Laporan Tugas Besar

IF2124 Teori Bahasa Formal dan Otomata

Compiler Bahasa Python

Disusun Oleh:

K03 – Kelompok 11 (CompilerLul)

13520131 Steven

13520156 Dimas Faidh Muzaki

13520161 M Syahrul Surya Putra

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2021

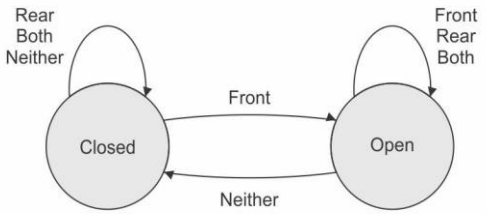
DAFTAR ISI

BAB I
DASAR TEORI

1.1 Finite Automata

Finite automata adalah model yang baik untuk komputer yang memiliki jumlah atau kapasitas memori terbatas.

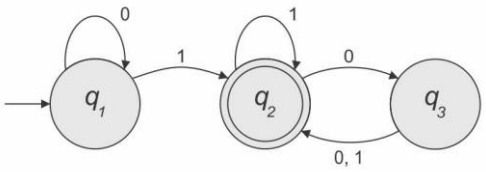
Salah satu contoh yang dapat dilakukan dari model *finite automata* adalah pintu otomatis yang dapat buka tutup tanpa menggunakan bantuan dari manusia.



state	input signal: Neither	Front	Rear	Both
Closed	Closed	Open	Closed	Closed
Open	Closed	Open	Open	Open

Kontroler pintu otomatis tersebut bergerak dari *state* yang satu ke *state* yang lain, tergantung dari masukan (*input*) yang diterima.

Misal, jika kontroler tersebut dimulai dengan status ***closed*** dan menerima rangkaian sinyal *input* ***front, rear, neither, front, both, neither, rear,*** dan ***neither***, maka status tersebut menjadi ***closed*** (**mulai**)***, open, open, closed, open, open, closed, closed,*** dan ***closed.***



Finite-state machine (FSM) atau *finite-state automaton* (FSA, automata) atau *finite automata* adalah model komputasi matematika.

Mesin ini dapat berubah dari kondisi yang satu ke kondisi yang lain, bergantung dari *input* yang didapatkan.

Terdapat jua jenis finite state:

1. *Deterministic Finite Automata* (DFA) / *Deterministic Finite-State Machine* (DFSM) / *Deterministic Finite-State Automata* (DFSA)
2. *Non-Deterministic Finite Automata* (NFA) / *Non-Deterministic Finite-State Machine* (NFSM) / *Non-Deterministic Finite-State Automata* (NFSA)

1.2 Context Free Grammar dan Parsing

Context free Grammar atau yang biasanya dikenal dengan istilah CFG merupakan sebuah tata bahasa bebas konteks adalah kumpulan berhingga dari variabel- variabel biasa yang disebut non terminal. Variabel tersebut merepresentasikan suatu bahasa. Bahasa yang direpresentasikan oleh non terminal dideskripsikan secara rekursif, dimana tiap simbol primitive disebut terminal.

Contoh dari CFG adalah:

CFG Example

- Language of palindromes
  - We can easily show using the pumping lemma that the language $L = \{ w \mid w = w^R \}$ is not regular.
  - However, we can describe this language by the following context-free grammar over the alphabet {0,1}:

$P \rightarrow \varepsilon$
$P \rightarrow 0$
$P \rightarrow 1$
$P \rightarrow 0P0$ Inductive definition
$P \rightarrow 1P1$

