

LAPORAN TUGAS BESAR
PARSER BAHASA PYTHON
IF2124 Teori Bahasa Formal dan Otomata



Kelompok 2 – Nasgor Jawara

MUHAMMAD FIKRI RANJABI	13520002
GEDE PRASIDHA BHAWARNAWA	13520004
AHMAD ALFANI HANDOYO	13520023

TEKNIK INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

2021

DAFTAR ISI

Bab 1 Teori Dasar	3
1.1 Finite Automata.....	3
1.2. Context Free Grammar	3
1.3. Chomsky Normal Form.....	4
1.4. Cocke-Younger Kasami	4
1.5. Bahasa Pemrograman Python.....	4
Bab II Analisis Finite Automata dan Context Free Grammar.....	6
2.1 Finite Automata	6
2.2 Context Free Grammar	6
Bab III Implementasi dan Pengujian.....	16
i. Spesifikasi Struktur Data, Fungsi, dan Prosedur	16
2.1. File cyk_parser.py	16
2.2. File CFG2CNF.py	16
2.3. File helper.py.....	17
2.4. File py_parser.py	17
2.5. File parserprogram.py	17
2.6. File main.py.....	17
ii. Screenshot Test Case	18
Bab IV Penutup.....	28
4.1 Kesimpulan	28
4.2 Saran	28
4.3 Refleksi	28
Bab V Lampiran	29

Bab 1

Teori Dasar

1.1 Finite Automata

Finite automata atau otomata terbatas adalah suatu konsep mesin abstrak yang dapat mengolah dan mengenali pola-pola sederhana. Dalam istilah otomata, pola-pola ini disebut sebagai regular languages. Sebuah finite automata terdiri atas kumpulan states, input, serta transition function. Paling tidak, di dalam suatu finite automata terdapat start state, atau state dimana finite automata membaca string input, dan final state, yang mana menunjukkan bahwa string habis dibaca oleh finite automata dan string tersebut diterima oleh finite automata tersebut. Suatu finite automata, selanjutnya disebut FA, dibagi menjadi dua jenis berdasarkan transition functionnya. Transition function adalah suatu kumpulan fungsi/aturan yang menunjukkan ke state mana FA harus berada berdasarkan input yang diterima oleh FA. Terdapat Deterministic FA dan Non-Deterministic FA. Deterministic FA hanya memperbolehkan suatu FA untuk berada pada satu state di suatu waktu, sementara Non-Deterministic FA memperbolehkan suatu FA untuk berada pada beberapa state pada suatu waktu dan berhenti ketika salah satu state yang ditempati adalah final state.

Finite Automata (FA) baik itu deterministic maupun non-deterministic dapat dinyatakan secara formal sebagai berikut:

$$FA = (Q, \Sigma, \beta, q_0, F)$$

Dengan nilai Q sebagai semua state yang terdapat di dalam FA, Σ sebagai symbol-symbol input pada FA, β sebagai list dari transition function atau fungsi perubahan state pada FA berdasarkan nilai input yang masuk ke FA, q_0 adalah start state dari FA dan F adalah list dari semua final state pada FA.

1.2. Context Free Grammar

Context Free Grammar adalah notasi formal yang digunakan untuk mengekspresikan suatu language secara rekursif. Sebuah grammar terdiri dari satu atau lebih variable yang menggambarkan language. Context Free Grammar memiliki struktur sebagai berikut.

$$G = (V, T, P, S)$$

Dengan G adalah grammar, V berupa variable, T berupa terminal, P berupa himpunan produksi, dan S merupakan start symbol.

$$E \Rightarrow E * E \Rightarrow I * E \Rightarrow a * E \Rightarrow$$

$$a * (E) \Rightarrow a * (E + E) \Rightarrow a * (I + E) \Rightarrow a * (a + E) \Rightarrow$$

$$a * (a + I) \Rightarrow a * (a + I0) \Rightarrow a * (a + I00) \Rightarrow a * (a + b00)$$

Penurunan CFG dapat dilakukan secara Leftmost atau Rightmost Derivations. Setiap langkah penurunan akan menghasilkan string terbatas yang berada dalam rule produksi.

Gambar di atas adalah sebuah contoh penurunan Context Free Grammar untuk mengecek apakah suatu language dapat diterima oleh grammar tersebut.

1.3. Chomsky Normal Form

Chomsky Normal Form adalah bentuk dari sebuah Context Free Grammar yang didapatkan dengan menerapkan beberapa rule sebagai berikut ke dalam grammar:

1. Eliminasi Useless Symbol

Symbol dapat berupa variable atau terminal yang tidak muncul pada penurunan grammar dari start symbol.

2. Eliminasi epsilon Production

epsilon Production dengan bentuk $A \rightarrow \epsilon$ dalam variabel A.

3. Eliminasi Unit Production

langkah ini dilakukan agar tidak ada variable yang berulang.

Setelah ketiga langkah di atas dilakukan, kemudian kita ubah bentuk production pada grammar menjadi sebagai berikut:

1. $A \rightarrow BC$, dimana A, B, dan C adalah variable
2. $A \rightarrow a$, dengan A sebagai variable dan a sebagai terminal

1.4. Cocke-Younger Kasami

Cocke-Younger Kasami merupakan sebuah algoritma yang dapat digunakan untuk mengetes apakah suatu string diterima sebuah CNF atau tidak.

{S,A,C}				
-	{S,A,C}			
-	{B}	{B}		
{S,A}	{B}	{S,C}	{S,A}	
{B}	{A,C}	{A,C}	{B}	{A,C}
b	a	a	b	a

The table for string *baaba* constructed by the CYK algorithm

Algoritma CYK bergerak secara bottom up dengan baris pertama adalah terminal yang sesuai dengan input. Pengisian CYK dilakukan baris per baris sampai ke baris paling atas yang menandakan bahwa variable yang diisi sudah mewakili keseluruhan dari input. Jika variable paling atas (X15) memiliki Start Symbol sebagai elemennya, maka sudah dipastikan bahwa string masukan diterima dan berlaku sebaliknya.

