ALPI_SER502 - Team 26

Ankit Vutukuri
Jubanjan Dhar
Harmish Ganatra
Sudhanva Hebbale

Introduction

- ALPI -> Another Lexer Parser Interpreter

- Why ALPI?
  - Simple syntax
  - Advanced features
  - Easy debugging

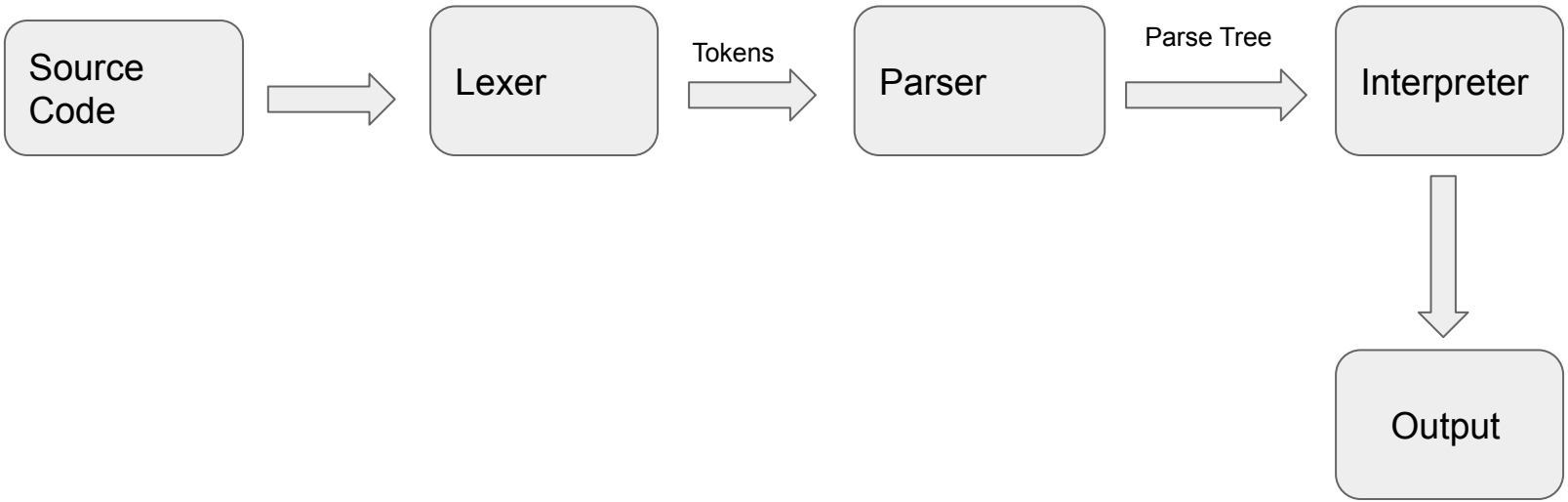
Basic Features

1. Primitive Data Types - Number, String, Boolean
2. Loops - For(traditional and range), While
3. Conditional Statements - If, If-then-else, Ternary
4. Boolean Operations - &, ||, >, <, ==, !
5. Assignment Operator with Identifiers
6. Operators - +,-,*,/,%,++,--
7. Comments

Additional Features

1. Lists
2. Dictionary
3. Functions
4. String Operations
    a. Concat
    b. Length
    c. Reverse
5. Simplified and easy to use Print Statement

Project Flowchart



Lexer

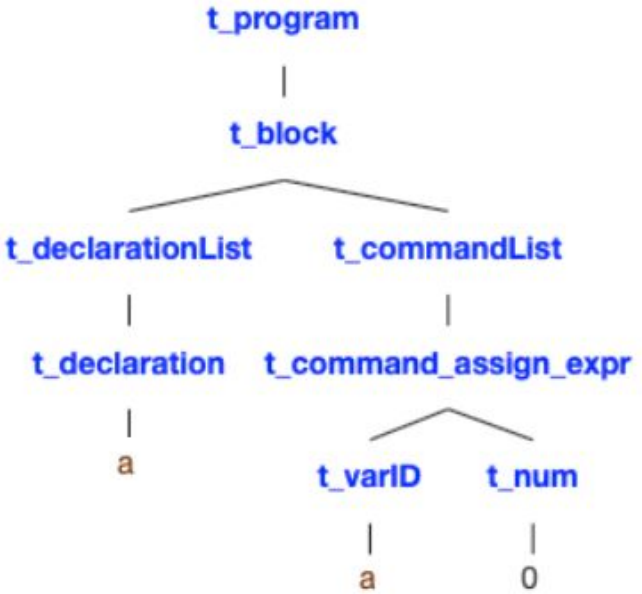
- The source code is read into the parser and checked for comments and removed using regular expressions.
- The processed data is then passed through the NLTK tokenizer object and we get subsequent tokens.
- Tokens are further processed to handle some cases (eg. i +1, i+ 1, ` ‘ etc).
- Lastly, all tokens are checked for integers and floats and then pushed to a text file for the parser to process.