More compactly: $P \rightarrow \varepsilon \mid 0 \mid 1 \mid 0P0 \mid 1P1$

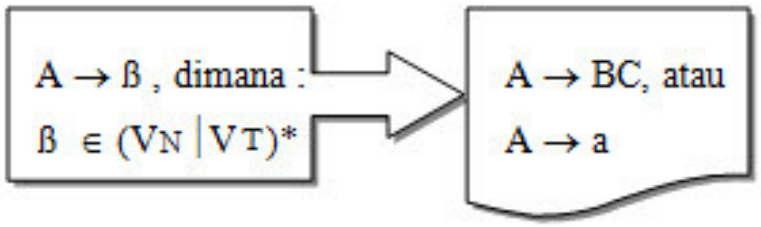
Terdapat 2 buah metode parsing, yaitu parsing top-down dan parsing bottom-up. Parsing top-down merupakan parsing yang dimulai dari simbol awal S sampai kalimat X. Sebaliknya, parsing bottom-up merupakan parsing yang dimulai dari kalimat X sampai simbol awal S.

Metode parsing top-down terdiri dari 2 metode yaitu dengan menggunakan metode backup seperti *Brute-Force* dan tanpa backup seperti *recursive descent.* Metode parsing bottom-up salah satunya adalah Grammar Preseden Sederhana (GPS).

1.3 Chomsky Normal Form (CNF)

Bentuk normal Chomsky atau Chomsky Normal Form (CNF) merupakan salah satu bentuk normal yang sangat berguna untuk Context Free Grammar (CFG). Bentuk normal Chomsky dapat dibuat dari sebuah tata bahasa bebas konteks yang telah mengalami penyederhanaan yaitu penghilangan produksi useless, unit, dan ε. Dengan kata lain, suatu tata bahasa bebas konteks dapat dibuat

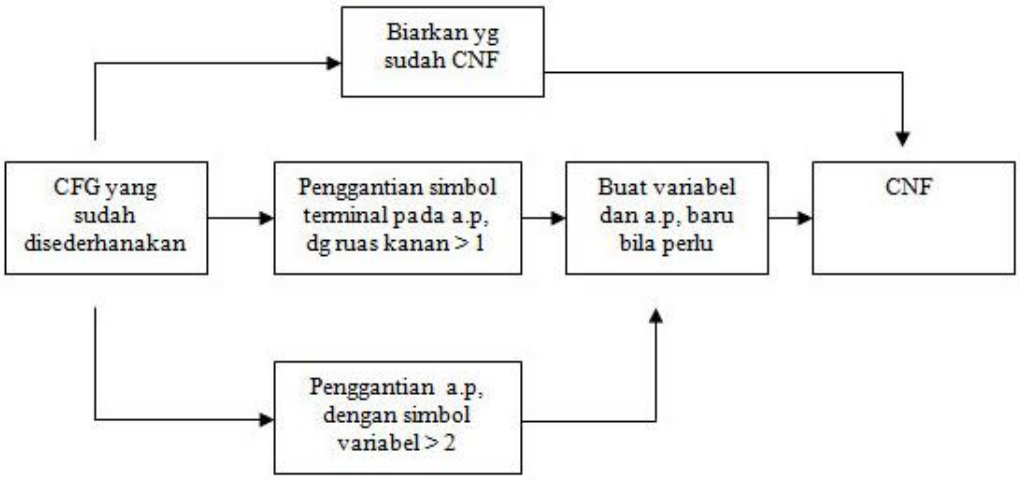
menjadi bentuk normal Chomsky dengan syarat tata bahasa bebas kontesk tersebut tidak memiliki produksi useless, tidak memiliki produksi unit, dan tidak memiliki produksi epsilon.



