LAPORAN TUGAS BESAR TEORI BAHASA DAN AUTOMATA

Lexical Analyzer dan Parser Sederhana untuk Perulangan “for” pada Golang

Kelompok 13

Laporan di susun oleh:
Nandika Abiyoga Santosa - 1301213421
Adrian Yoris Mbake Woka - 130121084
Bagas Mukti Wibowo - 1301213249

Program Studi Informatika
Fakultas Informatika
Universitas Telkom 2023

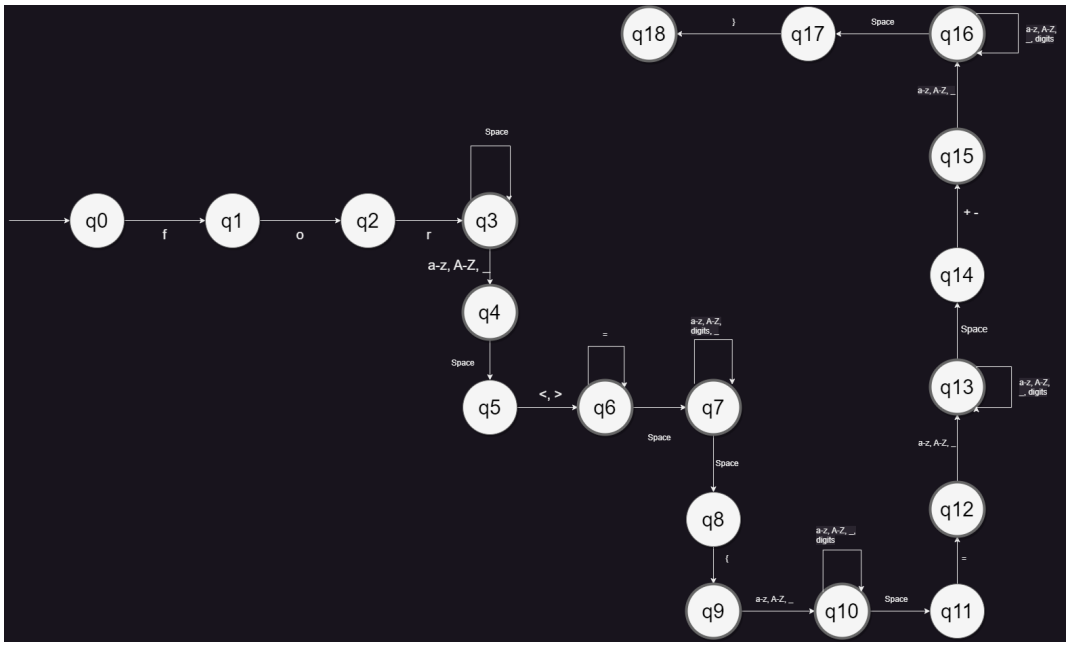
A. Context Free Grammar

Pada permasalahan kali ini kelompok kami mendapatkan tugas berupa cara mendefinisikan sebuah context free grammar (CFG) yang mempersentasikan aturan bahasa sederhana untuk bahasa manusia.Kemudian bedasarkan CFG yang sudah didefinisikan, selanjutnya mahasiswa diminta untuk membuat sebuah program sederhana yaitu Lexcial Analyzer untuk (mendefinikan sebuah lexcial/token/kata valid sesuai symbol terminal yang didefinisikan) dan parser untuk (apakah susunan token atau kata sudah memenuhi aturan dari grammar). Berikut adalah CFG yang telah kami bentuk:

<statement> : for <condition> { <action> }
<condition> : <variable> <comparation> <variable>
<expression>: <variable> <operator> <variable>
<condition> : true | false
<action> : <variable> "=" <expression>
<variable> : [i, j]
<comparation> : < | > | <= | >= | == | !=
<operator>: + | - | * | /

B. Rancangan Finite Automata

Untuk selanjutnya adalah merupakan masalah dari tugas besar ini yaitu Rancangan Finite Automata. Berikut adalah rancangan automata yang telah kelompok kami buat :



C. PARSER TABEL LL

Setelah kami merancang Finite Automata dan menentukan CFG (Context Free Grammar),permasalahan selanjut nya yaitu membuat Parse Tabel. Berikut adalah Parse tabel yang telah kami buat dengan ketentuan sebagai berikut :

