LAPORAN TUGAS BESAR 2

IF 2124 Teori Bahasa Formal dan Otomata

Compiler Bahasa Python

Semester I Tahun 2021/2022

Disusun oleh:

Farhan Hafiz	13520027
Fitrah Ramadhani Nugroho	13520030
Muhammad Akmal Arifin	13520037

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2021

Daftar Isi

Bab 1:
Teori Dasar

1.1 Finite Automata

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. 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.

Sebuah finite automata terdiri dari lima komponen (Q, Σ, δ, q0, F), dengan keterangan :

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. $q0 \subseteq Q$ adalah states awal.
5. $F \subseteq Q$ adalah himpunan states akhir.

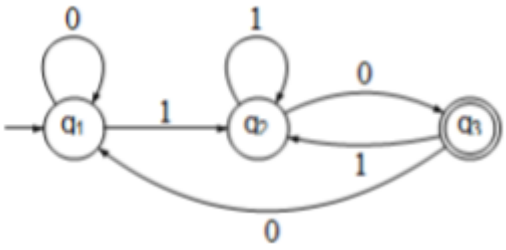
Penyajian Finite Automata

- Penyajian FA Menggunakan Diagram Transisi

  Diagram transisi untuk finite automata M = (Q, Σ, δ, q0, F) adalah suatu graf yang didefinisikan sebagai berikut :

  1. Terdapat simpul untuk setiap state Q.
  2. Untuk setiap state $q \epsilon Q$ dan setiap simbol input $a \in \Sigma$, berlaku δ(q, a) = p. Diagram transisi memiliki busur berlabel a dari state q ke state p.
  3. Terdapat anak panah berlabel start yang mengarah ke state awal q0 dan anak panah ini tidak berasal dari state manapun.

4. State yang merupakan state akhir (F) akan ditandai dengan lingkaran ganda, sedang state yang lain menggunakan lingkaran tunggal.



- Penyajian FA Menggunakan Tabel Transisi

δ	0	1
→ q_1	q_1	q_2
q_2	q_3	q_2
* q_3	q_1	q_2

Tabel transisi di atas memiliki keterangan :

1. Simbol pada kolom sebelah kiri adalah state.
2. Simbol pada baris paling atas adalah simbol .
3. Simbol yang berada "'dalam"' tabel merupakan fungsi transisi.
4. Simbol panah (→) pada kolom sebelah kiri menunjukkan start simbol.
5. Simbol (*) pada kolom sebelah kiri menunjukkan state final.

1.2 Context Free Grammar

Context Free Grammar

Context Free Grammar (CFG) adalah tata Bahasa yang mempunyai tujuan sama seperti tata Bahasa regular yaitu merupakan suatu cara untuk menunjukkan bagaimana menghasilkan suatu untai-untai dalam sebuah Bahasa.

Definisi formal dari CFG dapat didefinisikan sebagai berikut :

G = (V, T, P, S)

Dengan keterangan :

V = Himpunan terbatas variable

T = Himpunan terbatas terminal

P = Himpunan terbatas dari produksi

S = start symbol

Context Free Grammar (CFG) merupakan sebuah tata Bahasa dimana tidak terdapat pembatasan pada hasil produksinya. Contoh pada aturan produksi :

$$\alpha \rightarrow \beta$$

batasannya hanyalah ruas kiri (α) adalah sebuah simbol variabel. Sedangkan contoh aturan produksi yang termasuk CFG adalah seperti di bawah ini :

$$B \rightarrow CDeFg$$

$$D \rightarrow BcDe$$

Proses parsing adalah proses pembacaan string dalam bahasa sesuai CFG tertentu, proses ini harus mematuhi aturan produksi dalam CFG tersebut.

Chomsky Normal-Form

Chomsky Normal Form adalah salah satu bentuk dari Context Formal Grammar. CNF dapat dibuat dari sebuah Context Free Grammar yang telah disederhanakan dengan menghilangkan produksi *useless*, unit, dan ε . Suatu CFG dapat dibuat menjadi CNF dengan syarat CFG tersebut tidak memiliki produksi *useless*, unit, ε . CNF berguna untuk mengecek keanggotaan suatu string dengan menggunakan algoritma CYK.

Parsing

Context Free Grammar (CFG) menjadi dasar dalam pembentukan suatu parser/proses analisis sintaksis. Bagian sintaks dalam suatu kompilator kebanyakan di definisikan dalam tata Bahasa bebas konteks. Pohon penurunan (derivation tree / parse tree) berguna untuk menggambarkan simbol-simbol variabel menjadi simbol-simbol terminal setiap simbol variabel akan di turunkan menjadi terminal sampai tidak ada yang belum tergantikan.

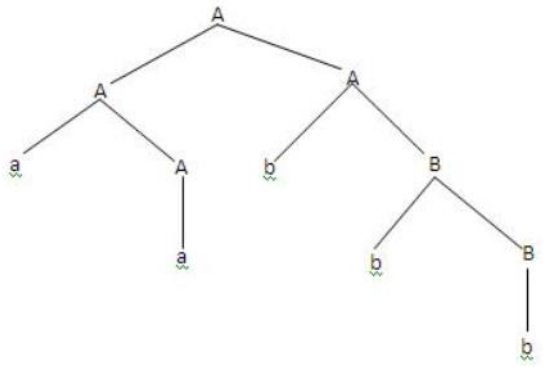
Contoh, terdapat CFG dengan aturan produksi dengan symbol awal S :

$S \rightarrow AB$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

Maka, jika ingin dicari gambar pohon penurunan dengan string : ‘aabbb’ hasilnya akan seperti di bawah ini :



Parse Tree

Proses penurunan / parsing bisa dilakukan dengan cara sebagai berikut :

- Penurunan terkiri (*leftmost derivation*): simbol variabel terkiri yang di perluas terlebih dahulu.
- Penurunan terkanan (*rightmost derivation*) : simbol variabel terkanan yang diperluas terlebih dahulu.

