SER502 Project

NovelC Demonstration

Team - 9

Abhinaw Sarang (1217265205)

Sagar Khar (1217125416)

Saksham Jhawar (1217883758)

Smit Dharmeshumar Shah (1217106696)

Overview of Demonstration

- Language Introduction
- Features
- Tools used
- High Level Design
- Tool Installation
- Language Grammar – Tokenization and Parsing
- Interpreter (Evaluator)
- Sample run

Language Introduction

- Our language is inspired from existing languages such as Python and Java.
- Easy to code, because of familiar keywords used.
- Grammar is written in Python whereas Evaluator is written in SWI-Prolog.
- The name novelC has been chosen considering the situation this language was developed in.

Features

- Data types
- Integer (int)
- String (string)
- Boolean (bool)

- Logical Operators (Additional features)
- Equals / Not Equals
- Greater than / less than
- Greater than equals / less than equals
- String Equal
- String Concatenation

- Mathematical Operators
- Addition
- Subtraction
- Multiplication
- Division
- Parenthesis (Additional feature)

- Boolean Operators
- And
- Or
- Not
- Ternary

Features (Continued)

- Decision Constructs
- If-else
- If-elseif-else (Additional feature)

- Loop Constructs
- Traditional For loop
- While loop
- Do While loop (Additional feature)
- For in range loop