Transformasi CFG Ke CNF

Langkah-langkah pembentukan bentuk normal Chomsky secara umum sebagai berikut:

- Biarkan aturan produksi yang sudah dalam bentuk normal Chomsky
- Lakukan penggantian aturan produksi yang ruas kanannya memuat simbol terminal dan panjang ruas kanan > 1
- Lakukan penggantian aturan produksi yang ruas kanannya memuat > 2 simbol variabel
- Penggantian-penggantian tersebut bisa dilakukan berkali-kali sampai akhirnya semua aturan produksi dalam bentuk normal Chomsky
- Selama dilakukan penggantian, kemungkinan kita akan memperoleh aturan-aturan produksi baru, dan juga memunculkan simbol-simbol variabel baru



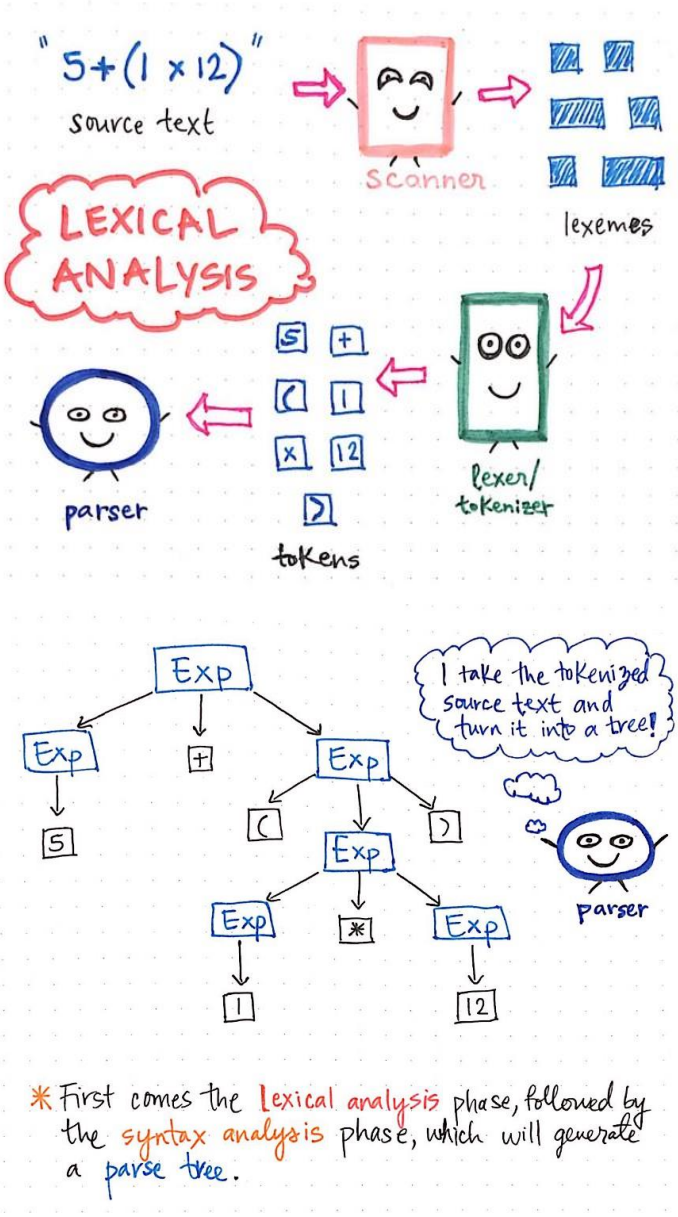
Langkah-langkah pembentukan bentuk normal Chomsky

Algoritma CYK merupakan algoritma *parsing* dan keanggotaan (*membership)* untuk tata Bahasa bebas konteks. Algortima ini diciptakan oleh J. Cocke, DH. Younger, dan T. Kasami. Syarat untuk

penggunaan algortima ini adalah tata bahasa harus berada dalam *bentuk normal Chomsky* Obyektif dari algortima ini adalah untuk menunjukkan apakah suatu *string* dapatdiperoleh dari suatu tata bahasa.

1.4 Lexer

Lexer adalah suatu komponen yang berfungsi untuk mendeteksi bagian terkecil (token) dari suatu bahasa. Proses pendeteksian ini disebut *lexing*. Biasanya *regular expressions* digunakan untuk melakukan proses *lexing* karena bentuk token biasanya sangat sederhana. Setelah selesai melakukan lexing, token-token yang dihasilkan kemudian dibuat menjadi bentuk pohon sedemikian rupa sehingga dapat dilakukan parsing. Untuk gambaran yang lebih jelasnya, dapat dilihat pada gambar di bawah ini.