Misal : Grammar sebagai berikut :

- S → aAS | a
- A → SbA | ba

Untuk memperoleh string ‘aabbaa’ dari grammar diatas dilakukan dengan cara :

- Penurunan terkiri: S → aAS → aSbAS → aabAS → aabbaS → aabbaa
- Penurunan terkanan : S → aAS → aAa → aSbAa → aAbbaa → aabbaa

Ambiguitas

Ambiguitas terjadi bila terdapat lebih dari satu pohon penurunan yang berbeda untuk memperoleh suatu string.

Misalkan terdapat tata bahasa sebagai berikut :

- S → A | B
- A → a
- B → a

Untuk memperoleh untai ‘a’ bisa terdapat dua cara penurunan sebagai berikut :

- S → A → a

- $S \rightarrow B \rightarrow a$

1.3 Cocke-Younger Kasami

Cocke-Younger Kasami (CYK) adalah salah satu algoritma parsing dengan menggunakan Chomsky Normal Form yang berasal dari Context-Free Grammar. Tujuan algoritma CYK adalah sebagai membership testing atau menunjukkan apakah suatu string atau kalimat dapat diterima dari language CFG tersebut.

Berikut merupakan tabel yang dibentuk untuk mengecek string ‘baaba’ merupakan bagian dari CFG atau bukan.

X_{15}				
X_{14}	X_{25}			
X_{13}	X_{24}	X_{35}		
X_{12}	X_{23}	X_{34}	X_{45}	
X_{11}	X_{22}	X_{33}	X_{44}	X_{55}
b	a	a	b	a

Dari tabel tersebut X_{15} adalah akar sehingga bila akar mengandung start symbol, maka string merupakan bagian dari Language tersebut.

1.4 Bahasa Pemrograman 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 juga didukung oleh komunitas yang besar.

Python adalah bahasa *interpreter* tingkat tinggi (*high-level*), dan juga *general-purpose*. Python diciptakan oleh Guido van Rossum dan dirilis pertama kali pada tahun 1991. Filosofi desain pemrograman Python mengutamakan *code readability* dengan penggunaan -nya. Python adalah bahasa multiparadigma karena mengimplementasi paradigma fungsional, imperatif, berorientasi objek, dan reflektif. *whitespace*

Dalam proses pembuatan program dari sebuah bahasa menjadi instruksi yang dapat dieksekusi oleh mesin, terdapat pemeriksaan sintaks atau kompilasi bahasa yang dibuat oleh programmer. Kompilasi ini bertujuan untuk memastikan instruksi yang dibuat oleh programmer mengikuti aturan yang sudah ditentukan oleh bahasa tersebut. Baik bahasa berjenis *interpreter* maupun *compiler*, keduanya

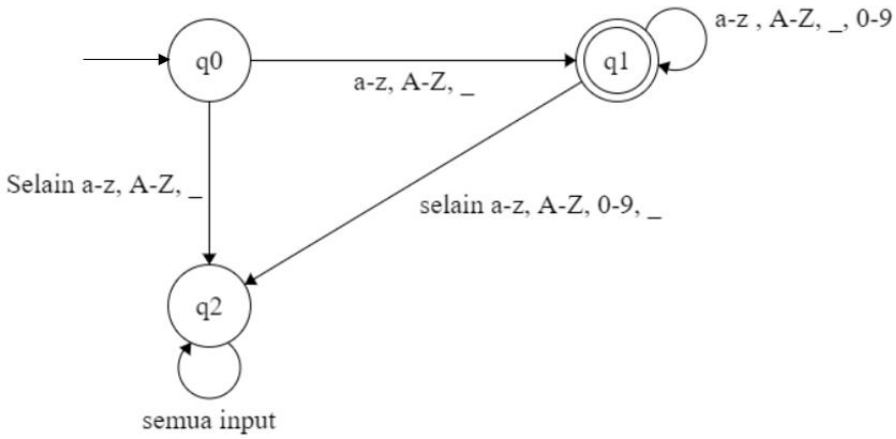
pasti melakukan pemeriksaan sintaks. Perbedaannya terletak pada apa yang dilakukan setelah proses pemeriksaan (kompilasi/*compile*) tersebut selesai dilakukan.

Compiler untuk Python dapat diimplementasikan untuk statement-statement dan syntax-syntax bawaan python menggunakan konsep CFG dan FA. Konsep CFG dapat digunakan untuk pengerjaan compiler yang mengevaluasi syntax program, sedangkan konsep FA dapat digunakan untuk nama variable. Selanjutnya, algoritma CFG harus dikonversikan ke bentuk CNF (Chomsky Normal Form). Algoritma CNF ini akan menjadi masukan ke program dalam algoritma CYK (Cocke-Younger-Kasami).