1.5. Bahasa Pemrograman Python

Berikut adalah syntax python yang perlu diperhatikan dalam pembuatan FA dan CFG:

- Input ('prompt') *prompt berupa teks bebas sebagai pesan
- print(x), x bisa berupa string dan variable
- If (boolean): *boolean adalah kasus yang menghasilkan true atau false
- Elif (boolean): *boolean adalah kasus yang menghasilkan true atau false
- Else:
- For variable in x: *x bisa berupa string, variable, atau range
- While (boolean):
- Def variable (x): *x bisa berupa string atau variable
- Import variable
- Import variable as variable
- From variable import variable as variable
- Class variable:

Bab II

Analisis Finite Automata dan Context Free Grammar

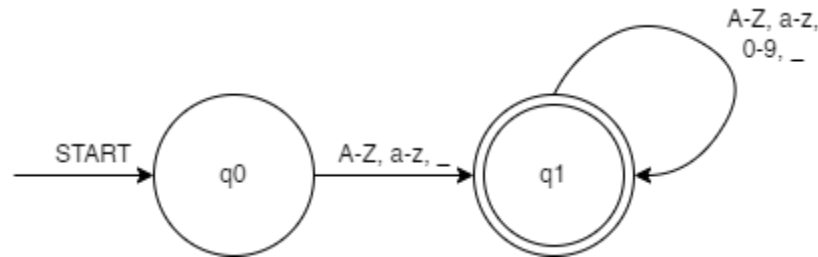
2.1 Finite Automata

Dalam implementasi Finite Automata untuk mengecek apakah penamaan variabel sudah sesuai, digunakan regex dengan library re yang sudah disediakan oleh Python. Regex yang digunakan untuk menghasilkan variabel yaitu `[A-Za-z_][A-Za-z_0-9]*` akan ekivalen dengan Finite Automata berikut. Bila string yang dimasukkan mencapai final state yaitu q_1 , maka penamaan variabel sudah sesuai. Secara definisi tidak formal, string yang menghasilkan variabel harus mempunyai awalan karakter A-Z, a-z, atau `_`, namun tidak bisa 0-9.

Bahasa dari FA dideskripsikan sebagai suatu Non-Deterministic Finite Automata (NFA):

$$A = (\{q_0, q_1\}, \{A-Z, a-z, 0-9, _ \}, \delta, q_0, \{q_1\})$$

Diagram Transisi:



Tabel Transisi:

State	A-Z	a-z	0-9	_
q₀	q ₁	q ₁	∅	q ₁
*q₁	q ₁	q ₁	q ₁	q ₁

2.2 Context Free Grammar

Berikut adalah Context Free Grammar yang dibuat untuk nantinya dikonversi menjadi bentuk CNF.

$$G = (V, T, P, S)$$

Non-Terminal Symbol/Variable (V)

START	BACKVARIABLE	ALLNUMBERS	NEGATIVEFLOAT
ASSIGN	VALUE	LIST	OPERATORS
ALPHABET	INTEGER	SET	COMPARISON
WHOLEWORD	STRING	DICTIONARY	ASSIGNMENT
ESCAPE	FLOAT	TUPLE	FOR
VARIABLE	BOOLEAN	NEGATIVEINTEGER	WHILE
PRINT	WHILE	LOOPVARIABLE	ELSE
ASSIGNMENT	FOR	DEL	INPUT
BREAK	IMPORT	IF	RETURN
PASS	CLASS	ELIF	TYPECASTING
TEXT	FROM	IFONLINE	

Terminal Symbol (T)

input	bool	bytes	!	>
print	set	'	=	<
exit	dict	“	%	and
#	range	(/	or
type	tuple)	*	not
global	list	[-	is
str	complex]	+	in
int	float	{	}	True
False	pass	break	from	as
none	elif	continue	dir	open
for	else	def	import	read
while	if	return	class	readline
close	number	string	,	space
write	;	variable	.	_
:				

Production (P)

LHS	RHS
START	START START VARIABLE ASSIGN ASSIGNMENT IF ELIF ELSE PRINT FOR WHILE DEF CLASS IMPORT FROM COMMENT VARIABLE ASSIGN ITERABLES VARIABLE ASSIGN IFONELINE RETURN
ASSIGN	= + = - = * = + = - = * = / = % = / / = * * =
SPACE	space
TEXT	TEXT TEXT string
STRING	' TEXT ' " TEXT " STRING + STRING
INTEGER	number
VARIABLE	variable
COMPARISON	< > < = > = ! = = =
BOOLEAN	(BOOLEAN) True False BOOLEAN and BOOLEAN BOOLEAN or BOOLEAN not BOOLEAN VALUE is VALUE VALUE is not VALUE VALUE COMPARISON VALUE
VALUE	(VALUE) INTEGER STRING BOOLEAN VALUE COMPARISON VALUE VALUE OPERATORS VALUE;