1.5 Python

Python adalah bahasa pemrograman interpretatif multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. Python diklaim sebagai bahasa yang menggabungkan kapabilitas, kemampuan, dengan sintaksis kode yang sangat jelas, dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif.

Python merupakan bahasa pemrograman yang dianggap mudah untuk dipelajari, sekalipun oleh para pemula. Kode-kode yang ada di dalamnya mudah dibaca dan dapat menjalankan banyak fungsi kompleks dengan mudah karena banyaknya *standard library*. Meskipun begitu, terdapat kekurangan dari Python yang layak menjadi pertimbangan. Kekurangan ini yaitu cukup lambat dijalankan terutama untuk pengembangan *platform* Android dan iOS. Itulah mengapa kedua *operating system* tersebut dikembangkan dengan bahasa yang berbeda.

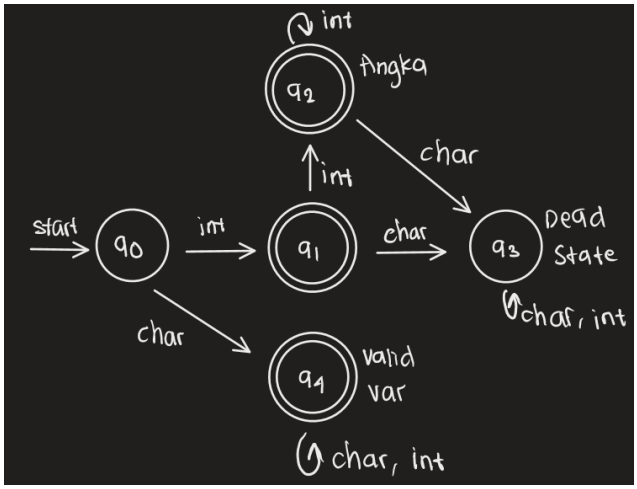
Pada python, keyword merupakan sesuatu yang telah dikhususkan. Akibatnya, kita tidak dapat mendeklarasikan variabel dengan menggunakan keyword yang telah disediakan oleh python.

Keywords in Python programming language

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

BAB II
HASIL

Berikut hasil FA yang kelompok kami telah buat



Karena pada rules_lexer.py kami menggunakan *Regular Expression*, FA di atas kami ubah juga menjadi *Regular Expression*. Untuk ekspresinya sendiri ialah sebagai berikut:

- Untuk mengenali bukan variable : \d+[a-zA-Z_] + [\da-zA-Z_0-9]*
- Untuk mengenali number : \d+
- untuk mengenali variable : [a-zA-Z_] + [\da-zA-Z_0-9]*

Berikut hasil CFG yang kelompok kami telah buat

S	S S | COM IM | IM COM | METHOD COM | COM | IM | METHOD | FOR_METHOD | PRINT_METHOD | IFELSE | WHILE_METHOD | DEF | WITH | CLASS | OBJECT | ASSIGNMENT | RETURN | PASS | RAISE | BREAK | CONTINUE | ARBITRARY_METHOD
COM	COMS NEWLINE | COMS
COMS	‘COMMENT'
IM	‘FROM' A | B
A	IDENTIFIER B
B	‘IMPORT' C
C	IDENTIFIER | IDENTIFIER D
D	‘AS' E
E	IDENTIFIER | IDENTIFIER IMEND
IMEND	NEWLINE
IDENTIFIER	‘IDENTIFIER' WITH_METHOD | 'IDENTIFIER" | NUMBER | 'TRUE' | 'FALSE' | OBJECT