Bab 2:
Hasil FA dan CNF

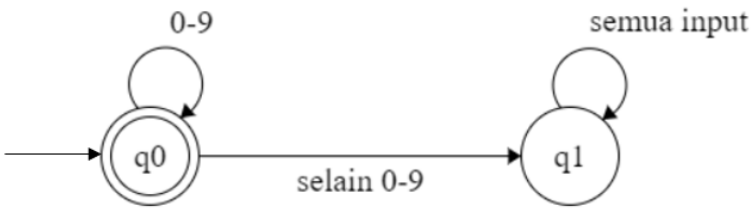
2.1 Hasil Finite Automata

FA Variable Checker



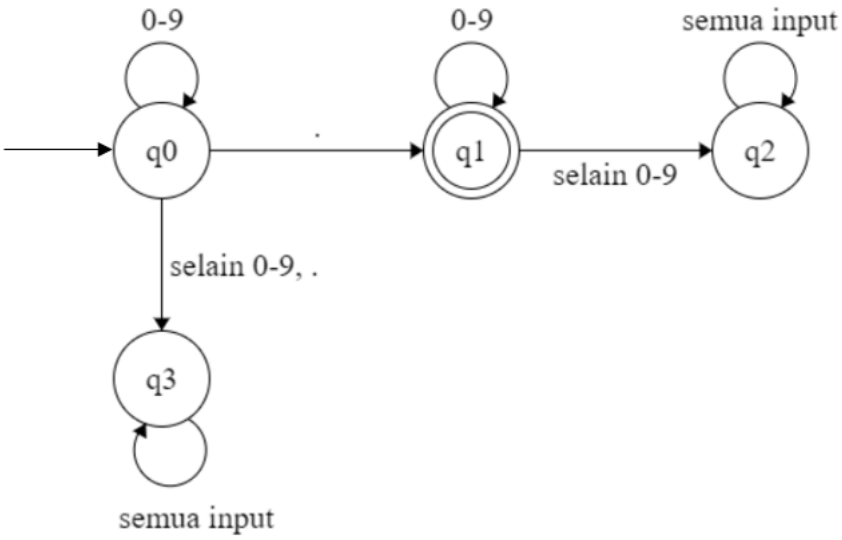
Gambar di atas merupakan Finite Automata yang digunakan untuk mengecek apakah sebuah variabel valid atau tidak. qo merupakan start state, q1 merupakan final state dimana ia hanya menerima karakter pertama berupa huruf ataupun underscore (_), untuk karakter selanjutnya dapat berupa huruf, angka, ataupun underscore (_), apabila input selain daripada itu, akan menuju state buangan yaitu q2.

FA Number Checker



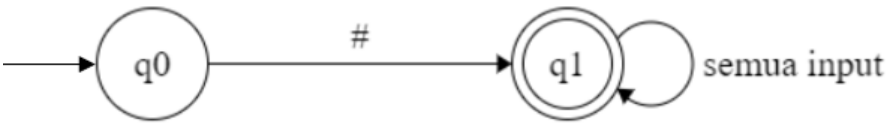
Gambar di atas merupakan Finite Automata yang digunakan untuk mengecek apakah sebuah bilangan berupa integer valid atau tidak. qo merupakan start state sekaligus final state dengan menerima input berupa angka dari 0 hingga 9, apabila input selain angka, akan menuju state buangan yaitu q1.

FA Float Checker



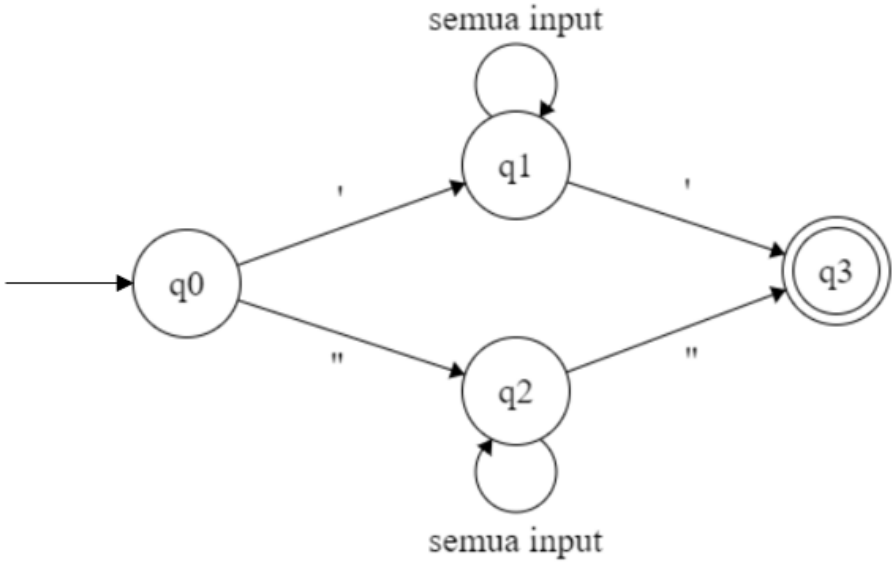
Gambar di atas merupakan Finite Automata yang digunakan untuk mengecek apakah sebuah bilangan berupa float valid atau tidak. qo merupakan start state dengan menerima input berupa angka dari 0 hingga 9, selanjutnya jika input berupa titik (.) maka akan menuju final state q1, apabila input selain angka 0-9 akan menuju state buangan yaitu q2.

