

Tugas Besar
IF2124 Teori Bahasa Formal dan Automata
Compiler Bahasa Python



Disiapkan oleh:
Kelas 02 / Kelompok coba dulu
Juan Louis Rombetasik - 13519075
Christian Tobing Alejandro - 13519109
Christian Gunawan - 13519199

Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
2019

Teori Dasar

A. Finite Automata

Finite Automata (FA) adalah mesin abstrak berupa sistem model matematika dengan masukan dan keluaran diskrit yang dapat mengenali bahasa paling sederhana (bahasa reguler) dan dapat diimplementasikan secara nyata di mana sistem dapat berada di salah satu dari sejumlah berhingga konfigurasi internal disebut state.

Suatu finite automata terdiri dari beberapa bagian. Finite automata mempunyai sekumpulan state dan aturan-aturan untuk berpindah dari state yang satu ke state yang lain, tergantung dari simbolnya. Finite automata mempunyai state awal, sekumpulan state dan state akhir. Finite automata merupakan kumpulan dari lima elemen atau dalam bahasa matematis dapat disebut sebagai 5-tuple.

Sebuah finite automata terdiri dari lima komponen ($Q, \Sigma, \delta, q_0, F$), di mana :

1. Q adalah himpunan set berhingga yang disebut dengan himpunan states.
2. Σ adalah himpunan berhingga alfabet dari simbol .
3. $\delta : Q \times \Sigma$ adalah fungsi transisi, merupakan fungsi yang mengambil states dan alfabet input sebagai argumen dan menghasilkan sebuah state. Fungsi transisi sering dilambangkan dengan δ .
4. $q_0 \in Q$ adalah states awal.
5. $F \subseteq Q$ adalah himpunan states akhir.

B. Context Free Grammar

Context Free Language (CFG) adalah sebuah bahasa yang memiliki cakupan lebih luas daripada regular language dan memberikan deskripsi bahasa dengan mendeskripsikan struktur string secara rekursif dalam bahasa.

CFG memiliki empat komponen penting:

1. Terminal (T), berisi simbol-simbol yang akan menjadi string dalam sebuah bahasa
2. Variabel atau non terminal (V) yang mendefinisikan satu set bahasa
3. Start symbol (S) adalah salah satu dari variabel yang akan dipanggil pertama kali
4. Aturan produksi (P) merupakan definisi rekursif suatu bahasa. Suatu aturan produksi

memiliki bentuk:

$V_1 \rightarrow X$

di mana V_1 adalah head, sebuah variabel dan X , body dari aturan, dapat berupa terminal atau variabel lagi yang akan membentuk sebuah string.

Komponen CFG biasanya dinotasikan dengan four-tuple sebagai $G = (V, T, P, S)$. Misalnya, grammar $G = (\{P\}, \{0, 1\}, A, P)$ memiliki sebuah variabel P , dua terminal 0 dan 1, aturan produksi A , dan start symbol P . Grammar kemudian dapat digunakan untuk menentukan apakah sebuah string termasuk dalam

suatu language atau tidak. Untuk mengetahuinya, digunakan parse tree yang menggambarkan penurunan suatu string.

C. Syntax Python

Syntax python yang perlu diperhatikan adalah

1. IF Statement : if (elmt_if) :
2. ELIF Statement : IF Statement
 Elif (elmt_elif) :
3. ELSE Statement : IF Statement
 Elif Statement
 else :
4. Defining function : def -identifier- (-parameters-) :
5. Identifier
6. False
7. None
8. True
9. And
10. Or
11. Not
12. For loop
13. While loop
14. Operasi Aritmatika
15. Operasi komparasi
16. Statement import
17. Break, return, pass, continue
18. Defining Class
19. Assignment
20. Import
21. Statement
22. Comment
23. Multiline Comment
24. String
25. Array
26. Parentheses dan Bracket
27. Dll

Hasil Pengujian

A. Finite Automata

Finite Automata adalah hasil sebelum dari Context Free Grammar. Jadi, CFG adalah hasil sederhana dari FA. Jika konversi dilakukan secara manual, langkah-langkah yang lebih dulu dikerjakan adalah mengeliminasi, baru melakukan binerisasi. Hasil FA ada pada [link](#) ini.

B. Context Free Grammar

Untuk membuat sebuah Context Free Grammar (CFG), komponen-komponen penyusunnya ditentukan pada Terminal yang mendefinisikan kata kunci yang membentuk string, Non Terminal yang merupakan set bahasa dari Python, Start Symbol yang dimulai dari S sebagai string mulai, dan Production Rule yang berada pada [link](#) ini.

