CS315 Course Project I

Subhan Ibrahimli
Cavid Gayibli
Selim Can Gulsever

March 4, 2019

Introduction

The project is about the design of a new programming language for propositional calculus.This newly designed language will be similar to imperative languages. The main difference is that this new language will be specifically for propositional calculus. This part of the project is about the design of the language and the implementation of its lexical analyzer.

BNF

Backus-Naur Form (BNF) is a syntax for describing syntax. It's used to write a formal representation of a context-free grammar. In computer science, BackusNaur form or Backus normal form (BNF) is a notation technique for context-free grammars, often used to describe the syntax of languages used in computing, such as computer programming languages, document formats, instruction sets and communication protocols. In that part, there will be demonstrated our programming language BNF.

Description of Non-Terminals

In this section, there will be illustrated the descriptions of the given BNF forms.

Conclusions

In this section, we describe the results

1.BNF

1. <main_program> ::= LETSROLL LEFT_PARANT RIGHT_PARANT LEFT_BRACE NEW_LINE <program> RIGHT_BRACE

2. <program> ::= <statements> | <statement> <statements>

3. <statements> ::= <statement><new_line> | <statement><new_line><statements>

4. <statement> ::= <while> | <if> | <for> | <single_state> | <function_type>

5. <new_line> ::= NEW_LINE | NEW_LINE <new_line> | <comment>

6. <while> ::= WHILE LEFT_PARANT <expression> RIGHT_PARANT LEFT_BRACE <statements> RIGHT_BRACE

7. <if> ::= IF LEFT_PARANT <expression> RIGHT_PARANT LEFT_BRACE NEW_LINE <statements> RIGHT_PARANT | IF LEFT_PARANT <expression> RIGHT_PARANT LEFT_BRACE NEW_LINE <statements> RIGHT_BRACE ELSE LEFT_BRACE NEW_LINE <statements> RIGHT_BRACE

8. <for> ::= FOR LEFT_PARANT <declaration> COLON <expression> COLON <assignment_op> RIGHT_PARANT LEFT_BRACE NEW_LINE <statements> RIGHT_BRACE

9. <single_state> ::= <assign_state> | <declaration_state> | <return_state> | <function_state>

10. <assign_state> ::= <variable_name> ASSIGN_OP <expression>

11. <declaration_state> ::= <var_type> <assign_state> | <var_type> <var_names>

12. <expression> ::=

<term> <low_op> <expression>

| <term>

| <term> <comp_op> <expression>

| <term> <prop_op_med> <expression>

| <term> <prop_op_low> <expression>

13. <term> ::=
        <var_name> <high_op> <term>
        | <prop_op_high> <variable_ident>
    | <integer> <high_op> <term>
    | <float> <high_op> <term>
    | <var_name>
    | <integer>
    | <float>
    | <string>
14. <low_op> ::= PLUS | MINUS
15. <high_op> ::= MULTIPLY | DIVISION | EXCLUSIVE_OR
16. <prop_op_high> ::= NEGATION
17. <prop_op_med> ::= DISJUNCTION | CONJUNCTION
18. <prop_op_low> ::= IMPLICATION | DOUBLE_IMPLICATION
19. <comp_op> ::=
        ASSIGNMENT_OP
    | SMALL
    | GREAT
    | EQUALITY_CHECK
    | SMALL_OR_EQUAL
    | GREAT_OR_EQUAL
    | NOT_EQUAL
20. <var_name> ::= VAR_NAME
21. <var_names> ::= <var_name> | <var_name> COMMA <var_names>
22. <var_type> ::= TYPE_INT | TYPE_STRING | TYPE_FLOAT
23. <function_state> ::=
        <var_name> LEFT_PARANT RIGHT_PARANT
        | <var_name> LEFT_PARANT <arguments> RIGHT_PARANT
24. <return_state> ::= RETURN <expression> <new_line>

25. <arguments> ::= <argument> | <argument> COMMA <arguments>

26. <argument> ::= <integer> | <float> | <string> | <var_name>

27. <function_type> ::= <non_void_function> | <void_function>

28. <integer> ::= INT

29. <float> ::= FLOAT

30. <string> ::= STRING

31. <non_void_function> ::= <var_type> <var_name> LEFT_PARANT RIGHT_PARANT LEFT_BRACE NEW_LINE <statements> <return_state> RIGHT_BRACE | <var_type> <var_name> LEFT_PARANT <parameters> RIGHT_PARANT LEFT_BRACE NEW_LINE <statements> <return_state> LEFT_BRACE

32 <void_function> ::= VOID <var_name> LEFT_PARANT RIGHT_PARANT LEFT_BRACE NEW_LINE <statements> <return_state> RIGHT_BRACE | VOID <var_name> LEFT_PARANT <parameters> RIGHT_PARANT LEFT_BRACE NEW_LINE <statements> <return_state> LEFT_BRACE

33. <parameters> ::= <var_type> <var_name> | <var_type> <var_name> COMMA <parameters>

34. <comment> ::= COMMENT <new_line>

35. <print> ::= PRINT LEFT_PARANT <expression> RIGHT_PARANT

36. <println> ::= PRINT_LINE LEFT_PARANT <expression> RIGHT_PARANT

2. Description of Non-Terminals

1. <main_program>: This is the scope where the whole program is going to be executed (like main method in Java or main function in C languages.