	OPERATORS -> / / + - * * % * /
PRINT	print (VALUE)
COMMENT	# TEXT " " " TEXT " " " ' ' ' TEXT ' ' '
ASSIGNMENT	VARIABLE VALUE VALUE OPERATORS VALUE INPUT METHOD
FOR	for VARIABLE in range (INTEGER) : for VARIABLE in range (INTEGER , INTEGER) : for VARIABLE in range (INTEGER , INTEGER , INTEGER) : for VARIABLE in VARIABLE : for VARIABLE in range (INTEGER) : START for VARIABLE in range (INTEGER , INTEGER) : START for VARIABLE in range (INTEGER , INTEGER , INTEGER) : START for VARIABLE in VARIABLE : START for VARIABLE in STRING : START
BREAK	break
PASS	pass
CONTINUE	continue
WHILE	while BOOLEAN : while BOOLEAN : START
IMPORT	import VARIABLE import VARIABLE as VARIABLE
FROM	from VARIABLE IMPORT
CLASS	class VARIABLE
LOOPVARIABLE	LOOPVARIABLE , LOOPVARIABLE VARIABLE
DEF	def VARIABLE (LOOPVARIABLE) : def VARIABLE () :
IFONELINE	VALUE if BOOLEAN VALUE if BOOLEAN else VALUE VALUE if BOOLEAN else IFONELINE; IF -> if BOOLEAN :
ELIF	elif BOOLEAN :
ELSE	else :
INPUT	input (STRING) TYPECASTING (INPUT) input ()

RETURN	return return BOOLEAN return VARIABLE return STRING return INTEGER
TYPECASTING	float int complex
ITERABLES	LIST TUPLE SET DICTIONARY
ELEMENT	ELEMENT , ELEMENT STRING ALLNUMBERS
LIST	[ELEMENT] [LIST]
TUPLE	(ELEMENT) (TUPLE)
SET	{ ELEMENT } { SET }
DICTELEMENT	DICTELEMENT , DICTELEMENT " VARIABLE " : STRING " VARIABLE " : DICTIONARY
DICTIONARY	{ DICTELEMENT }

Start Symbol (S): S

2.3 Chomsky Normal Form

Non-Terminal Symbol/Variable (V)

START	SPACE	INTEGER	S4	M2
A1	TEXT	VARIABLE	J1	N1
B1	Y4	COMPARISON	R4	VALUE
C1	STRING	W4	K1	O1
ASSIGN	F1	U4	Q4	P1
D1	X4	T4	P4	Q1
E1	G1	BOOLEAN	L1	OPERATORS
Z4	H1	I1	M1	O4
PRINT	T5	W6	Y5	A14
R1	ASSIGNMENT	I4	Y6	A15
R2	U1	X1	Y7	A16
N4	M4	X2	Y8	A17
COMMENT	L4	X3	Y9	B11
S1	K4	X5	Y10	B12
S2	J4	X6	Z1	B13
S3	FOR	X7	Z2	B14
S5	W1	X8	Z3	B15
T1	W2	Y1	A11	B16
T2	W3	Y2	A12	B17
T3	W5	Y3	A13	B18
B19	D11	G4	I11	M11

C11	D12	WHILE	C4	M12
C12	D13	F11	CLASS	M13
C13	D14	G11	J11	A4
C14	E11	G12	LOOPVARIABLE	IFONELINE
C15	E12	F4	K11	N11
C16	E13	IMPORT	B4	O11
C17	E14	E4	DEF	O12
C18	BREAK	H11	L11	O13
C19	PASS	H12	L12	P11
C110	H4	D4	L13	P12
C111	CONTINUE	FROM	L14	P13
IF	S12	ELEMENT	Z11	E21
Q11	T11	W11	A21	E22
ELIF	T12	U3	SET	E23
R11	U11	LIST	B21	F21
ELSE	RETURN	X11	C21	F22
INPUT	TYPECASTING	Y11	DICTELEMENT	F23
S11	ITERABLES	TUPLE	D21	DICTIONARY
G21	S0			

Terminal Symbol (T)

input	bool	bytes	!	>
print	set	'	=	<
exit	dict	“	%	and
#	range	(/	or
type	tuple)	*	not
global	list	[-	is
str	complex]	+	in
int	float	{	}	True
False	pass	break	from	as
none	elif	continue	dir	open
for	else	def	import	read
while	if	return	class	readline
close	number	string	,	space
write	;	variable	.	—
:				

Production (P)

LHS	RHS
START	START START VARIABLE A1 VARIABLE B1 VARIABLE C1 A4 Q11 Y3 R11 Z3 J4 O4 R1 K4 W1 K4 X1 K4 Y1 K4 Z1 K4 A11 K4 B11 K4 C11 K4 D11 K4 E11 G4 F11 G4 G11 B4

	L11 B4 M11 C4 J11 F4 VARIABLE F4 H11 D4 I11 N4 TEXT X4 S1 Y4 T1 return RETURN BOOLEAN RETURN VARIABLE RETURN STRING RETURN INTEGER
A1	ASSIGN ASSIGNMENT
B1	ASSIGN ITERABLES
C1	ASSIGN IFONELINE
ASSIGN	= OPERATORS ASSIGN OPERATORS ASSIGN OPERATORS ASSIGN OPERATORS ASSIGN OPERATORS ASSIGN OPERATORS ASSIGN OPERATORS ASSIGN OPERATORS D1 OPERATORS E1
D1	OPERATORS ASSIGN
E1	OPERATORS ASSIGN
Z4	space
SPACE	Z4
TEXT	TEXT TEXT string
Y4	' U4 Y5
STRING	Y4 F1 X4 G1 STRING H1
F1	TEXT Y4
X4	" U4 X5
G1	TEXT X4
H1	OPERATORS STRING
INTEGER	number
VARIABLE	variable
COMPARISON	< > COMPARISON ASSIGN COMPARISON ASSIGN W4 ASSIGN ASSIGN ASSIGN
W4	! U4 W5
U4	(
T4) Y4 T5
BOOLEAN	U4 I1 True False BOOLEAN J1 BOOLEAN K1 Q4 BOOLEAN VALUE L1 VALUE M1 VALUE N1
I1	BOOLEAN T4
S4	and X4 S5
J1	S4 BOOLEAN
R4	or
K1	R4 BOOLEAN
Q4	not
P4	is
L1	P4 VALUE
M1	P4 M2
M2	Q4 VALUE
N1	COMPARISON VALUE