Implementasi & Pengujian

A. Spesifikasi Teknis Program

Algoritma Struktur Data

Tiga file utama yang membentuk program kami ialah mesin_grammar.py, parser_cyk.py, dan lexer.py. mesin_grammar.py melakukan konversi CFG menjadi CNF yang hasilnya disimpan di dalam file CNF.txt pada folder grammar. parser_cyk.py melakukan mekanisme pencocokan untuk mengkonversi dalam bahasa CNF. Keluaran dari parser_cyk.py adalah "Accepted" atau "Syntax Error". Pada lexer.py digunakan untuk membaca masukan bahasa dan kemudian dijadikan token-token untuk dicocokkan dengan parser_cyk.py. Pada lexer.py dibutuhkan parser_cyk untuk membaca dan mengeluarkan *output*.

Fungsi dan Prosedur

Di dalam file mesin_grammar.py, terdapat tiga fungsi. Pertama fungsi *baca_grammar* untuk membaca grammar yang sudah dibuat di file dan mengkonversinya menjadi list rules. Kedua fungsi *tambah_rule* untuk menambahkan rule baru ke dalam kamus, caranya dengan prosedur akan memvalidasi apakah rule tersebut sudah berada pada *head rule* atau belum, jika sudah maka rule akan ditambahkan pada *body rule*. Ketiga adalah fungsi *konversi_grammar* untuk mengkonversi rule dengan mengecek rule yang sudah ada

Di dalam file parser_cyk.py terdapat dua kelas yang berisikan fungsi dan prosedur. Pertama adalah kelas Node untuk menyimpan informasi grammar yang terdiri dari dua non terminal. Kedua adalah kelas parser untuk melakukan parsing dari CNF(Rules dict) dengan bahasa yang dimasukan, terdiri dari satu non terminal, satu fungsi, dan satu prosedur.

Di dalam file lexer.py terdapat tiga kelas dan dua fungsi utama. Pertama adalah kelas token untuk menyimpan informasi token yang berisikan dua fungsi non terminal. Kedua kelas LexerError untuk mengembalikan nilai dimana error terjadi yang berisikan satu fungsi non terminal. Ketiga kelas Lexer untuk memproses *input* bahasa menjadi token-token satu fungsi non terminal dan tiga fungsi non terminal. Keempat adalah fungsi *read_files_input* untuk membaca masukan bahasa dari file. Kelima fungsi *write_files_output* untuk menulis *output* bahasa yang telah menjadi token-token untuk kemudian digunakan untuk *parsing*.

Antarmuka

Pada tugas kali ini, antar muka hanya menggunakan CLI. Untuk menjalankan program, silahkan masukkan jalankan script `pypret.[sh|bat]*` di terminal pada root folder. Script pypret menerima 1 input argument <nama_file> yang bertipe data berupa string. Jika file ada maka akan dilanjutkan dengan menjalankan python src/lexer.py <nama_file>. Jika jumlah parameter tidak sama dengan 1, maka akan mengoutput "invalid arguments". Jika file tidak ada, maka akan mengeluarkan *output* "file <nama_file> tidak ada".

*Keterangan: pypret.bat dijalankan pada OS windows dan pypret.sh dijalankan pada OS linux

B. Capture Berbagai Kasus

Testing dilakukan dengan menjalankan ./pypret.[sh|bat] <nama_file>
Semua file test terdapat pada folder test

```
# file pypret.sh
if [ "$#" -eq 1 ]
then
    if [ -f "$1" ]
    then
        echo lexing file "$1"
        python3 src/lexer.py "$1"
    else
        echo file "$1" tidak ada
    fi
else
    echo invalid arguments
fi
```

```
# file pypret.bat
@echo off
REM MIT-License PYTHON LEXER and Parser (c) 2021, Coba Dulu. 13519075,
13519109, 13519199

REM mencatat jumlah argument (Argument Count)
SET argC=0
FOR %%x in (%*) DO SET /A argC+=1

REM jika argument == 1 maka proses file
IF %argC% == 1 (
    REM mengecek existensi file
    IF exist %1 (
        REM file ada
        @echo Lexing file %1%
        python src/lexer.py %1%
    ) ELSE (
        REM file tidak ada
        @echo file %1% tidak ada
    )
) ELSE (
    @echo invalid arguments.
)

exit /B 1
```