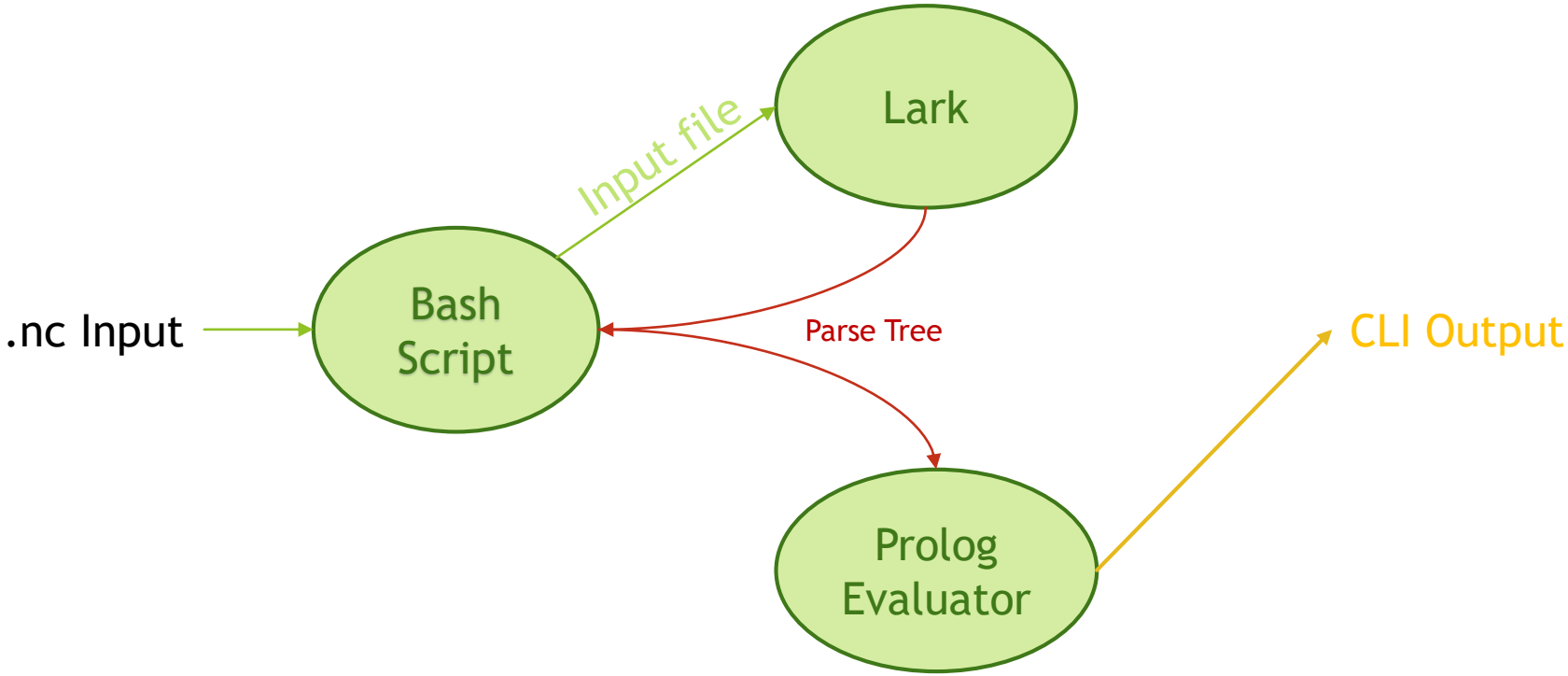
Tools used

- Tokenization and Parse tree generation using Python Lark
- Evaluation of Parse tree in SWI - Prolog
- Bash script for single point execution

Tool Installation

- Prerequisite: Python3+ ,SWI-Prolog
- Pip install lark-parser

High Level Design



Grammar

```
program: block PEND

block: (declarations)? commands

declarations : (declaration SEMI)+

declaration: INT I ASSIGN N
           | STRING I ASSIGN S
           | BOOL I ASSIGN TRUE
           | BOOL I ASSIGN FALSE
           | INT I
           | STRING I
           | BOOL I

commands : (command)+

command: IF OBRAC boolean CBRAC OCURL commands CCURL elseif ELSE OCURL commands CCURL SEMI
       | WHILE OBRAC boolean CBRAC OCURL commands CCURL SEMI
       | DO OCURL commands CCURL WHILE OBRAC boolean CBRAC SEMI
       | FOR OBRAC INT I ASSIGN N SEMI boolean SEMI updateexp CBRAC OCURL commands CCURL SEMI
       | FOR I IN RANGE OBRAC N COMMA N CBRAC OCURL commands CCURL SEMI
       | I ASSIGN exp SEMI
       | I ASSIGN ter SEMI
       | PRINT values SEMI
       | declarations

elseif :  (ELIF OBRAC boolean CBRAC OCURL commands CCURL)*

boolean : TRUE | FALSE
        | mathexp EQUALS mathexp
        | mathexp NOTEQUALS mathexp
        | mathexp LT mathexp
        | mathexp LTE mathexp
        | mathexp GT mathexp
        | mathexp GTE mathexp
        | stringexp EQUALS stringexp
        | boolexp

updateexp : I ASSIGN mathexp
          | I DPLUS
          | I DMINUS

mathexp : mathexp ADD mathexp
        | mathexp SUB mathexp
        | mathexp MUL mathexp
        | mathexp DIV mathexp
        | OBRAC mathexp CBRAC
        | identifier | number

stringexp : stringexp ADD stringexp | S

boolexp : boolean AND boolean
        | boolean OR boolean
        | NOT boolean
        | OBRAC boolean CBRAC

exp : mathexp | stringexp | boolexp

ter : boolean TIF exp TELSE exp

values : identifier | number | str | boolean

identifier: I

number: N

str:  S

SPACE : " "
BEND : "BEnd"
PEND : "End"
SEMI : ";"
COMMA : ","
INT     : "int"
ASSIGN  : "="
DPLUS    : "++"
DMINUS    : "--"
EQUALS    : "=="
NOTEQUALS    : "!="
LT     : "<"
LTE     : "<="
GT     : ">"
GTE     : ">="
ADD     : "+"
SUB     : "-"
MUL     : "*"
DIV     : "/"
AND     : "and"
OR     : "or"
NOT     : "not"

STRING  : "string"
BOOL    : "bool"
IF      : "if"
ELSE    : "else"
ELIF    : "elif"
WHILE    : "while"
DO        : "do"
FOR       : "for"
IN        : "in"
RANGE    : "range"
OBRAC    : "("
CBRAC    : ")"
OCURL    : "{"
CCURL    : "}"
PRINT    : "print"
TIF    : "?"
TELSE    : ":"
TRUE    : "true"
FALSE    : "false"
WHITE : /\s+(?=([^\"]*[\"][^\"]*[\"])*[^\"]*$)/

%import common.ESCAPED_STRING -> S
%import common.SIGNED_NUMBER -> N
%import common.WORD -> I

%ignore WHITE
```