FA Comment Checker



Gambar di atas merupakan Finite Automata yang digunakan untuk mengecek apabila sebuah line merupakan comment. qo merupakan start state, apabila input karakter pertamanya berupa ‘#’, maka akan menuju final state q1.

FA String Checker



Gambar di atas merupakan Finite Automata yang digunakan untuk mengecek apabila sebuah line merupakan sebuah string. qo merupakan start state dan input akan menuju final state q3 apabila diapit oleh dua buah petik satu (‘...’) atau pun diapit oleh dua buah petik dua (“...”)

2.2 Hasil CFG

Berikut ini hasil dari Context-Free Grammar yang telah kami buat dalam bentuk Chomsky Normal Form.

G = (V, T, P, S)

Non-Terminal Symbol / Variabel (V)

S	S S	FORMAT_FOR	FORMAT_DEF
FORMAT_FROM	FORMAT_IMPORT	FORMAT_CLASS	CONDITION
FORMAT_IN	FORMAT_IS	FORMAT_IS	FORMAT_WHILE
FORMAT_WITH	PASS	FORMAT_RETURN	FORMAT_AS
BREAK	CONTINUE	FORMAT_AND_OR	FORMAT_NOT
TRUE	FALSE	NONE	CONDITION
FORMAT_ELIF	FORMAT_ELSE	FORMAT_IF	FORMAT_IF_ELSE

EQUATION	FUNCTION	VARIABLE	VALUE
ARRAY	NUM	STRING	

Terminal Symbol (T)

if	else	elif	for	in
def	class	from	import	as
is	pass	return	while	Break
continue	not	True	False	None
var	raise	And	or	num
>	>=	(	)	[
]	{	}	,	:
.	<	<=	=	==
!=	+	-	*	/
%	//	@	‘	“
!=				

Productions (P)

Production	Hasil
S	S S | FORMAT_IF | FORMAT_ELSE | FORMAT_ELIF | FORMAT_FOR | FORMAT_DEF | FORMAT_CLASS | FORMAT_FROM | FORMAT_IMPORT | FORMAT_IN | FORMAT_IS | FORMAT_WHILE | FORMAT_WITH | PASS | RETURN | FORMAT_AS | BREAK | CONTINUE | FORMAT_AND_OR | FORMAT_NOT | TRUE | FALSE | NONE | EQUATION | CONDITION | EQUATION
FORMAT_IF	IF IF1
IF1	CONDITION COLON | CONDITION2 COLON | VAR COLON | ARRAY COLON | FUNCTION COLON | KALIMAT COLON | FORMAT_IN COLON | BRACKETR COLON | BRACKETS COLON | BRACKETC COLON | CONDITION IF2 | CONDITION2 IF2 | BRACKETR IF2 | BRACKETS IF2 | BRACKETC IF2 | VAR IF2 | ARRAY IF2 | FUNCTION IF2 | KALIMAT IF2 | FORMAT_IN IF2
IF2	COLON FUNCTION | COLON EQUATION | COLON EQUATION2 |

	COLON CONDITION | COLON CONDITION2 | COLON VAR | COLON VALUE | COLON NUM | COLON BRACKETR | COLON BRACKETS | COLON BRACKETC
IF	if
FORMAT_ELSE	ELSE COLON | ELSE ELSE1
ELSE1	COLON FUNCTION | COLON EQUATION | COLON EQUATION2 | COLON CONDITION | COLON CONDITION2 | COLON VAR | COLON VAR | COLON NUM
ELSE	else
FORMAT_ELIF	ELIF COLON | ELIF ELIF1
ELIF1	CONDITION COLON | CONDITION2 COLON | VAR COLON | ARRAY COLON | FUNCTION COLON | KALIMAT COLON | FORMAT_IN COLON | BRACKETR COLON | BRACKETS COLON | BRACKETC COLON | CONDITION ELIF2 | CONDITION2 ELIF2 | FORMAT_IN ELIF2 | BRACKETR ELIF2 | BRACKETS ELIF2 | BRACKETC ELIF2 | VAR ELIF2 | ARRAY ELIF2 | FUNCTION ELIF2 | KALIMAT ELIF2
ELIF2	COLON FUNCTION | COLON EQUATION | COLON EQUATION2 | COLON CONDITION | COLON CONDITION2 | COLON VAR | COLON VALUE | COLON NUM
ELIF	elif
FORMAT_IF_ELSE	EQUATION IF_ELSE1 | VAR IF_ELSE1 | STRING IF_ELSE1 | NUM IF_ELSE1 | ARRAY IF_ELSE1 | FUNCTION IF_ELSE1 | KALIMAT IF_ELSE1 | FLOAT IF_ELSE1
IF_ELSE1	IF IF_ELSE2
IF_ELSE2	CONDITION IF_ELSE3 | CONDITION2 IF_ELSE3
IF_ELSE3	ELSE VALUE | ELSE VAR | ELSE NUM | ELSE STRING | ELSE ARRAY | ELSE FUNCTION | ELSE FLOAT | ELSE KALIMAT | ELSE TRUE | ELSE FALSE
FORMAT_FOR	FOR FOR1