VALUE	U4 O1 VALUE P1 VALUE Q1 number Y4 F1 X4 G1 STRING H1 U4 I1 True False BOOLEAN J1 BOOLEAN K1 Q4 BOOLEAN VALUE L1 VALUE M1 VALUE N1
O1	VALUE T4
P1	COMPARISON VALUE
Q1	OPERATORS VALUE
OPERATORS	OPERATORS OPERATORS + - OPERATORS OPERATORS % * /
O4	print
PRINT	O4 R1
R1	U4 R2
R2	VALUE T4
N4	#
COMMENT	N4 TEXT X4 S1 Y4 T1
S1	X4 S2
S2	X4 S3
S3	TEXT S4 }
S5	X4 X4
T1	Y4 T2
T2	Y4 T3
T3	TEXT T4 {
T5	Y4 Y4
ASSIGNMENT	VALUE U1 METHOD variable U4 O1 VALUE P1 VALUE Q1 X3 S11 TYPECASTING T11 X3 U11 number Y4 F1 X4 G1 STRING H1 U4 I1 True False BOOLEAN J1 BOOLEAN K1 Q4 BOOLEAN VALUE L1 VALUE M1 VALUE N1
U1	OPERATORS VALUE
M4	range
L4	in
K4	for
J4	:
FOR	K4 W1 K4 X1 K4 Y1 K4 Z1 K4 A11 K4 B11 K4 C11 K4 D11 K4 E11
W1	VARIABLE W2
W2	L4 W3
W3	M4 W4 [
W5	INTEGER W6
W6	T4 J4
I4	,
X1	VARIABLE X2
X2	L4 X3
X3	M4 X4 input
X5	INTEGER X6
X6	I4 X7

X7	INTEGER X8
X8	T4 J4
Y1	VARIABLE Y2
Y2	L4 Y3
Y3	M4 Y4 elif
Y5	INTEGER Y6
Y6	I4 Y7
Y7	INTEGER Y8
Y8	I4 Y9
Y9	INTEGER Y10
Y10	T4 J4
Z1	VARIABLE Z2
Z2	L4 Z3
Z3	VARIABLE J4 else
A11	VARIABLE A12
A12	L4 A13
A13	M4 A14
A14	U4 A15
A15	INTEGER A16
A16	T4 A17
A17	J4 START
B11	VARIABLE B12
B12	L4 B13
B13	M4 B14
B14	U4 B15
B15	INTEGER B16
B16	I4 B17
B17	INTEGER B18
B18	T4 B19
B19	J4 START
C11	VARIABLE C12
C12	L4 C13
C13	M4 C14
C14	U4 C15
C15	INTEGER C16
C16	I4 C17
C17	INTEGER C18
C18	I4 C19
C19	INTEGER C110
C110	T4 C111
C111	J4 START
D11	VARIABLE D12
D12	L4 D13
D13	VARIABLE D14
D14	J4 START
E11	VARIABLE E12
E12	L4 E13
E13	STRING E14

E14	J4 START
BREAK	break
PASS	pass
H4	continue
CONTINUE	H4
G4	while
WHILE	G4 F11 G4 G11
F11	BOOLEAN J4
G11	BOOLEAN G12
G12	J4 START
F4	import
IMPORT	F4 VARIABLE F4 H11
E4	as
H11	VARIABLE H12
H12	E4 VARIABLE
D4	from
FROM	D4 I11
I11	VARIABLE IMPORT
C4	class
CLASS	C4 J11
J11	VARIABLE J4
LOOPVARIABLE	LOOPVARIABLE K11 variable
K11	I4 LOOPVARIABLE
B4	def
DEF	B4 L11 B4 M11
L11	VARIABLE L12
L12	U4 L13
L13	LOOPVARIABLE L14
L14	T4 J4
M11	VARIABLE M12
M12	U4 M13
M13	T4 J4
A4	if
IFONELINE	VALUE N11 VALUE O11 VALUE P11
N11	A4 BOOLEAN
O11	A4 O12
O12	BOOLEAN O13
O13	Z3 VALUE
P11	A4 P12
P12	BOOLEAN P13
P13	Z3 IFONELINE
IF	A4 Q11
Q11	BOOLEAN J4
ELIF	Y3 R11
R11	BOOLEAN J4
ELSE	Z3 J4
INPUT	X3 S11 TYPECASTING T11 X3 U11
S11	U4 S12

G21	DICTELEMENT S3
S12	STRING T4
T11	U4 T12
T12	INPUT T4
U11	U4 T4
RETURN	return RETURN BOOLEAN RETURN VARIABLE RETURN STRING RETURN INTEGER
TYPECASTING	float int complex
ITERABLES	LIST TUPLE SET DICTIONARY
S0	START START VARIABLE A1 VARIABLE B1 VARIABLE C1 A4 Q11 Y3 R11 Z3 J4 O4 R1 K4 W1 K4 X1 K4 Y1 K4 Z1 K4 A11 K4 B11 K4 C11 K4 D11 K4 E11 G4 F11 G4 G11 B4 L11 B4 M11 C4 J11 F4 VARIABLE F4 H11 D4 I11 N4 TEXT X4 S1 Y4 T1 return RETURN BOOLEAN RETURN VARIABLE RETURN STRING RETURN INTEGER
ELEMENT	ELEMENT W11 STRING ALLNUMBERS
W11	I4 ELEMENT
U3]
LIST	W3 X11 W3 Y11
X11	ELEMENT U3
Y11	LIST U3
TUPLE	U4 Z11 U4 A21
Z11	ELEMENT T4
A21	TUPLE T4
SET	T3 B21 T3 C21
B21	ELEMENT S3
C21	SET S3
DICTELEMENT	DICTELEMENT D21 X4 E21 X4 F21
D21	I4 DICTELEMENT
E21	VARIABLE E22
E22	X4 E23
E23	J4 STRING
F21	VARIABLE F22
F22	X4 F23
F23	J4 DICTIONARY
DICTIONARY	T3 G21

Start Symbol (S): S

Bab III

Implementasi dan Pengujian

i. Spesifikasi Struktur Data, Fungsi, dan Prosedur

2.1. File `cyk_parser.py`

File `cyk_parser.py` ini memiliki fungsi yang menerima sebuah grammar dalam bentuk Chomsky Normal Form (CNF) dan menjalankan Algoritma CYK untuk mengetes apakah sebuah input merupakan bagian dari suatu language CFG yang ada.