Parse Tree Generation

```
tree(program, [
    tree(block, [
        tree(commands, [
            tree(command, [
                token(PRINT, 'print'),
                tree(values, [
                    tree(str, [
                        token(S, '"Hello World!"')
                        ])
                    ]),
                    token(SEMI, ';')
                ])
            ])
        ]),
        token(PEND, 'End')
    ])
```

Program Snapshots

```
int n = 5;
int m = 5;
if (n < m) {
        print "n is less than m.";
}
elif (n == m) {
        print "n is equal to m.";
}
else {
        print "n is greater than m.";
};
End
```

```
(base) Abhinaws-MacBook-Pro:src sarang$ sh novelC.sh ../data/compare2num.nc
Compiling...

Compilation successful!

Interpreting...

"n is equal to m."

Done!
(base) Abhinaws-MacBook-Pro:src sarang$
```

```
int x = 7;
int i = 7;
int factorail = 1;
while(i >= 1) {
  factorail = factorail*i;
  i=i-1;
};
print "Factorial of 7 is:";
print factorail;
End
```

```
(base) Abhinaws-MacBook-Pro:src sarang$
(base) Abhinaws-MacBook-Pro:src sarang$ sh novelC.sh ../data/factorial.nc
Compiling...

Compilation successful!

Interpreting...

"Factorial of 7 is:"
5040

Done!
(base) Abhinaws-MacBook-Pro:src sarang$
```

Program Snapshots (Continued)

```
1   bool bOne = false;
2   bool bTwo = false;
3   int x =1;
4   int y =2;
5   bTwo = (x==y) or true;
6   print bTwo;
7
8
9   int bThree = 0;
10  x = 2;
11  bThree = (x==y) ? 1 : 0;
12  print bThree;
13
14
15  int a = 10;
16  int b = 20;
17  int c = 0;
18  int d = 0;
19  int e = 0;
20  c = a + b;
21  d = a*b;
22  e = b/a;
23
24  print c;
25  print d;
26  print e;
27
28  string vv = "502 Project";
29  if(true) {
30          print vv;
31  } else {
32    print "inside else";
33  };
34
35  End
```

```
[(base) Abhinaws-MacBook-Pro:src sarang$
[(base) Abhinaws-MacBook-Pro:src sarang$ sh novelC.sh ../data/constraints13.nc
Compiling...

Compilation successful!

Interpreting...

true
1
30
200
2
"502 Project"

Done!
[(base) Abhinaws-MacBook-Pro:src sarang$
[(base) Abhinaws-MacBook-Pro:src sarang$
```

Program Snapshots (Continued)

```
int x = 0;
int y = 1;
int z = 0;
print "Fibonacci series till 10th term";
print x;
print y;
for(int i = 0; i < 8 ; i++) {
  z = x + y;
  print z;
  y = x;
  x = z;
};

End
```

```
(base) Abhinaws-MacBook-Pro:src sarang$ sh novelC.sh ../data/fibonacci.nc
Compiling...

Compilation successful!

Interpreting...

"Fibonacci series till 10th term"
0
1
1
1
2
3
5
8
13
21

Done!
(base) Abhinaws-MacBook-Pro:src sarang$
```

```
int x = 20;
print "Numbers are:";
for i in range(1,21) {
  print i;
};

int result = 0;
do {
  result = result + x;
  x = x - 1;
} while (x != 0);
print "Sum is:";
print result;
End
```

```
(base) Abhinaws-MacBook-Pro:src sarang$ sh novelC.sh ../data/sum20num.nc
Compiling...

Compilation successful!

Interpreting...

"Numbers are:"
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
"Sum is:"
210

Done!
(base) Abhinaws-MacBook-Pro:src sarang$
```

Thank you and Stay Safe!

SER502-Spring2020-Team9