FOR1	VARIABLE FOR2 | VAR FOR2
FOR2	IN FOR3 | IN FOR4
FOR3	VAR COLON | VALUE COLON | FUNCTION COLON | BRACKETR COLON | BRACKETS COLON | BRACKETC COLON
FOR4	FOR3 FUNTION | FOR3 EQUATION | FOR3 EQUATION2 | FOR3 CONDITION | FOR3 CONDITION2 | FOR3 ARRAY | FOR3 KALIMAT
FOR	for
FORMAT_DEF	DEF DEF1 | DEF DEF2
DEF1	FUNCTION COLON
DEF2	DEF1 FUNTION | DEF1 KALIMAT | DEF1 EQUATION | DEF1 EQUATION2 | DEF1 CONDITION | DEF1 CONDITION2 | DEF1 VALUE | DEF1 FORMAT_RETURN | DEF1 VAR | DEF1 NUM | DEF1 STRING | DEF1 KALIMAT | DEF1 BRACKETC | DEF1 BRACKETR | DEF1 BRACKETS
DEF	def
FORMAT_CLASS	CLASS CLASS1
CLASS1	VAR COLON
CLASS	class
FORMAT_FROM	FROM FROM1
FROM1	VAR FORMAT_IMPORT
FROM	from
FORMAT_IMPORT	IMPORT VARIABEL | IMPORT IMPORT1 | IMPORT VAR
IMPORT1	VARIABEL IMPORT2 | VAR IMPORT2
IMPORT2	AS VARIABEL | AS VAR
IMPORT	import
AS	as
FORMAT_IN	VALUE IN1 | VARIABLE IN1 | VAR IN1
IN1	IN VAR | IN VALUE | IN BRACKETS | IN BRACKETC
IN	in
FORMAT_IS	BRACKETS OBJECT_IS_S | BRACKETC OBJECT_IS_C | TRUE OBJECT_IS | FALSE OBJECT_IS | NONE OBJECT_IS

OBJECT_IS	IS EQUATION2 | IS BRACKETS | IS BRACKETC | IS BRACKETR | IS TRUE | IS FALSE | IS NONE | IS ARRAY
IS	is | =
PASS	pass
FORMAT_RETURN	RETURN VALUE | RETURN VAR | RETURN NUM | RETURN ARRAY | RETURN FUNCTION | RETURN CONDITION | RETURN CONDITION2 | RETURN BRACKETR | RETURN BRACKETS | RETURN BRACKETC
RETURN	return
FORMAT_WHILE	WHILE WHILE1 | WHILE WHILE2
WHILE1	VAR COLON | NUM COLON | STRING COLON | CONDITION COLON | CONDITION2 COLON | EQUATION2 COLON | EQUATION2 COLON | FUNCTION COLON | KALIMAT COLON
WHILE2	WHILE1 FUNCTION | WHILE1 KALIMAT | WHILE1 EQUATION | WHILE1 EQUATION2 | WHILE1 VAR | WHILE1 VALUE | WHILE1 NUM
WHILE	while
FORMAT_WITH	WITH WITH1
WITH1	VALUE FORMAT_AS_W1 | VAR FORMAT_AS_W1 | ARRAY FORMAT_AS_W1 | FUNCTION FORMAT_AS_W1 | KALIMAT FORMAT_AS_W1 | BRACKETR FORMAT_AS_W1 | BRACKETS FORMAT_AS_W1 | BRACKETC FORMAT_AS_W1 | VALUE COLON | VAR COLON | ARRAY COLON | NUM COLON | BRACKETR COLON | BRACKETS COLON | BRACKETC COLON | KALIMAT COLON
WITH	with
BRACKETR	ROUNDO LIST_VALUE_R | ROUNDO ROUNDC
BRACKETS	SQUAREO LIST_VALUE_S | SQUAREO SQUAREC
BRACKETC	CURLYO LIST_VALUE_C | CURLYO CURLYC

LIST_VALUE_R	VALUE ROUNDC | VAR ROUNDC | NUM ROUNDC | STRING ROUNDC | ARRAY ROUNDC | FUNCTION ROUNDC | EQUATION ROUNDC | KALIMAT ROUNDC | BRACKETR ROUNDC | BRACKETS ROUNDC | BRACKETC ROUNDC
LIST_VALUE_S	VALUE SQUAREC | VAR SQUAREC | NUM SQUAREC | ARRAY SQUAREC | STRING SQUAREC | FUNCTION SQUAREC | EQUATION SQUAREC | KALIMAT SQUAREC | VALUE LIST_VALUE_S1 | VAR LIST_VALUE_S1 | NUM LIST_VALUE_S1 | ARRAY LIST_VALUE_S1 | FUNCTION LIST_VALUE_S1 | EQUATION LIST_VALUE_S1 | EQUATION2 LIST_VALUE_S1 | VAR FOR_LIST | NUM FOR_LIST | ARRAY FOR_LIST | FLOAT FOR_LIST
LIST_VALUE_S1	COLON SQUAREC | COLON LIST_VALUE_S
LIST_VALUE_C	NUM CURLYC | VAR CURLYC | ARRAY CURLYC | FLOAT CURLYC | FUNCTION CURLYC | BRACKETC CURLYC | BRACKETR CURLY | BRACKETS CURLYC | VALUE CURLYC | VARIABLE CURLYC | NUM CURLYC | STRING CURLYC | KALIMAT CURLYC
FOR_LIST	FOR FOR_LIST1
FOR_LIST1	VARIABLE FOR_LIST2 | VAR FOR_LIST2
FOR_LIST2	IN FOR_LIST3
FOR_LIST3	VAR SQUAREC | FUNCTION SQUAREC
FORMAT_AS_W1	AS FORMAT_AS_W2
FORMAT_AS_W2	VARIABLE COLON | BRACKETR COLON | BRACKETS COLON | BRACKETC COLON | VAR COLON | STRING COLON
BREAK	break
CONTINUE	continue