<statement> : for <condition> { <action> }
<condition> : <variable> <comparation> <variable>
<expression>: <variable> <operator> <variable>
<condition> : true | false
<action> : <variable> "=" <expression>
<variable> : [i, j]
<comparation> : < | > | <= | >= | == | !=
<operator>: + | - | * | /

	for	true	false	i	j	<	>	<=	>=	!=	==	{	+	-	*	/	}	EOS
STATEMENT	for <condition> { <action> }	error	error	error	error	error	error	error	error	error	error	error	error	error	error	error	error	error
CONDITION	error	true	false	<variable> <comparation> <variable>	<variable> <comparation> <variable>	error	error	error	error	error	error	error	error	error	error	error	error	error
ACTION	error	error	error	<variable> "=" <expression>	<variable> "=" <expression>	error	error	error	error	error	error	error	error	error	error	error	error	error
VARIABLE	error	error	error	i	j	error	error	error	error	error	error	error	error	error	error	error	error	error
OPERATOR	error	error	error	error	error	error	error	error	error	error	error	error	+	-	*	/	error	error
COMPARATION	error	error	error	error	error	<	>	<=	>=	!=	==	error	error	error	error	error	error	error
EXPRESSION	error	error	error	<variable> <operator> <variable>	<variable> <operator> <variable>	error	error	error	error	error	error	error	error	error	error	error	error	Error

D. Program

a. Lexcial Analyzer

Code program Lexcial Analyzer digunakan untuk menguji setiap kata yang akan dimasukan adalah merupakan kata yang valid atau tidak. Ada Juga token yang kami uji kali ini yaitu repretasi dasar dari for yang ada pada program Golang, Berikut hasil yang telah kami buat beserta pengujiannya.

```python
import string

string.whitespace = " \n"

class LexicalAnalyzer:
    def __init__(self):
        self.transitions = {}
        self.accepted_states = set()
        self.init_state = None
        self.current_state = None
        self.current_token = ''

    def add_init_state(self, state):
        self.init_state = state

    def add_transition(self, state, read, target_state):
        for char in read:
            self.transitions[(state, char)] = target_state

    def add_accepted_state(self, state):
        self.accepted_states.add(state)

    def analyze(self, input_str, verbose=False):
        if not input_str.endswith('#'):
            input_str = input_str + '#'

        cursor_pos = 0
        self.current_state = self.init_state
        self.current_token = ''
```

```python

        while cursor_pos < len(input_str):
            is_accepted = self.current_state in self.accepted_states

            if verbose:
                print({
                    'current_state': self.current_state,
                    'current_token': self.current_token,
                    'input': input_str[cursor_pos]
                })

            if is_accepted and input_str[cursor_pos] in string.whitespace + '#':
                print(f'Current Token: {self.current_token}')
                self.current_token = ''

            if input_str[cursor_pos] not in string.whitespace:
                self.current_token += input_str[cursor_pos]

            self.current_state = self.transitions.get((self.current_state, input_str[cursor_pos]))

            if not self.current_state:
                print(f'Invalid Token: {input_str[cursor_pos]}')
                break

            cursor_pos += 1

        if self.current_state == 'accept':
            print("Your code is valid!")