WITH_METHOD	WITH_METHOD' IDENTIFIER
PRINT_METHOD	PRINT M1
M1	LP M2
M2	OBJECT RP | RP | STRING MS | FLOAT MF | IDENTIFIER MS | NUMBER | EXPRESSION RP
MS	RP | PLUS MPLUSSTRING | MULTIPLY MMULTIPLYINTEGER | COMA M2
MMULTIPLYINTEGER	NUMBER MS
MPLUSSTRING	STRING MS | IDENTIFIER MS
MF	RP | COMA M2 | PLUS MNUMBER | MULTIPLY MNUMBER | MINUS MNUMBER | DIVIDE MNUMBER | POWER MNUMBER
MI	RP | COMA M2 | PLUS MNUMBER | MULTIPLY M2 | MINUS MNUMBER | DIVIDE MNUMBER | POWER MNUMBER
MNUMBER	NUMBER MF | NUMBER MI | FLOAT MF | FLOAT MI
NUMBER	‘NUMBER'
MULTIPLY	‘MULTIPLY'
PRINT	‘PRINT'
PLUS	‘PLUS'
COMA	‘COMA'
FUNC	‘FUNC'
STRING	‘STRING'
FLOAT	‘FLOAT'
FOR_METHOD	FOR F1
F1	IDENTIFIER F2
F2	IN F3
F3	IDENTIFIER COLON | ARBUTRARY_METHOD F10
F10	RP COLON | RP
FOR	‘FOR'
IN	‘IN'
LEN	‘LEN'
RANGE	‘RANGE'
COLON	‘COLON'
IFELSE	IF IF2 | IF4 | IF7 | IF IF97
IF97	OBJECT IF0
IF0	IN IF1
IF1	EXPRESSION IF3
IF2	COLON IFNL
IF3	COLON
IFNL	NEWLINE IFNEXT

IFNEXT	CONTENT
IF4	ELIFTKN IF99
IF99	ELIF IF5
IF5	EXPRESSION IF6 | IDENTIFIER IF0
IF6	COLON IFNL | COLON
IF7	ELIFTKN IF98
IF98	ELSE IF8
IF8	COLON IF9 | COLON
IF9	NEWLINE IF10 | IF10
IF10	CONTENT
IF	‘IF'
ELSE	‘ELSE'
ELIF	‘ELIF'
ELIFTKN	‘ELIFTKN'
CONTENT	S
EXPRESSION	OBJECT | OBJECT A | EXPRESSION A | NOT EXPRESSION | EXPRESSION B | LP C
A	COMPARE_OP EXPRESSION
B	BINARY_OP EXPRESSION | ARITHMETIC_OP EXPRESSION
EXPRESSION	OBJECT | OBJECT A | EXPRESSION A | NOT EXPRESSION | EXPRESSION B | LP C
C	EXPRESSION RP
COMPARE_OP	‘DOUBLEEQUAL' | 'GREATER_OR_EQUAL_THAN' | 'LESS_OR_EQUAL_THAN' | 'GREATER_THAN' | 'LESS_THAN' | 'NOT_EQUAL'
BINARY_OP	‘IS' | 'AND' | 'OR'
NOT	‘NOT'
WHILE_METHOD	WHILE W1
W1	IDENTIFIER W4 | EXPRESSION W4 | LP W2
W2	IDENTIFIER W3 | EXPRESSION W3
W3	RP W4
W4	COLON NEWLINE | COLON
WHILE	‘WHILE'
NEWLINE	‘NEWLINE'
DEF	DEFWORD DEF1
DEF1	IDENTIFIER DEF2 | IDENTIFIER COLON
DEF2	DEF4
DEF2	LP DEF3