FORMAT_AND_OR	VARIABEL AND1_OR1 | BRACKETR AND1_OR1 | BRACKETS AND1_OR1 | BRACKETC AND1_OR1 | FUNCTION AND1_OR1 | CONDITION2 AND1_OR1 | ARRAY AND1_OR1
AND1_OR1	AND1_OR1 AND1_OR1 | AND VAR | AND BRACKETR | AND BRACKETS | AND BRACKETC | AND FUNCTION | AND CONDITION2 | AND ARRAY | AND KALIMAT | OR CONDITION2 | OR VAR | OR BRACKETR | OR BRACKETS | OR BRACKETC | OR FUNCTION | OR ARRAY | OR KALIMAT
AND	And
OR	or
FORMAT_NOT	NOT VARIABEL | NOT VAR | NOT KALIMAT | NOT ARRAY | NOT FUNCTION | NOT BRACKETR | NOT BRACKETS | NOT BRACKETC
NOT	not
TRUE	True
FALSE	False
NONE	None
VARIABLE	VAR VARIABLE1
VARIABLE1	VARIABLE1 VARIABLE1 | COMMA VAR
VALUE	VAR VALUE1 | NUM VALUE1 | STRING VALUE1 | EQUATION VALUE1 | EQUATION2 VALUE1 | ARRAY VALUE1
VALUE1	VALUE1 VALUE1 | COMMA NUM | COMMA VAR | COMMA STRING | COMMA EQUATION | COMMA EQUATION2 | COMMA ARRAY
ARRAY	VAR BRACKETS | VAR ARRAY1
ARRAY1	ARRAY1 BRACKETS | ARRAY1 ARRAY1 | BRACKETS BRACKETS
KALIMAT	VAR KALIMAT1 | FUNCTION KALIMAT1
KALIMAT1	DOT VAR | DOT KALIMAT | DOT FUNCTION
VAR	var
FUNCTION	VAR FUNCTION1

FUNCTION1	ROUNDO FUNCTION2 | ROUNDO ROUNDC
FUNCTION2	STRING ROUNDC | VALUE ROUNDC | VAR ROUNDC | NUM ROUNDC | ARRAY ROUNDC | FUNCTION ROUNDC | EQUATION ROUNDC | EQUATION2 ROUNDC
CONDITION	ROUNDO CONDITION1
CONDITION1	CONDITION2 ROUNDC
CONDITION2	True | False | VAR COMP1 | NUM COMP1 | ARRAY COMP1 | FUNCTION COMP1 | KALIMAT COMP1 | STRING COMP1| VAR AND1_OR1| CONDITION2 AND1_OR1 | KALIMAT AND1_OR1 | NOT VAR | NOT FUNCTION | NOT BRACKETR | NOT BRACKETS | NOT BRACKETC | NOT ARRAY | NOT KALIMAT | CONDITION2 FORMAT_NOT
COMP1	COMP1 COMP1 | COMPARE VAR | COMPARE NUM | COMPARE FUNCTION | COMPARE EQUATION2 | COMPARE EQUATIONR | COMPARE ARRAY | COMPARE KALIMAT | COMPARE STRING
EQUATION	EQUATION2 | VAR EQUATION1 | ARRAY EQUATION1 | VARIABLE EQUATION1
EQUATION1	ASSIGN EQUATION2 | ASSIGN EQUATIONR | ASSIGN VALUE | ASSIGN VAR | ASSIGN BRACKETS | ASSIGN BRACKETR | ASSIGN BRACKETC | ASSIGN NUM | ASSIGN TRUE | ASSIGN FALSE | ASSIGN NONE | ASSIGN FUNCTION | ASSIGN STRING | ASSIGN FORMAT_IF_ELSE | ASSIGN ARRAY | ASSIGN KALIMAT
EQUATIONR	EQUATIONC ROUNDC
EQUATIONC	ROUNDO EQUATION2
EQUATION2	VAR EQUATION3 | NUM EQUATION3 | BRACKETS EQUATION3 | BRACKETC EQUATION3 | BRACKETR EQUATION3 | STRING EQUATION3 |

	FUNCTION EQUATION3 | ARRAY EQUATION3 | KALIMAT EQUATION3
EQUATION3	EQUATION3 EQUATION3 | OP VAR | OP NUM | OP BRACKETS | OP BRACKETC | OP BRACKETR | OP FUNCTION | OP ARRAY | OP KALIMAT
NUM	num
STRING	PETIK1 PETIK1 | PETIK2 PETIK2
PETIK1	‘
PETIK2	“
ROUNDO	(
ROUNDC	)
SQUAREO	[
SQUEREC	]
CURLYO	{
CURLYC	}
COMMA	,
COLON	:
COMPARE	> | < | <= | = | == | !=
OP	+ | - | * | / | % | ** | // | @
PLUS	+
DOT	.
NEG	-

Bab 3 : Implementasi dan Pengujian

3.1 Spesifikasi Teknis Program

File Checker.py

File Checker.py berisi mengenai prosedur dan fungsi yang berguna untuk mengecek dan mengatasi kasus-kasus berupa string, variable, number, dan comment.

