PES University, RR Campus Bangalore 560085

Automata Formal Language and Logic

UE23CS242A

Mini Project

3rd Semester, Academic Year 2024

Date: 15/11/2024

To describe and write Grammar for the constraints of a programming language

Language : R

Constrains: if-else condition, for loop, while loop

Name: **Ankita Muni** SRN: **PES1UG23CS081** Sec: **'B'**

Lexer Program:

r_lexer.py > ...

```python
import ply.lex as lex

tokens = (
    'NUMBER',
    'ID',
    'PLUS', 'MINUS', 'TIMES', 'DIVIDE', 'MODULO', 'POWER',
    'LPAREN', 'RPAREN', 'LBRACE', 'RBRACE',
    'LT', 'GT', 'LTE', 'GTE', 'EQ', 'NEQ',
    'ASSIGN',
    'SEMICOLON',
    'IF', 'ELSE', 'WHILE', 'FOR', 'IN',
    'PRINT',
    'STRING',
    'COLON'
)

# Regular expression rules for simple tokens
t_PLUS    = r'\+'
t_MINUS   = r'-'
t_TIMES   = r'\*'
t_DIVIDE  = r'/'
t_MODULO  = r'%'
t_POWER   = r'\^'
t_LPAREN  = r'\('
t_RPAREN  = r'\)'
t_LBRACE  = r'\{'
t_RBRACE  = r'\}'
t_LT      = r'<'
t_GT      = r'>'
t_LTE     = r'<='
t_GTE     = r'>='
t_EQ      = r'=='
t_NEQ     = r'!='
t_ASSIGN  = r'='
t_SEMICOLON = r';'
t_COLON   = r':'
t_IN = r'IN'
```

```python
def t_NUMBER(t):
    r'\d+'
    t.value = int(t.value)
    return t

def t_ID(t):
    r'[a-zA-Z_][a-zA-Z_0-9]*'
    t.type = reserved.get(t.value, 'ID')
    return t

def t_STRING(t):
    r'\".*?\"'
    t.value = t.value[1:-1]
    return t

def t_newline(t):
    r'\n+'
    t.lexer.lineno += len(t.value)

t_ignore  = ' \t'

def t_error(t):
    print(f"Illegal character '{t.value[0]}' at line {t.lexer.lineno}")
    t.lexer.skip(1)

lexer = lex.lex()

reserved = {
    'if': 'IF',
    'else': 'ELSE',
    'while': 'WHILE',
    'for': 'FOR',
    'in': 'IN',
    'print': 'PRINT'
}
```

Parser Program:

r_parser.py > p_program

```python
import ply.yacc as yacc
from r_lexer import tokens

def p_program(p):
    '''program : statement_list'''
    p[0] = p[1]

def p_statement_list(p):
    '''statement_list : statement
                      | statement_list statement'''
    if len(p) == 2:
        p[0] = [p[1]] if p[1] is not None else []
    else:
        p[0] = p[1] + [p[2]] if p[2] is not None else p[1]

def p_statement(p):
    '''statement : if_statement
                 | for_statement
                 | while_statement
                 | expression
                 | assignment_statement
                 | statement SEMICOLON'''
    if len(p) == 2:
        p[0] = p[1]
    else:
        p[0] = p[1]

def p_if_statement(p):
    '''if_statement : IF LPAREN expression RPAREN LBRACE statement_list RBRACE
                    | IF LPAREN expression RPAREN LBRACE statement_list RBRACE ELSE LBRACE statement_list RBRACE'''
    if len(p) == 8:
        p[0] = {'type': 'if', 'condition': p[3], 'body': p[6]}
    else:
        p[0] = {'type': 'if', 'condition': p[3], 'body': p[6], 'else_body': p[10]}
```

```python
def p_for_statement(p):
    '''for_statement : FOR LPAREN ID IN expression COLON expression RPAREN LBRACE statement_list RBRACE'''
    p[0] = {'type': 'for', 'variable': p[3], 'start': p[5], 'end': p[7], 'body': p[10]}

def p_while_statement(p):
    '''while_statement : WHILE LPAREN expression RPAREN LBRACE statement_list RBRACE'''
    p[0] = {'type': 'while', 'condition': p[3], 'body': p[6]}

def p_assignment_statement(p):
    '''assignment_statement : ID ASSIGN expression'''
    p[0] = {'type': 'assignment', 'variable': p[1], 'value': p[3]}

def p_expression(p):
    '''expression : term
                  | expression PLUS term
                  | expression MINUS term
                  | expression TIMES term
                  | expression DIVIDE term
                  | expression MODULO term
                  | expression POWER term
                  | expression EQ term
                  | expression NEQ term
                  | expression LT term
                  | expression GT term
                  | expression LTE term
                  | expression GTE term
                  | PRINT LPAREN expression RPAREN'''
    if len(p) == 2:
        p[0] = p[1]
    elif p[1] == 'print':
        p[0] = {'type': 'print', 'expression': p[3]}
    else:
        p[0] = {'type': 'binary_op', 'op': p[2], 'left': p[1], 'right': p[3]}