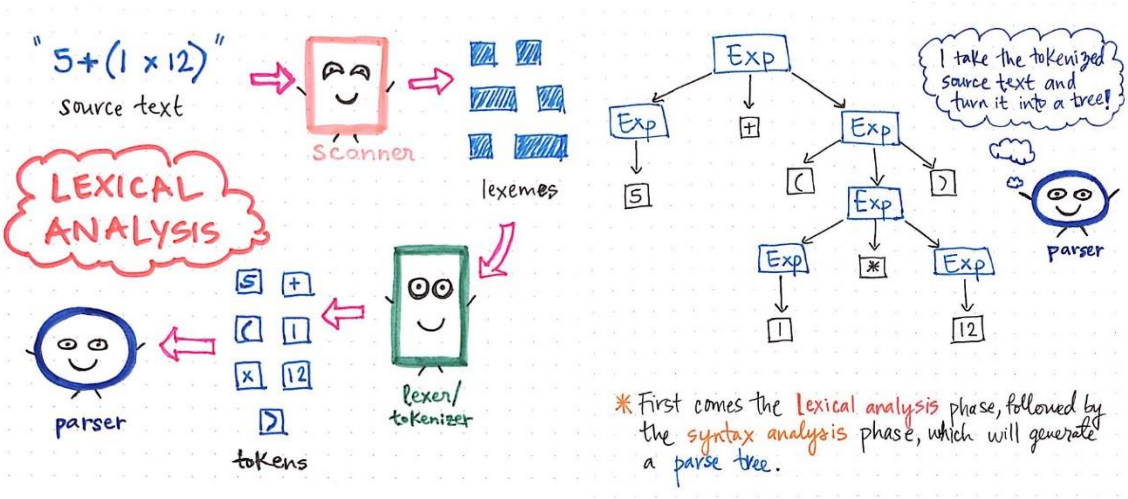
DEF3	DEF4 | IDENTIFIER DEF4 | STRING DEF4 | NUMBER DEF4 | IDENTIFIER | STRING | NUMBER | DEF3 DEFINBETWEEN
DEFINBETWEEN	COMA DEF3
DEF4	RP COLON
DEFWORD	‘DEF'
WITH	WITHWORD WITH0
WITH0	IDENTIFIER WITH1
WITH1	LP WITH2
WITH2	IDENTIFIER WITH3 | STRING WITH3 | NUMBER WITH3
WITH3	RP WITH4 | COMA WITH2
WITH4	COLON | 'AS' WITH5
WITH5	IDENTIFIER COLON
WITHWORD	‘WITH'
CLASS	CLASSWORD CLASS1
CLASS1	IDENTIFIER COLON | IDENTIFIER DEF2
CLASSWORD	‘CLASS'
OBJECT	STRING' | NUMBER | 'IDENTIFIER' | 'FALSE' | 'NONE' | 'TRUE' | OBJECT OBJ1 | STRING OBJ1 | OBJECT OBJ3 | OBJECT OBJ0 | OBJECT OBJ0 | OBJECT OBJIN
OBJIN	IN OBJECT
OBJ0	ARRAY | ARRAY OBJ1 | ARRAY OBJ3
OBJ1	‘WITH_METHOD' OBJ2
OBJ2	OBJECT | 'IDENTIFIER' ARRAY | OBJECT OBJ3
OBJ3	LP OBJ4 | LP OBJRP
OBJRP	RP
OBJ4	IDENTIFIER OBJ5 | STRING OBJ5 | NUMBER OBJ5 | OBJECT OBJ5 | EXPRESSION OBJ5 | OBJ5
OBJ5	ARITHMETIC_OP AOP1 | RP | RP OBJ1
AOP1	OBJ4
OBJ2	OBJ1
ASSIGNMENT	OBJECT ASS1
ASS1	EQUALS OBJECT | EQUALS NUMBER | EQUA:S STRING | EQUALS ARRAY | EQUALS EXPRESSION
ARRAY	‘LB' ARR1
ARR1	‘RB' | OBJECT 'RB' | STRING 'RB' | NUMBER 'RB' | 'IDENTIFIER' ARR2 | 'IDENTIFIER' 'RB'
ARR2	FOR_METHOD 'RB'
EQUALS	‘EQUALS'