No	Prosedur/Fungsi	Tujuan
1	string_checker	Mengatasi kasus input berupa string yaitu dengan parameter diantara tanda petik satu (‘) atau tanda petik dua (“) dengan cara menghapus string yang ada diantara kedua tanda tersebut
2	comment_checker	Mengatasi kasus input berupa comment yaitu dengan parameter tanda pagar (#) dengan cara menghapus string yang ada setelah tanda tersebut
3	multicomment_checker	Mengatasi kasus input berupa multipleline comment yaitu dengan parameter diantara tanda petik satu tiga kali (‘’’) dengan cara menghapus string yang ada diantara kedua tanda tersebut
4	variable_checker	Mengecek apakah sebuah rule/line merupakan sebuah variabel dengan cara memastikan karakter pertamanya berupa huruf ataupun underscore(_) serta karakter-karakter selanjutnya berupa huruf atau angka atau tanda underscore (_)
5	number_checker	Mengecek apakah sebuah rule/line merupakan sebuah bilangan dengan memastikan bahwa setiap karakternya berupa angka
6	float_checker	Memastikan bahwa sebuah line merupakan float yang valid dengan melihat jumlah dan letak titik diantara angka
7	replace_operator	Memastikan bahwa sebuah line memiliki operator yang valid
8	joining_line_checker	Mengatasi kasus input berupa *joining line* yaitu dengan paramter *backslash* (\) dengan cara menggabungkan line tersebut dengan line dibawahnya, dan menghapus line dibawahnya.

File line_parser.py

File line_parser.py berisi fungsi yang berguna untuk melakukan proses pemisahan pada tiap line testcase. File ini berisi fungsi pada tabel berikut ini

No	Prosedur/Fungsi	Tujuan
1	symbols_parser	Memisahkan simbol-simbol yang ada pada input menjadi satu bagian list tersendiri
2	rules_parser	Mengonversi CFG dari text file 'CNF_GABUNGAN.txt' ke dalam python dengan tipe data dictionary

File cyk_algorithm.py

File cyk_algorithm.py berisi fungsi yang berguna untuk

No	Prosedur/Fungsi	Tujuan
1	cyk_algorithm	Menjalankan algoritma cyk, yaitu melakukan pengecekan terhadap suatu kalimat apakah kalimat tersebut dapat dibentuk dengan CFG yang ada

File main.py

File main.py merupakan source code yang berisi main program sebagai compiler bahasa Python. File ini mengimport fungsi/prosedur yang telah dijelaskan sebelumnya yang digunakan untuk memeriksa tiap testcase yang diinput. Pada file ini, program akan membaca CNF dari suatu file yang berisi daftar rules dengan menggunakan rules_parser. Program selanjutnya akan membaca input testcase yang akan diperiksa dari pengguna. Setelah memastikan tiap line testcase menggunakan checker yang ada, tiap list line tersebut akan dibaca dalam cyk-algorithm. Input berhasil lolos compile apabila akar dari cyk-table berisi start symbol.

3.2 Pengujian Program

Pada Bagian ini, akan dilakukan beberapa uji kasus terhadap compiler Python yang telah kami buat. Berikut ini pengujian program yang telah kami lakukan :

Input dan Output Sederhana

Gambar di bawah merupakan potongan source code python sederhana yang akan dilakukan pengecekan menggunakan compiler python yang telah kami buat

```
A = int(input())
B = int(input())
C = int(input())
D = A + B - B*C
Hello_World = "Yuk Bisa Yuk"
print(Hello_World)
print("Keos Banget")
print("Bismillah" + "Jalan" + "Aamiin")
```

Setelah melakukan run program compiler python kami, berikut tampilah hasil yang diperoleh :

```
\Tubes\TubesTBFO\src> py main.py testcase1.py
Accepted
```

Dari hasil percobaan pertama, hasil yang dikeluarkan sesuai dengan yang diharapkan, yaitu “Accepted” yang berarti compile berhasil. Pada percobaan pertama ini, grammar production yang sering digunakan adalah VAR, EQUATION, ASSIGN, dan FUNCTION. Program akan melakukan parsing menggunakan CYK hingga memperoleh proses akhir yang menunjukkan start symbol.

Percabangan

Gambar di bawah merupakan potongan source code python sederhana yang akan dilakukan pengecekan menggunakan compiler python yang telah kami buat

```
Botol = str(input())
Piring = True
if (Botol == "minum"):
    print ("Berhasil!")
elif (Botol == "Kaca"):
    print ("Gagal!")

if (Piring == True):
    Mangkok = True
else :
    Mangkok = False

a = int(input())
if (a>=10):
    a += 1
elif (a>=2 and a<10):
    a = a - 5
else :
    a = 0
```

Setelah melakukan run program compiler python kami, berikut tampilah hasil yang diperoleh :

```
\Tubes\TubesTBFO\src> py main.py testcase2.py
Accepted
```

Dari hasil percobaan pertama, hasil yang dikeluarkan sesuai dengan yang diharapkan, yaitu “Accepted” yang berarti compile berhasil. Pada percobaan kedua ini, grammar production yang sering digunakan adalah VAR, EQUATION, ASSIGN, FORMAT_IF, FORMAT_ELIF, FORMAT_ELSE, COMPARE. Program akan melakukan parsing menggunakan CYK hingga memperoleh proses akhir yang menunjukkan start symbol.