```

```python
def p_term(p):
    '''term : factor
            | term TIMES factor
            | term DIVIDE factor
            | term MODULO factor
            | term POWER factor'''
    if len(p) == 2:
        p[0] = p[1]
    else:
        p[0] = {'type': 'binary_op', 'op': p[2], 'left': p[1], 'right': p[3]}

def p_factor(p):
    '''factor : NUMBER
              | STRING
              | ID
              | LPAREN expression RPAREN'''
    p[0] = p[1]

def p_error(p):
    if p:
        raise SyntaxError(f"Syntax error at line {p.lineno}, position {p.lexpos}: Unexpected token '{p.value}'")
    else:
        raise SyntaxError("Syntax error: Unexpected end of input")

parser = yacc.yacc()

def parse(data):
    result = parser.parse(data)
    if result is None:
        raise SyntaxError("Invalid syntax")
    return result

__all__ = ['parse']
```

Main Program:

main.py > parse_file

```python
from r_parser import parse
import sys

def parse_file(filename):
    with open(filename, 'r') as f:
        data = f.read()
    return parse(data)

if __name__ == '__main__':
    try:
        result = parse_file('input.txt')
        print("Input syntax is correct")
        print("Parse result:")
        print(result)
    except SyntaxError as e:
        print("Input syntax is incorrect")
        print(f"Error: {str(e)}")
        sys.exit(1)

```

Input 1: (if-else condition without error)

input.txt

```
if (a > 0)
{
    if (a < 10)
    {
        a = a + 1;
    }
    else
    {
        a = a - 1;
    }
}

```

Output 1:

```
PS C:\Users\Admin\Desktop\SEMESTER 3\AUTOMATA FORMAL LANGUAGES AND LOGIC\r> python main.py
Input syntax is correct
Parse result:
[{'type': 'if', 'condition': {'type': 'binary_op', 'op': '>', 'left': 'a', 'right': 0}, 'body': [{'type': 'if', '
condition': {'type': 'binary_op', 'op': '<', 'left': 'a', 'right': 10}, 'body': [{'type': 'assignment', 'variable
': 'a', 'value': {'type': 'binary_op', 'op': '+', 'left': 'a', 'right': 1}}], 'else_body': [{'type': 'assignment'
, 'variable': 'a', 'value': {'type': 'binary_op', 'op': '-', 'left': 'a', 'right': 1}}]}]}]
PS C:\Users\Admin\Desktop\SEMESTER 3\AUTOMATA FORMAL LANGUAGES AND LOGIC\r>
```

Input 2: (if-else condition with error)

input.txt

```
if (a > 0)
{
    if (a < 10)
    {
        a = a + 1;

    else
    {
        a = a - 1;
    }
}

```

Output 2:

```
PS C:\Users\Admin\Desktop\SEMESTER 3\AUTOMATA FORMAL LANGUAGES AND LOGIC\r> python main.py
Input syntax is incorrect
Error: Syntax error at line 7, position 68: Unexpected token 'else'
PS C:\Users\Admin\Desktop\SEMESTER 3\AUTOMATA FORMAL LANGUAGES AND LOGIC\r>
```

Input 3: (while loop without error)

input.txt

```
x = 0;
y = 0;
while (x < 10)
{
    x = x + 1;
    y = y + x;
}

```

Output 3:

```
PS C:\Users\Admin\Desktop\SEMESTER 3\AUTOMATA FORMAL LANGUAGES AND LOGIC\r> python main.py
Input syntax is correct
Parse result:
[{'type': 'assignment', 'variable': 'x', 'value': 0}, {'type': 'assignment', 'variable': 'y', 'value': 0}, {'type
': 'while', 'condition': {'type': 'binary_op', 'op': '<', 'left': 'x', 'right': 10}, 'body': [{'type': 'assignmen
t', 'variable': 'x', 'value': {'type': 'binary_op', 'op': '+', 'left': 'x', 'right': 1}}, {'type': 'assignment',
'variable': 'y', 'value': {'type': 'binary_op', 'op': '+', 'left': 'y', 'right': 'x'}}]}]
PS C:\Users\Admin\Desktop\SEMESTER 3\AUTOMATA FORMAL LANGUAGES AND LOGIC\r>
```

Input 4: (while loop with error)

input.txt

```
x = 0;
y = 0;
while (x  10)
{
    x = x + 1;
    y = y + x;
}

```

Output 4:

```
PS C:\Users\Admin\Desktop\SEMESTER 3\AUTOMATA FORMAL LANGUAGES AND LOGIC\r> python main.py
Input syntax is incorrect
Error: Syntax error at line 3, position 26: Unexpected token '10'
PS C:\Users\Admin\Desktop\SEMESTER 3\AUTOMATA FORMAL LANGUAGES AND LOGIC\r>
```

Input 5: (for loop without error)

input.txt

```
for (i in 1:5)
{
    print(i)
}

```

Output 5:

```
PS C:\Users\Admin\Desktop\SEMESTER 3\AUTOMATA FORMAL LANGUAGES AND LOGIC\r> python main.py
Input syntax is correct
Parse result:
[{'type': 'for', 'variable': 'i', 'start': 1, 'end': 5, 'body': [{'type': 'print', 'expression': 'i'}]}]
PS C:\Users\Admin\Desktop\SEMESTER 3\AUTOMATA FORMAL LANGUAGES AND LOGIC\r>
```

Input 6: (for loop without error)

input.txt

```
for (i i 1:5)
{
    print(i)
}

```

Output 6:

```
PS C:\Users\Admin\Desktop\SEMESTER 3\AUTOMATA FORMAL LANGUAGES AND LOGIC\r> python main.py
Input syntax is incorrect
Error: Syntax error at line 1, position 7: Unexpected token 'i'
PS C:\Users\Admin\Desktop\SEMESTER 3\AUTOMATA FORMAL LANGUAGES AND LOGIC\r>
```