No	Fungsi/Prosedur	Keterangan
1	<code>def list_duplicate(list,elmt)</code>	Mengembalikan sebuah list yang berisi indeks elemen list yang bernilai sama dengan elemen masukan
2	<code>def concatenate(a,b)</code>	Mengembalikan sebuah set yang berisi hasil concatenate variable a dan b.
3	<code>def check_duplicate(list,elmt)</code>	Mengembalikan nilai true jika elemen masukan sudah berada di dalam list
4	<code>def run(variable,terminal,filename)</code>	Menjalakan algoritma CYK dari masukan variable, terminal, dan file

2.2. File `CFG2CNF.py`

File `CFG2CNF.py` memiliki fungsi untuk mengubah masukan Context Free Grammar ke dalam bentuk Chomsky Normal Form. `CFG2CNF` ini diambil dari referensi <https://github.com/adelmassimo/CFG2CNF>. Output dari program berupa file `cnf.txt` yang dapat langsung digunakan di program utama CYK Parser.

No	Fungsi/Prosedur	Keterangan
1	<code>isUnitary(rule,variables)</code>	Mengembalikan nilai true jika suatu production rule memiliki satu buah symbol dan non-terminal dari daftar variable
2	<code>isSimple(rule)</code>	Mengembalikan nilai true jika suatu production rule memiliki satu buah symbol dan non-terminal dari V
3	<code>START(productions,variables)</code>	Menambahkan start symbol S_0 ke dalam production rule
4	<code>TERM(productions,variables)</code>	Mengubah production rule ke bentuk 2 variable (non-terminal) atau 1 terminal
5	<code>BIN(productions,variables)</code>	Mengeliminasi production rule yang tidak unitary
6	<code>DEL(productins)</code>	Menghapus production rule non-terminal
7	<code>unit_routine(rules,variables)</code>	Mengecek apakah production rule berbentuk unary
8	<code>UNIT(productions,variables)</code>	Menghapus unit production

2.3. File helper.py

File helper.py digunakan dalam file CFG2CNF.py untuk mengonversi Context Free Grammar ke dalam bentuk Chomsky Normal Form. File ini merupakan program pembantu dari CFG2CNF.py. Berikut adalah fungsi/prosedur yang ada pada program tersebut.

No	Fungsi/Prosedur	Keterangan
1	union(list1,list2)	Melakukan operasi union pada list
2	loadModel(modelPath)	Memasukkan aturan variable, terminal, dan production rule
3	cleanProduction(expression)	Melakukan formatting simbol -> dan pada production rule
4	cleanAlphabet(expression)	Menggabungkan alphabet yang dipisahkan dengan koma
5	seekAndDestory(target,productions)	Melakukan penghapusan pada production rule masukan
6	setupDict(productions,variables,terms)	Membentuk production rule ke dalam bentuk dictionary
7	rewrite(target,production)	Menulis ulang production rule
8	dict2Set(dictionary)	Mengonversi dictionary ke bentuk set
9	pprintRules(rules)	Menulis production rule
10	prettyForm(rules)	Melakukan formatting pada output rule

2.4. File py_parser.py

File py_parser.py digunakan untuk melakukan parsing pada masukan yang akan dites. Setiap kata dan simbol-simbol pada masukan akan dipisahkan dengan spasi yang nantinya mengembalikan sebuah list yang elemennya berupa masukan berbentuk token. Token ini nantinya akan digunakan ke dalam Algoritma CYK sebagai input.

No	Fungsi/Prosedur	Keterangan
1	parser(filename)	Melakukan parsing pada masukan dan mengembalikan list dengan elemennya merupakan token

2.5. File parserprogram.py

File readcnf.py ini membaca cnf.txt dan menghasilkan variable dan terminal

No	Fungsi/Prosedur	Keterangan
1	read(filename)	Membaca masukan dari sebuah file text cnf dan mengembalikan nilai berupa variable dan terminal dari grammar

2.6. File main.py

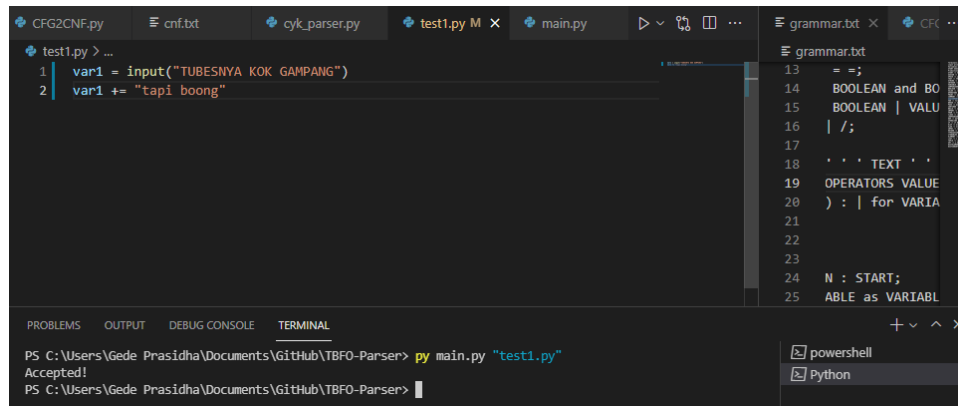
File main.py merupakan program utama yang dijalankan melalui terminal dengan command <python main.py "input.py">, program dijalankan dengan kondisi sudah ada file CNF.txt pada direktori yang sama. Kemudian program utama akan

memanggil `py_parser` untuk membaca input dari “input.py”, setelah itu input akan diterima dan diproses oleh `cyk_parser` beserta file CNF yang dibaca. Program akan mengembalikan pesan “Accepted!” jika input diterima language CFG dan memberikan pesan “Syntax Error!” jika input tidak diterima oleh language CFG.