Parser

- The tokens generated from the lexer is fed as the input to the parser.
- The grammar is written as a set of definite clauses in Definite Clause Grammar(DCG) format.
- Implemented in SWI-Prolog version 8.1
- The grammar rules are modified to generate tree nodes.
- The program matches the grammar rules with the source tokens, generating the parse tree.

Sample Parse Tree

[begin, num, a, a, =, 0, end],

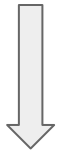


Interpreter

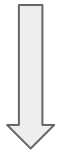
- The parse tree generated from the parser is given to the interpreter as input.
- Consists of predicates that executes every command in the source code.
- These predicates take their respective tree nodes as input to perform the task.
- Uses a list of tuples to implement the environment for the source code variables.

Parser --> Interpreter.

begin, num, a, =, 0, print, a, end]



T = *t_program*(*t_block*(*t_declarationList*(*t_init_expr*(a, *t_num*(0))),
t_commandList(*t_command_print*(*t_print_var*(*t_varID*(a)))))



```
a = 0
```

Sample Program 1

```
begin
        num x = 2
        num y = 3
        print x, y
end
```

```
?- main(P).
[begin,num,x,=,2,num,y,=,3,print,x,',',y,end]
x = 2
y = 3
P = t_program(t_block(t_declarationList(t_init_expr(x, t_num(2)), t_declaration
List(t_init_expr(y, t_num(3)))), t_commandList(t_command_print(t_print_var(t_va
rID(x), t_print_var(t_varID(y)))))))
```

Sample Program 2

```
begin
        num x = 8
        num y = 4
        num z
        num u
        num v
        num w
        num s
        num t
        z = x + y - 2 * x
        u = x - ( y / 2 + 6 * x )
        v = x * y * ( 2 / x * y )
        w = x / y * 4 - u + ( y % 2 )
        print x, y, z, u, v, w
end
```

```
?- main(P).
[begin,num,x,=,8,num,y,=,4,num,z,num,u,num,v,num,w,num,s,num,t,z,=,x,+,y,-,2,*,x,u,=,x,-,'(',y,/,2,+,6,*,x,
')',v,=,x,*,y,*,'(',2,/,x,*,y,')',w,=,x,/,y,*,4,-,u,+,'(',y,'%',2,')',print,x,',',y,',',z,',',u,',',v,',',w
,end]
x = 8
y = 4
z = -4
u = -42
v = 32.0
w = 50
P = t_program(t_block(t_declarationList(t_init_expr(x, t_num(8)), t_declarationList(t_init_expr(y, t_num(4)
), t_declarationList(t_declaration(z), t_declarationList(t_declaration(u), t_declarationList(t_declaration(
v), t_declarationList(t_declaration(w), t_declarationList(t_declaration(...), t_declarationList(...)))))))
, t_commandList(t_command_assign_expr(t_varID(z), t_subr(t_add(t_id(t_varID(x)), t_id(t_varID(y))), t_mult(
t_num(2), t_id(t_varID(x))))), t_commandList(t_command_assign_expr(t_varID(u), t_subr(t_id(t_varID(x)), t_b
rackets(t_add(t_div(t_id(...), t_num(...)), t_mult(t_num(...), t_id(...)))))), t_commandList(t_command_assi
gn_expr(t_varID(v), t_mult(t_mult(t_id(t_varID(...)), t_id(t_varID(...))), t_brackets(t_mult(t_div(..., ...
), t_id(...))))), t_commandList(t_command_assign_expr(t_varID(w), t_add(t_subr(t_mult(..., ...), t_id(...))
, t_brackets(t_mod(..., ...)))), t_commandList(t_command_print(t_print_var(t_varID(...), t_print_var(..., .
..)))))))))) ;
false.
```