2. <program>: It contains all the statements which are needed to be executed to do needed tasks.

3. <statement>: A line of code which can be either *while, if, for, single_state or function_def.*

Ex: if(x == 5)

4. <statements>: A combination of statements with different types.

```
Ex: if(x == 5) {
      x = x - 10
        }
```

5. <while>: While statement is a control flow statement that allows code to be executed repeatedly based on a given condition. It takes expression in it and continues with left brace and statement in it. Finally finishes with the right brace.

```
Ex: while(x<5) {
      x=x + 1
  }
```

6. <if>: If statement is a programming conditional statement that, if proved true, performs a function or displays information. It's syntax is like this, if, left paranthesis and expression. After expression there will be right paranthesis and left brace. After left brace there will be some statements and right paranthesis or like this, if, left paranthesis and expression. After expression right paranthesis and left brace. After left brace there will be some statements,

right brace and else statment. After else statment there will be left brace and some statements and finally right brace.

```
Ex: if(x==1) {
    y = y+1
}
else {
    z = z+1
}
```

7 <for>: For loop is a control flow statement for specifying iteration, which allows code to be executed repeatedly. It's syntax is like this, for, left paranthesis and declaration in it and semi colon. After semi colon there is another expression and semi colon. After semi colon there will be an assignment operator and right paranthesis in order to finish for loop's parameters. Left brace will allow user to right plenty of statements and finally by using right brace user will finish whole for loop.

```
Ex: for (i = 1: i < 10: i = i + 1) {
    a = a * b
    }
```

8. <single_state>: Can be either assignment statement or declaration statement or return statement or function statement.

9. <function_type>: Can be either non void function or void function.

10. <assign_state>: Assignment is an operation which assigns the values. For example, x = 10, or x = 2 * 5 + 10.

11. <new_line>: New line provides user to pass to one or multiple lines below.

12. <declaration_state>: Declaration statement is the operation where one or more new variables declared. The syntax is like this, variable type and assignment statement or variable type and variable name.

Ex: int x

string y

float z

13. <expression> : Expressions are essential building blocks of any Java program, usually created to produce a new value, although sometimes an expression assigns a value to a variable. Expressions are built using values, variables, operators and method calls.

14. <low_op> : is used to execute addition or subtraction.

15. <high_op> : is used to execute multiplication or division.

16. <prop_op_high> : is used to execute negation

17. <prop_op_med> : is used to execute conjunction or disjunction

18. <prop_op_low> : is used to execute implication or double implication

19. <var_name> : is used to give a particular name for a variable which are used to store information to be referenced and manipulated

20. <var_names> : is the same as *var_name*, the only difference is, that it is used for multiple variables

21. <var_type> : an element that determines what type of data it comprises

22. <function_state> : is used to make a procedure for a particular purpose, in which a procedure is executed with data given as an input or *without* it

23. <return_state> It is used to exit from a method, with or without a value and it's syntax is like this, return, expression and newline.

24. <arguments> : the values actually supplied to the procedure when it is called.

25. <argument> : the value actually supplied to the procedure when it is called.

26. <integer>: Integer can be either with a sign plus or minus in front of it or just the number itself without signs.

Ex: +50, -50, 50.

27 <float>: It's a fundamental data type built into the compiler that's used to define numeric values with floatingdecimal points.

Ex: 0.057

28. <string>: A string is a sequence of characters that exist as an object of the class.

Ex: "Ali07"

29. <non-void function> : is a procedure with a given parameter or without parameter, which is used to do a particular execution with return type(s) value

30. <void function> : is a procedure with a given parameter or without parameter, with is used to do a particular execution without giving a return type(s) value

31. <parameters> : is used within a procedure, in which,its role is to receiving or passing data through,and its existence occurs within a scope of procedure

32. <comment> : is used to clarify what a line(s) of code does

33. <print> : is used to display the output of the desired expression

34. <println> : does the same execution as *print* does, the only difference is to move cursor to one line after displaying desired expression

35. <comment> : it is used to clarify what the code does in order to give an idea for code readers

36. <term>: Term is a part of an expression where the operations with a higher precedence are solved.

Ex: 6 + 1 * 4