CS301P Compiler Design Laboratory Exercises Week-3

Date: Aug 19 2024

Objectives

- To design *Context-Free Grammars (CFGs)* for the generation of strings corresponding to various simple languages.
- To implement parsers for each language using *Yet Another Compiler Compiler (YACC).*

Exercise Problems

1. Design a Context-Free Grammar (CFG) to accept the valid strings of the following language. $\{w \in \{x.x^R \mid x \in \{0,1\}^*\}$ and the number of 0's in w is divisible by 4$\}$, where x^R is the reverse string of x.

   Next is to mplement a parser for the above grammar.

   Sample Input:

   001100 → Accepted

   001010 → Rejected

   110011 → Rejected

2. Design a CFG that generates valid strings representing complex numbers. The valid format for a complex number is either 'a + ib' or 'a + bi', where: 'a' and 'b' are integers or floating-point numbers (including negative numbers), and 'i' is the imaginary unit. Spaces between components should be ignored.

   Next is to implement a parser that accepts only the valid complex numbers and rejects anything else.

   Sample Input:

3 + 4i	→	Accepted
-2 + 5i	→	Accepted
3.5 + i2	→	Accepted
-4.7 - 6.8i	→	Accepted
0 + 1i	→	Accepted
3 + 4	→	Rejected - missing 'i'
i + 4b	→	Rejected - Invalid format
3 + i	→	Rejected - missing imaginary part coefficient

3. Design a CFG to accept the all valid dates given in the format **DD-MON-YYYY**, where DD, MON, and YYYY take string type input from the valid ranges, [1 . . . 31], {Jan, Feb, . . ., Dec}, and [0000 . . . 9999], respectively.

Next is to implement a parser for the above grammar.

Sample Input:

10-Jan-2024 → Accepted

30-Feb-2001 → Rejected - Date out of range

29-Feb-1992 → Accepted

Feb-29-1992 → Rejected - Invalid format

29-Abc-1992 → Rejected - Invalid month

29-dec-2000 → Accepted

References

1. Flex Manual https://westes.github.io/flex/manual/
2. Lex & Yacc by John R. Levine, Tony Mason, and Doug Brown

Sumission Guidelines

1. The lex and yacc files should be named as probX.l, probX.y, respectively. Here X indicates the problem number.
2. The final target for each problem should be named as **parser**.
3. For each problem, consider taking multiple inputs separated by semicolon ; operator and print Accepted or Rejected with a reason, for each input.

   Note: Everything else shall remain same as previous labs.