CS301P Compiler Design Laboratory Exercises Week-4

Date: Aug 29 2023

Objectives

- To learn the working of a top-down parser: Recursive descent parser

Exercise

Recursive descent parser is a top down parser that consists of a set of procedures, one for each nonterminal. Execution begins with the procedure for the start symbol, which halts and announces success if its procedure body scans the entire input string. Pseudocode for a typical nonterminal A appears as shown following.

```
void A(){
Choose an A-production , A → $X_1 X_2 \ldots X_k$
for ( i = 1 to k ) {
        if ( $X_i$ is a nonterminal )
                call procedure $X_i()$
        else if ( $X_i$ equals the current input symbol $a$ )
                 advance the input to the next symbol;
        else /* an error has occurred */;
        }
}
```

In this lab, you need to implement back-tracking free recursive descent parser for the following grammar G. Here, the terminals are $\{+, *, id, num, (,)\}$

$E \rightarrow TE^1$
$E^1 \rightarrow +TE^1|\epsilon$
$T \rightarrow FT^1$
$T^1 \rightarrow *FT^1|\epsilon$
$F \rightarrow id|num|(E)$

The following are some of the valid expressions. The parser should read the expressions from an input file (taken as command-line arguments). For each expression, it should output "yes" if the expression is valid or "no" otherwise,

- $i * i * i + i$

- $i + i + i$
- $x + y + 3$
- $x * (y + z)$
- ...
- ...

Submission Guidelines

Same as week#1 except that the final executable should be named as **parser**.

Evaluation Guidelines

Same as week#1

Academic Honesty

Same as week#1