ARITHMETIC_OP	‘MINUS' | 'PLUS' | 'MULTIPLY' | 'DIVIDE' | 'POWER' | 'MOD"
RETURN	‘RETURN' OBJECT
PASS	‘PASS'
RAISE	‘RAISE' OBJECT
BREAK	‘BREAK'
CONTINUE	‘CONTINUE'
ARBITRARY_METHOD	IDENTIFIER ARM1
ARM1	LP ARM2
ARM2	ARBITRARY_METHOD RP | ARM 1 RP | OBJECT RP | EXPRESSION RP
LP	‘LP’
RP	‘RP’
LC	‘LC’
RC	‘RC’
MULTILINE	TRIPLEQUOTE

BAB III
IMPLEMENTASI DAN PENGUJIAN

3.1 Spesifikasi Teknis Program

Program yang kami buat terdiri dari 3 file utama, yaitu file main.py, grammarConverter.py, dan cyk_parser.py. Pada file main.py, dilakukan proses lexing dan hasil proses lexing yang berupa token tersebut diolah lagi oleh parser sesuai dengan grammar yang telah dibuat. Tidak hanya itu, pada program main.py juga menampilkan waktu parsing dan juga pesan error dan detail error apabila terdapat error. File grammarConverter.py merupakan file yang dimana terjadinya konversi grammar.txt menjadi grammar yang akan digunakan untuk keberlangsungannya parsing. File cyk_parser.py merupakan file yang dimana proses parsing terjadi. Inti dari program yang kami buat adalah sama dengan prosedur penggunaan lexer dalam parsing yang telah dijelaskan pada BAB 1 atau tertera pada gambar di bawah.



3.2 Screenshot Kasus

1. inputAcc.py

```python
def do_something(x):
    ''' This is a sample multiline 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"
```

```
D:\Kuliah\Semester 3\TBFO\Tubes-TBFO>main tc/inputAcc.py
Memulai parsing pada 14 baris kode
Parsing telah selesai
Tidak ada error yang ditemukan
Waktu yang dibutuhkan untuk melakukan parsing adalah  0.396 detik
--------------------------------------------------
```

2. inputReject.py

```python
def do_something(x):
    ''' This is a sample multiline comment
    '''
    x + 2 = 3
    if x == 0 + 1
        return 0    Expected ":"
    elif x + 4 == 1:
        else:    Expected expression
            return 2    Unexpected indentation
    elif x == 32:    Expected expression
        return 4    Unexpected indentation
    else:    Unindent not expected
        return "Doodoo"    Unexpected indentation
```

```
D:\Kuliah\Semester 3\TBFO\Tubes-TBFO>main tc/inputReject.py
Memulai parsing pada 13 baris kode
Terdeteksi 4 buah Error pada file
Detail kesalahan:
     Error pada line 4.
     Error pada line 5.
     Error pada line 10.
     Error pada line 12.
Waktu yang dibutuhkan untuk melakukan parsing adalah  0.517 detik
--------------------------------------------------
```

3. if.py

```python
n = None
def Func():
    if not False or True:
        return True
    elif False and True:
        return False
    else: return n is None
```

```
D:\Kuliah\Semester 3\TBFO\Tubes-TBFO>main tc/if.py
Memulai parsing pada 7 baris kode
Parsing telah selesai
Tidak ada error yang ditemukan
Waktu yang dibutuhkan untuk melakukan parsing adalah  0.302 detik
--------------------------------------------------
```

4. pass.py

```python
class ExampleClass:
    def __init__(self):
        pass
...
class Exam(ExampleClass):
    def __init__(self, boo):
        if not boo:
            raise ValueError
        pass
```

```
D:\Kuliah\Semester 3\TBFO\Tubes-TBFO>main tc/pass.py
Memulai parsing pada 8 baris kode
Parsing telah selesai
Tidak ada error yang ditemukan
Waktu yang dibutuhkan untuk melakukan parsing adalah  0.449 detik
--------------------------------------------------
```