ii. Screenshot Test Case

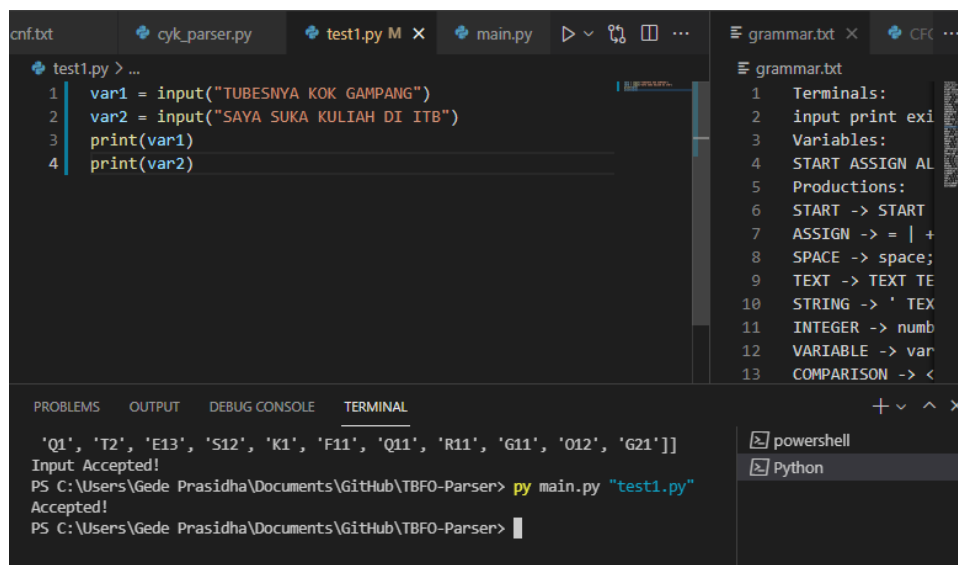
1. Test Case Input Output

Kasus dimana File Accepted:



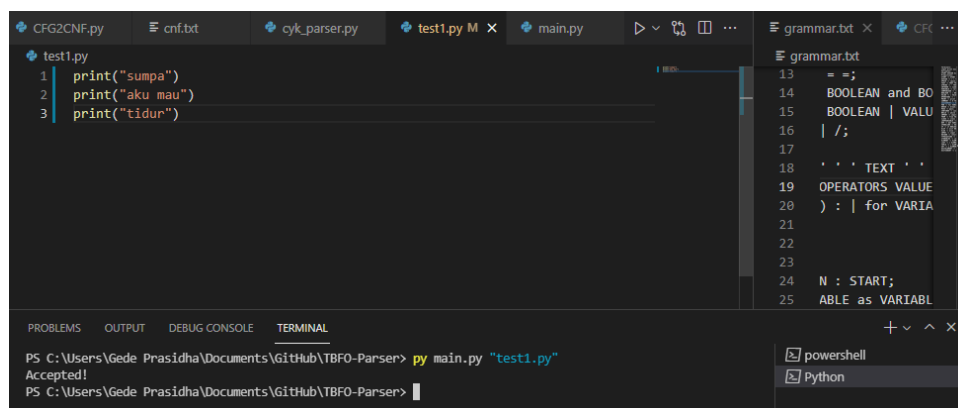
```
CFG2CNF.py  cnf.txt  cyk_parser.py  test1.py M x  main.py  grammar.txt  CFC ...
test1.py > ...
1 var1 = input("TUBESNYA KOK GAMPANG")
2 var1 += "tapi boong"

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser> py main.py "test1.py"
Accepted!
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser>
```



```
cnf.txt  cyk_parser.py  test1.py M x  main.py  grammar.txt  CFC ...
test1.py > ...
1 var1 = input("TUBESNYA KOK GAMPANG")
2 var2 = input("SAYA SUKA KULIAH DI ITB")
3 print(var1)
4 print(var2)

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
'Q1', 'T2', 'E13', 'S12', 'K1', 'F11', 'Q11', 'R11', 'G11', 'O12', 'G21']]
Input Accepted!
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser> py main.py "test1.py"
Accepted!
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser>
```



```
CFG2CNF.py  cnf.txt  cyk_parser.py  test1.py M x  main.py  grammar.txt  CFC ...
test1.py > ...
1 print("sumpa")
2 print("aku mau")
3 print("tidur")

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser> py main.py "test1.py"
Accepted!
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser>
```

Kasus dimana File Syntax Error:

```
test1.py > ...
1 var1 == input("TUBESNYA KOK GAMPANG")
2 var2 = input("SAYA SUKA KULIAH DI ITB")
3 print(var1)
4 print(var2)
```

```
grammar.txt
1 Terminals:
2 input print exi
3 Variables:
4 START ASSIGN AL
5 Productions:
6 START -> START
7 ASSIGN -> = | +
8 SPACE -> space;
9 TEXT -> TEXT TE
10 STRING -> ' TEX
11 INTEGER -> numb
12 VARIABLE -> var
13 COMPARISON -> <
```

```
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser> py main.py "test1.py"
Syntax Error!
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser>
```

```
test1.py
1 printf("sumpa")
2 print("aku mau")
3 print("tidur")
```

```
grammar.txt
13 = =;
14 BOOLEAN and BO
15 BOOLEAN | VALU
16 | /;
17
18 ' ' ' TEXT ' '
19 OPERATORS VALUE
20 ) : | for VARIA
21
22
23
24 N : START;
25 ABLE as VARIABLE
```

```
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser> py main.py "test1.py"
Syntax Error!
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser>
```

```
test1.py > ...
1 1var = float(input("TUBESNYA KOK GAMPANG"))
2 2var += input("SAYA SUKA KULIAH DI ITB")
```

```
grammar.txt
13 = =;
14 BOOLEAN and BO
15 BOOLEAN | VALU
16 | /;
17
18 ' ' ' TEXT ' '
19 OPERATORS VALUE
20 ) : | for VARIA
21
22
23
24 N : START;
25 ABLE as VARIABLE
```

```
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser> py main.py "test1.py"
Syntax Error!
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser>
```

