SER 502 Project 1 Milestone (Team 1)

Authors – Sumeet Suryawanshi (ssuryaw5), Akash Rana (akrana1) , Rohan Mathur (rmathu19), Sadhanand Srinivasan (ssrin172)

Language Name – **SARS**

(The language name is the initials of all the authors)

Parser

a. Design language – Prolog

b. Data structure used – List

c. Parsing technique – Top-down parsing using unification.

d. Grammar used – Definite Clause Grammar (DCG)

Interpreter

a. Design language – Prolog

b. Data structure used – List data structure

Design

1. Data Types

a. Integer

b. Bool

c. Char

d. String

2. Identifiers

Can contain lowercase, uppercase letters, numbers, and special characters.

3. Variable Declaration

Variables can be initialized with the permitted data types. Some examples of the variable declarations are given below:

Eg 1: int x;

Eg 2: int y = 10;

4. Loops

The language will support 'for','forrange' and 'while' condition loop. The loops will run through the block of statements as long as the conditional statement remains true. If the condition becomes false, then the control will exit the loop.

5. Terminators

a. A program block in SARS starts with "Start" and concludes with the "End".

b. Every statement must end with a ";" (a semicolon).

6. Conditionals

a. The language supports "if-else" conditional construct.

b. The statement written within the "if" block is evaluated if the condition is true, otherwise the "elseif" statement is evaluated, if both are false, then the statements in the "else" block will get executed.

7. Operators

a. The language supports basic arithmetic operations such as addition ('+'), subtraction ('-'), multiplication ('*'), and division ('/').

b. The language supports basic arithmetic comparisons such as less than, greater than, equal to, and not equal to.

c. Associativity: All the operators are left associative, and the assignment operators are right associative.

d. Priority: The language gives higher priority to the multiplication and division

operators as compared to the addition and subtraction operators.

e.

8. Print

The language supports printing the identifier values. All the datatypes in the language are supported by the print statement.

Grammar

Terminal Rules

INT := /^[0-9]+$/

BOOL := /‘True’ | ‘False’/

STRING := /\”[\x00-\x7F]*\”/

CHAR := /'[\x00-\x7F]'/

FLOAT := /([0-9]*[.])?[0-9]+/

DATATYPE := ‘int’ | ‘bool’ | ‘string’ | ‘float’ | ‘char’

IDENTIFIER := /^[a-zA-Z_$][a-zA-Z_$0-9]*$/

ADD := ‘+’

SUB := ‘-’

MUL := ‘*’

DIV := ‘/’

AND := ‘and’

OR := ‘or’

NOT := ‘not’

ASSIGN := ‘=’

COMPARE := ‘<’ | ‘>’ | ‘>=’ | ‘<=’ | ‘!=’ | ‘==’

CONDITIONOPERATORS := ‘and’ | ‘or’ | ‘not’

IF := ‘if’

ELSE := ‘else’

ELSEIF := ‘elseif’

WHILE := ‘while’

FOR := ‘for’

IN := ‘in’

RANGE := ‘range’

STARTBLOCK := ‘{‘

ENDBLOCK := ‘}’

DELIMITER := ‘;’

COMMA := ‘,’

STARTQUOTE = ‘”

ENDQUOTE := ‘”

SPACE := ‘ ’

BEGIN := ‘begin’

END := ‘end’

OPENPAREN := ‘(‘

CLOSEPAREN := ‘)’

PRINT := ‘print’

Non-terminal rules

program := BEGIN statements DELIMITER END

statements := everystatement DELIMITER statements | statements

everystatement := print | declare | assign | ifelse | while | for

declare := DATATYPE SPACE IDENTIFIER ASSIGN data | DATATYPE SPACE IDENTIFIER

assign := IDENTIFIER ASSIGN expression

print := PRINT SPACE STARTQUOTE STRING ENDQUOTE | PRINT SPACE IDENTIFIER | PRINT STARTQUOTE STRING ENDQUOTE | PRINT IDENTIFIER

ifelse := IF OPENPAREN condition CLOSEPAREN STARTBLOCK statements DELIMITER ENDBLOCK | IF OPENPAREN condition CLOSEPAREN STARTBLOCK

statements DELIMITER ENDBLOCK DELIMITER, elseif DELIMITER ELSE

STARTBLOCK statements DELIMITER ENDBLOCK | IF OPENPAREN condition

CLOSEPAREN STARTBLOCK statements DELIMITER ENDBLOCK DELIMITER ELSE

OPENPAREN statements DELIMITER CLOSEPAREN

elseif := elseif1 DELIMITER elseif | elseif1

elseif1 := ELSEIF OPENPAREN condition CLOSEPAREN STARTBLOCK statements DELIMITER ENDBLOCK

while := WHILE OPENPAREN condition CLOSEPAREN STARTBLOCK statements ENDBLOCK

for := FOR forInRange STARTBLOCK statements ENDBLOCK

forInRange := OPENPAREN IDENTIFIER ASSIGN expression DELIMETER IDENTIFIER

COMPARE expression DELIMETER CLOSEPAREN | OPENPAREN IDENTIFIER

ASSIGN expression DELIMETER IDENTIFIER COMPARE expression DELIMETER expression

CLOSEPAREN | IDENTIFIER IN RANGE OPENPAREN expression COMMA expression CLOSEPAREN

condition := IDENTIFIER SPACE COMPARE SPACE expression | IDENTIFIER SPACE COMPARE expression CONDITIONOPERATORS condition | BOOL

ternary_expression := condition '?' expression ':' expression

expression := ternary_expression | term ADD expression | term SUB expression | term

term := factor MUL term | factor DIV term | factor

factor := OPENPAREN expression CLOSEPAREN | data | IDENTIFIER

data := INT | BOOL | STRING | FLOAT | CHAR