Sample Program 3

```
begin
        num x = 0
        num u = 1
        num z = 1
        num y = 6
        num w = 1024
        for (i = 0 ; i < 10 ; i++){
                w = w / 2
        }
        print w

        for j in range(1, 10){
                x = x + 1
                y = y - 1
                z ++
        }
        print x, y, z
        while(not u == 3){
                u ++
        }
        print u
end
```

```
[?- main(P).
[begin,num,x,=,0,num,u,=,1,num,z,=,1,num,y,=,6,num,w,=,1024,for,'(',i,=,0,;,i,<,10,;,i,++,')','{',w,=,w,/,2
,'}',print,w,for,j,in,range,'(',1,',',10,')','{',x,=,x,+,1,y,=,y,-,1,z,++,'}',print,x,',',y,',',z,while,'('
,not,u,==,3,')','{',u,++,'}',print,u,end]
w = 1
x = 9
y = -3
z = 10
u = 3
P = t_program(t_block(t_declarationList(t_init_expr(x, t_num(0)), t_declarationList(t_init_expr(u, t_num(1)
), t_declarationList(t_init_expr(z, t_num(1)), t_declarationList(t_init_expr(y, t_num(6)), t_declarationLis
t(t_init_expr(w, t_num(1024)))))), t_commandList(t_command_for_inc(t_id(t_varID(i)), t_num(0), t_boolean_l
t(t_id(t_varID(i)), t_num(10)), t_increment(t_varID(i)), t_commandList(t_command_assign_expr(t_varID(w), t_
div(t_id(t_varID(w)), t_num(2))))), t_commandList(t_command_print(t_print_var(t_varID(w))), t_commandList(t
_command_for_range(t_id(t_varID(j)), t_num(1), t_num(10), t_commandList(t_command_assign_expr(t_varID(x), t
_add(t_id(...), t_num(...))), t_commandList(t_command_assign_expr(t_varID(...), t_subr(..., ...)), t_comman
dList(t_command_increment(...))))), t_commandList(t_command_print(t_print_var(t_varID(x), t_print_var(t_var
ID(...), t_print_var(...)))), t_commandList(t_command_while(t_boolean_not(t_boolean_equal(..., ...)), t_com
mandList(t_command_increment(...))), t_commandList(t_command_print(t_print_var(...)))))))))
```

Sample Program 4

```
begin
        num x = 2
        num y = 3
        num z
        num u
        func mod(x, y){
                num z = 0
                x = x % y
                print x, y, z
                return x
        }

        z = mod(20, 10)
        u = mod(4, 5)
        print z, u
end
```

```
?- main(P).
[begin,num,x,=,2,num,y,=,3,num,z,num,u,func,mod,'(',x,',',y,')','{',num,z,=,0,x,=,x,'%',y,p
rint,x,',',y,',',z,return,x,'}',z,=,mod,'(',20,',',10,')',u,=,mod,'(',4,',',5,')',print,z,'
',',u,end]
x = 0
y = 10
z = 0
x = 4
y = 5
z = 0
z = 0
u = 4
P = t_program(t_block(t_declarationList(t_init_expr(x, t_num(2)), t_declarationList(t_init_
expr(y, t_num(3)), t_declarationList(t_declaration(z), t_declarationList(t_declaration(u),
t_declarationList(t_funcDeclaration(t_funcDeclr(t_varID(...), t_parList(..., ...), t_declar
ationList(...), t_commandList(..., ...), t_return(...)))))))), t_commandList(t_command_func
Return(t_varID(z), t_funCall(t_id(t_varID(mod)), t_callParList(t_num(20), t_callParList(t_n
um(10))))), t_commandList(t_command_funcReturn(t_varID(u), t_funCall(t_id(t_varID(mod)), t_
callParList(t_num(4), t_callParList(t_num(5))))), t_commandList(t_command_print(t_print_var
(t_varID(z), t_print_var(t_varID(u)))))))))
```