2. Test Case Percabangan (If, Elif, Else)

Kasus If dan If Else (Multiline):

The screenshot shows a VS Code editor with the following files open: `cnf.txt`, `cyk_parser.py`, `test1.py 1, M`, and `main.py`. The `test1.py` file contains the following Python code:

```
1 if sekolah == "ITB":
2     print("saya bahagia")
3 else:
4     print("saya stres")
```

The `grammar.txt` file on the right contains the following grammar rules:

```
1 Terminals:
2 input print exi
3 Variables:
4 START ASSIGN AL
5 Productions:
6 START -> START
7 ASSIGN -> = | +
8 SPACE -> space;
9 TEXT -> TEXT TE
10 STRING -> ' TEX
11 INTEGER -> numb
12 VARIABLE -> var
13 COMPARISON -> <
```

The terminal at the bottom shows the command `py main.py "test1.py"` being executed, resulting in "Accepted!".

The screenshot shows a VS Code editor with the following files open: `CFG2CNF.py`, `cnf.txt`, `cyk_parser.py`, `test1.py 2, M`, and `main.py`. The `test1.py` file contains the following Python code:

```
1 if (A > 3) or (B <= 2):
2     print("ga ngerti lagi lah")
```

The `grammar.txt` file on the right contains the following grammar rules:

```
13 = =;
14 BOOLEAN and BO
15 BOOLEAN | VALU
16 | /;
17 . . . TEXT . .
18 OPERATORS VALUE
19 ) : | for VARIA
20
21
22
23
24 N : START;
25 ABLE as VARIABLE
```

The terminal at the bottom shows the command `py main.py "test1.py"` being executed, resulting in "Accepted!".

The screenshot shows a VS Code editor with the following files open: `CFG2CNF.py`, `cnf.txt`, `cyk_parser.py`, `test1.py M`, and `main.py`. The `test1.py` file contains the following Python code:

```
1 if (9 + 10 == 21):
2     print("anda pintar")
3 else:
4     print("tanya brainly ya")
```

The `grammar.txt` file on the right contains the following grammar rules:

```
13 = =;
14 BOOLEAN and BO
15 BOOLEAN | VALU
16 | /;
17 . . . TEXT . .
18 OPERATORS VALUE
19 ) : | for VARIA
20
21
22
23
24 N : START;
25 ABLE as VARIABLE
```

The terminal at the bottom shows the command `py main.py "test1.py"` being executed, resulting in "Accepted!".

Kasus If Elif Else :

The screenshot shows a VS Code editor with a file named `test1.py` containing the following Python code:

```
1 if sekolah == "ITB":
2     print("saya bahagia")
3 elif sekolah == "ITB Cirebon":
4     print("saya jauh dari bandung")
5 else:
6     print("saya masih sma")
```

The right sidebar shows a file named `grammar.txt` with a list of Python keywords and their corresponding tokens, such as `IMPORT -> impor`, `FROM -> from VA`, `CLASS -> class`, `LOOPVARIABLE ->`, `DEF -> def VARI`, `IFONELINE -> VA`, `IF -> if BOOLEA`, `ELIF -> elif BO`, `ELSE -> else ;`, `INPUT -> input`, `RETURN -> retur`, `TYPECASTING ->`, and `ITERABLES -> LI`.

The bottom panel shows the `TERMINAL` tab with the following output:

```
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser> py main.py "test1.py"
Accepted!
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser>
```

The screenshot shows a VS Code editor with a file named `test1.py` containing the following Python code:

```
1 if (9 + 10 == 21):
2     print("anda pintar")
3 elif (1 + 1 < 3):
4     print("anda sangat pintar")
5 else:
6     print("belajar lagi ya")
```

The right sidebar shows a file named `grammar.txt` with a list of Python keywords and their corresponding tokens, such as `= =;`, `BOOLEAN and BO`, `BOOLEAN | VALU`, `| /;`, `' ' ' TEXT ' '`, `OPERATORS VALUE`, `) : | for VARIA`, `N : START;`, and `ABLE as VARIABL`.

The bottom panel shows the `TERMINAL` tab with the following output:

```
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser> py main.py "test1.py"
Accepted!
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser>
```

The screenshot shows a VS Code editor with a file named `test1.py` containing the following Python code:

```
1 if (9 + 10 == 21):
2     print("anda pintar")
3 elif (1 + 1 < 3):
4     print("anda sangat pintar")
5 elif(qwerty == "qwerty"):
6     print("anda luar biasa pintar")
7 else:
8     print("belajar lagi ya")
```

The right sidebar shows a file named `grammar.txt` with a list of Python keywords and their corresponding tokens, such as `= =;`, `BOOLEAN and BO`, `BOOLEAN | VALU`, `| /;`, `' ' ' TEXT ' '`, `OPERATORS VALUE`, `) : | for VARIA`, `N : START;`, and `ABLE as VARIABL`.