1. inputReject.py

```
mizuday ~/repo/cobadulu main cat test/inputReject.py
def do_something(x):
    ''' This comment '''
    x + 2 = 3
    if x = 0 + 1
        return 0
    elif x + 4 = 1:
        else:
            return 2
    elif x = 32:
        return 4
    else:
        return "Doodoo"
mizuday ~/repo/cobadulu main ./pypret.sh test/inputReject.py
lexing file test/inputReject.py
Syntax error
  pada line 3
  x + 2 = 3
Syntax error
  pada line 4
  if x = 0 + 1
```

Kode program dalam inputReject.py gagal untuk *dicompile* karena:

- Pada line 3, syntactically salah karena ruas kiri merupakan suatu ekspresi aritmatika dan diassign dengan sebuah number
- Pada line 4, kurang semikolon
- Pada line 12, tanda kutip kurang satu buah

2. inputAcc.py

```
mizuday ~/repo/cobadulu main cat test/inputAcc.py
def do_something(x):
    ''' This comment '''
    if x = 0:
        return 0
    elif x + 4 = 1:
        if True:
            return 3
        else:
            return 2
    elif x = 32:
        return 4
    else:
        return "Doodoo"%
mizuday ~/repo/cobadulu main ./pypret.sh test/inputAcc.py
lexing file test/inputAcc.py
Accepted
```

Kode program dalam inputAcc.py berhasil untuk *dicompile* sesuai dengan peraturan yang ada.

3. File1.py

```
mizuday ~/repo/cobadulu ➜ main ➤ cat test/file1.py
import numpy as num
import pandas as py
importasd opencv as op
for i in range():
    while :
        if 1 > 0;
            # ini comment
            break::
        elif x < x:
            pass
        else:
            print("ini print")
            print()
mizuday ~/repo/cobadulu ➜ main ➤ ./pypret.sh test/file1.py
lexing file test/file1.py
Syntax error
    pada line 3
        importasd opencv as op
Syntax error
    pada line 5
        while :
Syntax error
    pada line 6
        if 1 > 0;
```

Kode program dalam file1.py gagal untuk *dicompile* karena

- Token importasd tidak terdaftar dalam CFG (dikenali sebagai identifier)
- Pada line 5, setelah while hanya terdapat kolon (titik dua). Tidak ada ekspresi, atau fungsi, test pada kelanjutan token while
- Pada line 6, seharusnya memakai simbol kolon (titik dua), bukan semikolon.

4. file2.py

```
mizuday ~/repo/cobadulu ➜ main ➤ cat test/file2.py
class method(ibasdasd):
    def funcname(self, parameter_list):
        pass
variavleBoolean = True
if (not(variavleBoolean)):
    while(1):
        if True:
            break
        elif False:
            pass
        else:
            fr = "ddwdwdw"
            var = 100
            print(fr+'frfe')
            print(fr, sd, sdasd, var)
def function(sentences, b):
    c = False
    if (len(sentences) > 5 and b) or c:
        pass
    else :
        i = len(sentences)
        for i in range(len(a) + 100):
            print(i)
            if (i > 10 and (i % 10 == 1)):
                continue%
mizuday ~/repo/cobadulu ➜ main ➤ ./pypret.sh test/file2.py
lexing file test/file2.py
Accepted
```

Kode program dalam file2.py berhasil untuk *dicompile* sesuai dengan peraturan yang ada.

5. input4.txt

```
mizuday ~/repo/cobadulu main cat test/input4.txt
N = 5
i = 1
sum = 0

whil (i <= 5) ::#
    sum += i
    i += 1%
mizuday ~/repo/cobadulu main ./pypret.sh test/input4.txt
lexing file test/input4.txt
Syntax error
  pada line 4
    whil (i <= 5) ::#
```

Kode program dalam input4.py gagal untuk *dicompile* karena terjadi kondisi yang salah pada statement *while*. Terjadi salah ketik 'whil' dan tanda titik dua yang ditulis salah ketik (dua kali titik dua) akan menyebabkan program mengalami error dan gagal untuk *dicompile*.

****Keterangan: jika terdapat symbol percent '%' pada line terakhir, maka artinya menandakan end of file (tidak ada new line pada line terakhir)**

Link Github : <https://github.com/ChristianGunawan/cobadulu>

Pembagian Tugas :

13519075	13519109	13519199
Analisis CFG	Analisis CFG	Membuat Repo
Parser + Syntactic Analyzer	Lexer	Laporan
Mesin Grammar	Laporan	Analisis CFG

Referensi

<https://www.geeksforgeeks.org/introduction-of-finite-automata/>

<https://stackoverflow.com/questions/13728581/how-does-the-cyk-algorithm-work>

<https://stackoverflow.com/questions/14954721/what-is-the-difference-between-a-token-and-a-lexeme>