5. while.py

```
def fun(num):
    while num < 100:
        if num < 10:
            num = num + 1
            continue
        if num > 50:
            break
        num = num + 1
    for i in range(num):
        print(i)
    return num
```

```
D:\Kuliah\Semester 3\TBFO\Tubes-TBFO>main tc/while.py
Memulai parsing pada 10 baris kode
Parsing telah selesai
Tidak ada error yang ditemukan
Waktu yang dibutuhkan untuk melakukan parsing adalah  0.578 detik
------------------------------------------------------
```

6. Import.py

```
from matplotlib import pyplot as plt
from tubes import pathName

with open(pathName, 'r') as my_file:
    lines = my_file.readlines()
    for line in lines:
        print(line)
```

```
PS C:\Users\Asus\desktop\github\Tubes-TBFO> py main.py ".\tc\import.py"
Memulai parsing pada 7 baris kode
Parsing telah selesai
Tidak ada error yang ditemukan
Waktu yang dibutuhkan untuk melakukan parsing adalah  0.264 detik
------------------------------------------------------------
```

7. numvar.py

```
1   inivar = "benar"
2   123var = "gakbolehgini"
```

```
PS C:\Users\Asus\desktop\github\Tubes-TBFO> py main.py ".\tc\numvar.py"
Memulai parsing pada 2 baris kode
Terdeteksi 1 buah Error pada file
Detail kesalahan:
     Error pada line 2.
Waktu yang dibutuhkan untuk melakukan parsing adalah  0.01 detik
-----------------------------------------------------------------
```

BAB IV
PENUTUP

4.1 **K**esimpulan

Compiler merupakan salah satu alat yang penting bagi semua orang yang ingin membuat kode. Pada Tugas Besar IF2124 Teori Bahasa Formal dan Otomata ini, telah dibuat suatu program yang fungsinya mirip dengan *compiler* menggunakan bahasa Python. Program ini bisa mengecek error yang terdapat pada file ataupun string yang dijadikan masukan, dengan catatan, file atau string yang dimasukan menggunakan sintaks bahasa Python yang benar untuk lolos "kompilasi".

4.2 Saran

Dalam pengerjaan tugas besar ini, kami sangat menyadari kendala dan keterbatasan pada program yang kami buat. Oleh karena itu, kami mendapatkan beberapa saran yang bisa membuat kami sebagai pembuat program ataupun programnya sendiri lebih baik. Berikut saran tersebut:

1. Mencicil tugas yang ada lebih awal lagi
2. Giat mencari referensi yang terkait dengan CFG, CNF, Lexer, dan metode lain yang bersangkutan
3. Perbaiki dan/atau kembangkan skill *time management*

REFERENSI

GHH “Raw Strings, Python and re, Normal vs Special Characters”, diakses pada 24 November pukul 13.24 WIB, https://stackoverflow.com/questions/41067866/raw-strings-python-and-re-normal-vs-special-characters

McHardy, Robert, “CYK-Parser”, diakses pada 21 November pukul 21.47 WIB, https://github.com/RobMcH/CYK-Parser

pitanga, “Text file parsing with python and with a list in grammar”, diakses pada 21 November pukul 21.23 WIB, https://stackoverflow.com/questions/45981339/text-file-parsing-with-python-and-with-a-list-in-grammar

Python, “cpython”, diakses pada 24 November pukul 13.10 WIB, https://github.com/python/cpython

Trofimovich, Ulya, “RE2C - a lexer generator based on lookahead TDFA”, diakses pada 19 November pukul 13.43 WIB, https://re2c.org/2021_trofimovich_re2c_a_lexer_generator_based_on_lookahead_tdfa_(slides).pdf

LAMPIRAN

Link Repository: https://github.com/loopfree/Tubes-TBFO

Tabel Pembagian Tugas

Nama	Tugas
Steven	Membuat Lexer & Laporan
Dimas Faidh Muzaki	Membuat Grammar Converter & Laporan
M Syahrul Surya Putra	Membuat CYK Parser & Laporan