker on the output produced by the parser, i.e. while traversing the parse tree apply the rules when there is a reduction. The root of the parse tree should give the desired answer. The type of the symbols should be fetched from the symbol table in this process. Also make sure that there should be a attChecker() function that checks the type of the attributes.