```

```
60  def main():
61      lexical = LexicalAnalyzer()
62
63      # add init state
64      lexical.add_init_state('q0')
65
66      # handle 'for' statement
67      lexical.add_transition('q0', 'f', 'q1')
68      lexical.add_transition('q1', 'o', 'q2')
69      lexical.add_transition('q2', 'r', 'q3')
70
71      # space or tab after 'for' statement
72      lexical.add_transition('q3', string.whitespace, 'q3')
73
74      lexical.add_accepted_state('q3')
75
76      # handle variable name
77      lexical.add_transition('q3', string.ascii_letters, 'q4')
78      lexical.add_transition('q4', string.ascii_letters + string.digits + '_', 'q4')
79
80      # space or tab after variable name
81      lexical.add_transition('q4', string.whitespace, 'q5')
82
83      lexical.add_accepted_state('q4')
84
85      # handle '<>=!' operator
86      lexical.add_transition('q5', '<>=!', 'q6')
87      lexical.add_transition('q6', '=', 'q6')
88      lexical.add_transition('q6', string.whitespace, 'q7')
```

```
89
90      lexical.add_accepted_state('q6')
91
92      # handle variable name
93      lexical.add_transition('q6', string.ascii_letters, 'q7')
94      lexical.add_transition('q7', string.ascii_letters + string.digits + '_', 'q7')
95
96      # space or tab after variable name
97      lexical.add_transition('q7', string.whitespace, 'q8')
98      lexical.add_accepted_state('q7')
99
100     # handle '{' symbol
101     lexical.add_transition('q8', '{', 'q9')
102     lexical.add_transition('q9', string.whitespace, 'q10')
103
104     lexical.add_accepted_state('q9')
105
106     lexical.add_transition('q9', string.ascii_letters + '_', 'q10')
107     lexical.add_transition('q10', string.ascii_letters + string.digits + '_', 'q10')
108     lexical.add_transition('q10', string.whitespace, 'q11')
109
110     lexical.add_accepted_state('q10')
111
112     lexical.add_transition('q11', '=', 'q12')
113     lexical.add_transition('q12', string.whitespace, 'q13')
114
115     lexical.add_accepted_state('q12')
116
117     lexical.add_transition('q12', string.ascii_letters + '_', 'q13')
```

```
    lexical.add_transition('q13', string.ascii_letters + string.digits + '_', 'q13')
    lexical.add_transition('q13', string.whitespace, 'q14')

    lexical.add_accepted_state('q13')

    lexical.add_transition('q14', '+-', 'q15')
    lexical.add_transition('q15', string.whitespace, 'q16')

    lexical.add_accepted_state('q15')

    lexical.add_transition('q15', string.ascii_letters + '_', 'q16')
    lexical.add_transition('q16', string.ascii_letters + string.digits + '_', 'q16')
    lexical.add_transition('q16', string.whitespace, 'q17')

    lexical.add_accepted_state('q16')

    lexical.add_transition('q17', '}', 'q18')
    lexical.add_transition('q18', '#', 'accept')
    lexical.add_accepted_state('q18')


    # analyze input string
    # input_str = 'for j < k { j = j + 1 }'
    # input_str = input('Enter your code: ')
    print("Enter/Paste your content. Ctrl-D or Ctrl-Z ( windows ) to save it.")
    contents = []
    while True:
        try:
            line = input()
        except EOFError:
            break
        contents.append(line)
    input_str = '\n'.join(contents)
    lexical.analyze(input_str, verbose=True)


if __name__ == '__main__':
    main()
```

1. Pengujian beserta hasil setelah menggunakan grammar yang telah dibuat maka hasil yang di dapat sebagai berikut :

```
PS C:\Users\Bagaz\Pract> py .\tba.py
Enter/Paste your content. Ctrl-D or Ctrl-Z ( windows ) to save it.
for i < j {
i = i + 1
}
^Z
{'current_state': 'q0', 'current_token': '', 'input': 'f'}
{'current_state': 'q1', 'current_token': 'f', 'input': 'o'}
{'current_state': 'q2', 'current_token': 'fo', 'input': 'r'}
{'current_state': 'q3', 'current_token': 'for', 'input': ' '}
Current Token: for
{'current_state': 'q3', 'current_token': '', 'input': 'i'}
{'current_state': 'q4', 'current_token': 'i', 'input': ' '}
Current Token: i
{'current_state': 'q5', 'current_token': '', 'input': '<'}
{'current_state': 'q6', 'current_token': '<', 'input': ' '}
Current Token: <
{'current_state': 'q7', 'current_token': '', 'input': 'j'}
{'current_state': 'q7', 'current_token': 'j', 'input': ' '}
Current Token: j
{'current_state': 'q8', 'current_token': '', 'input': '{'}
{'current_state': 'q9', 'current_token': '{', 'input': '\n'}
Current Token: {
{'current_state': 'q10', 'current_token': '', 'input': 'i'}
{'current_state': 'q10', 'current_token': 'i', 'input': ' '}
Current Token: i
```

```
Current Token: i
{'current_state': 'q11', 'current_token': '', 'input': '='}
{'current_state': 'q12', 'current_token': '=', 'input': ' '}
Current Token: =
{'current_state': 'q13', 'current_token': '', 'input': 'i'}
{'current_state': 'q13', 'current_token': 'i', 'input': ' '}
Current Token: i
{'current_state': 'q14', 'current_token': '', 'input': '+'}
{'current_state': 'q15', 'current_token': '+', 'input': ' '}
Current Token: +
{'current_state': 'q16', 'current_token': '', 'input': '1'}
{'current_state': 'q16', 'current_token': '1', 'input': '\n'}
Current Token: 1
{'current_state': 'q17', 'current_token': '', 'input': '}'}
{'current_state': 'q18', 'current_token': '}', 'input': '#'}
Current Token: }
Your code is valid!
PS C:\Users\Bagaz\Pract>
```