Looping

Gambar di bawah merupakan potongan source code python sederhana yang akan dilakukan pengecekan menggunakan compiler python yang telah kami buat

```
for i in range (10):
    for j in range(20):
        Test[i][j] = int(input())

x = 0
while (x < len(Test)):
    if (y == 10):
        print (x)
        x += 1
        continue
    else :
        break

for i in range (9,3,2) :
    print(Nilai)
```

Setelah melakukan run program compiler python kami, berikut tampilah hasil yang diperoleh :

```
\Tubes\TubesTBFO\src> py main.py testcase3.py
Accepted
```

Dari hasil percobaan pertama, hasil yang dikeluarkan sesuai dengan yang diharapkan, yaitu “Accepted” yang berarti compile berhasil. Pada percobaan ketiga ini, grammar production yang sering digunakan adalah VAR, EQUATION, ASSIGN, FORMAT_FOR, FORMAT_ELIF, WHILE, ARRAY. Program akan melakukan parsing menggunakan CYK hingga memperoleh proses akhir yang menunjukkan start symbol.

Class, Import, Def, Comment

Gambar di bawah merupakan potongan source code python sederhana yang akan dilakukan pengecekan menggunakan compiler python yang telah kami buat

```
import pandas as pd
import math
from matplotlib import numpy as np

#Ini Def Hitung
def hitung():
    print ("Hitung Berhasil")

#Ini Def Tulis
def tulis(X,Y):
    X = Y**2
    math.sqrt(X)
    return X

'''
Ini untuk Class Baca
'''
class baca :
    def baca2():
        print("Baca Berhasil")
    def baca3(Buku):
        Buku = False
        return Buku
```

Setelah melakukan run program compiler python kami, berikut tampilah hasil yang diperoleh :

```
\Tubes\TubesTBFO\src> py main.py testcase4.py
Accepted
```

Dari hasil percobaan pertama, hasil yang dikeluarkan sesuai dengan yang diharapkan, yaitu “Accepted” yang berarti compile berhasil. Pada percobaan keempat ini, grammar production yang sering digunakan adalah VAR, EQUATION, ASSIGN, FORMAT_FROM, FUNCTION, FORMAT_FOR, CLASS, RETURN. Program akan melakukan parsing menggunakan CYK hingga memperoleh proses akhir yang menunjukkan start symbol.

With, Raise

Gambar di bawah merupakan potongan source code python sederhana yang akan dilakukan pengecekan menggunakan compiler python yang telah kami buat

```
with open('example.txt', 'w') as my_file:
    my_file.write('Hello world!')

def reciprocal(num):
    try:
        r = 1/num
    except:
        print('Exception caught')
        return
    return r

print(reciprocal(10))
print(reciprocal(0))
```

Setelah melakukan run program compiler python kami, berikut tampilah hasil yang diperoleh :

```
ubes\TubesTBFO\src> py main.py testcase5.py
Accepted
```

Dari hasil percobaan pertama, hasil yang dikeluarkan sesuai dengan yang diharapkan, yaitu “Accepted” yang berarti compile berhasil. Pada percobaan kelima ini, grammar production yang sering digunakan adalah VAR, WITH, VALUE, KALIMAT, FUNCTION, dan RAISE. Program akan melakukan parsing menggunakan CYK hingga memperoleh proses akhir yang menunjukkan start symbol.

Test Case Salah

Gambar di bawah merupakan potongan source code python sederhana yang akan dilakukan pengecekan menggunakan compiler python yang telah kami buat

```
a = int(input())
if (a>=10)::
    a += 1
elif (a>=2 and a<10):
    a = a - 5
else :
    a = 0
```

Setelah melakukan run program compiler python kami, berikut tampilah hasil yang diperoleh :

```
Syntax Error
Terjadi kesalahan ekspresi pada line 2: "if (a>=10)::"
```

Dari hasil percobaan pertama, hasil yang dikeluarkan sesuai dengan yang diharapkan, yaitu “Syntax Error” yang berarti compile gagal. Selain itu, juga ditampilkan line letak dimana gagal meng-compile.

Link Github

Link Github : https://github.com/Fnhafiz/TubesTBFO.git

Pembagian Tugas

No	NIM/ Nama	Pembagian Tugas
1	13520027/ Farhan Hafiz	Membuat variable_checker, number_checker, float_checker, Membuat CFG, dan mengerjakan laporan
2	13520030/ Fitrah Ramadhani Nugroho	Membuat fungsi cyk_algorithm, rules_parser, membuat dan menggabungkan CFG, dan mengerjakan laporan.
3	13520037/ Muhammad Akmal Arifin	Membuat fungsi cyk_algorithm, symbols_parser, string_checker, comment_checker, multicomment_checker, joinning_line_checker, membuat CFG dan file main.py

Referensi

John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman, Introduction To Automata Theory Languanges, and Computation Third Edition,Pearson, 2014.

https://www.programiz.com/python-programming/keyword-list