The bottom panel shows the `TERMINAL` tab with the following output:

```
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser> py main.py "test1.py"
Accepted!
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser>
```

Kasus If Else If (One-line) :

```
test1.py > ...
1 | kuliah = "itb" if saya == "gede" else "its" if saya == "kecil" else "telkom"
```

```
grammar.txt
25 IMPORT -> impor
26 FROM -> from VA
27 CLASS -> class
28 LOOPVARIABLE ->
29 DEF -> def VARI
30 IFONELINE -> VA
31 IF -> if BOOLEA
32 ELIF -> elif BO
33 ELSE -> else ;;
34 INPUT -> input
35 RETURN -> retur
36 TYPECASTING ->
37 ITERABLES -> LI
```

```
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser> py main.py "test1.py"
Accepted!
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser>
```

```
test1.py > ...
1 | indeks = "A" if nilai > 75 else "AB" if nilai > 70 and nilai <= 75 else "B"
```

```
grammar.txt
13 = =;
14 BOOLEAN and BO
15 BOOLEAN | VALU
16 | /;
17
18 ' ' ' TEXT ' '
19 OPERATORS VALUE
20 ) : | for VARIA
21
22
23
24 N : START;
25 ABLE as VARIABL
```

```
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser> py main.py "test1.py"
Accepted!
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser>
```

```
test1.py > ...
1 | kosten = "tubis" if hari tua == True else "dago atas"
```

```
grammar.txt
13 = =;
14 BOOLEAN and BO
15 BOOLEAN | VALU
16 | /;
17
18 ' ' ' TEXT ' '
19 OPERATORS VALUE
20 ) : | for VARIA
21
22
23
24 N : START;
25 ABLE as VARIABL
```

```
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser> py main.py "test1.py"
Accepted!
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser>
```

Kasus Syntax Error

```
test1.py > ...
1 | kuliah = "itb" if saya = "gede" elif "its" if saya == "kecil" else "telkom"
```

```
grammar.txt
25 IMPORT -> impor
26 FROM -> from VA
27 CLASS -> class
28 LOOPVARIABLE ->
29 DEF -> def VARI
30 IFONELINE -> VA
31 IF -> if BOOLEA
32 ELIF -> elif BO
33 ELSE -> else ;;
34 INPUT -> input
35 RETURN -> retur
36 TYPECASTING ->
37 ITERABLES -> LI
```

```
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser> py main.py "test1.py"
Syntax Error!
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser>
```

```
test1.py
1 if sekolah == "ITB":
2     print("saya bahagia")
3 else sekolah == "ITB Cirebon":
4     print("saya jauh dari bandung")
5 else:
6     print("saya masih sma")
```

```
grammar.txt
25 IMPORT -> impor
26 FROM -> from VA
27 CLASS -> class
28 LOOPVARIABLE ->
29 DEF -> def VARI
30 IFONELINE -> VA
31 IF -> if BOOLEA
32 ELIF -> elif BO
33 ELSE -> else ;;
34 INPUT -> input
35 RETURN -> retur
36 TYPECASTING ->
37 ITERABLES -> LI
```

PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser> py main.py "test1.py"
Syntax Error!
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser>

```
test1.py
1 if sekolah == "ITB":
2     print("saya bahagia")
3 elif:
4     print("saya stres")
```

```
grammar.txt
25 IMPORT -> impor
26 FROM -> from VA
27 CLASS -> class
28 LOOPVARIABLE ->
29 DEF -> def VARI
30 IFONELINE -> VA
31 IF -> if BOOLEA
32 ELIF -> elif BO
33 ELSE -> else ;;
34 INPUT -> input
35 RETURN -> retur
36 TYPECASTING ->
37 ITERABLES -> LI
```

PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser> py main.py "test1.py"
Syntax Error!
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser>

3. Test Case Looping

Kasus While

```
test1.py
1 while 1 > 3:
2     print("indeks TBFO kelompok aku A")
```

```
grammar.txt
13 == =;
14 BOOLEAN and BO
15 BOOLEAN | VALU
16 | /;
17
18 ' ' ' TEXT ' '
19 OPERATORS VALUE
20 ) : | for VARIA
21
22
23
24 N : START;
25 ABLE as VARIABL
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser> py main.py "test1.py"
Accepted!
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser>

The screenshot shows the VS Code editor with the following files open: CFG2CNF.py, cnf.txt, cyk_parser.py, test1.py, main.py, grammar.txt, and CF. The test1.py file contains the following code:

```
1 while ((True) and not(False) ) :
2     print("indeks TBFO kelompok aku A")
```

The grammar.txt file contains the following code:

```
13 = =;
14 BOOLEAN and BO
15 BOOLEAN | VALU
16 | /;
17
18 ' ' ' TEXT ' '
19 OPERATORS VALUE
20 ) : | for VARIA
21
22
23
24 N : START;
25 ABLE as VARIABLE
```

The terminal output shows the command `py main.py "test1.py"` being executed, resulting in the output `Accepted!`.

The screenshot shows the VS Code editor with the following files open: CFG2CNF.py, cnf.txt, cyk_parser.py, test1.py, main.py, grammar.txt, and CF. The test1.py file contains the following code:

```
1 while (x != "bruh") :
2     print("indeks TBFO kelompok aku A")
```

The grammar.txt file contains the following code:

```
13 = =;
14 BOOLEAN and BO
15 BOOLEAN | VALU
16 | /;
17
18 ' ' ' TEXT ' '
19 OPERATORS VALUE
20 ) : | for VARIA
21
22
23
24 N : START;
25 ABLE as VARIABLE
```

The terminal output shows the command `py main.py "test1.py"` being executed, resulting in the output `Accepted!`.

Kasus For

The screenshot shows the VS Code editor with the following files open: CFG2CNF.py, cnf.txt, cyk_parser.py, test1.py, main.py, grammar.txt, and CF. The test1.py file contains the following code:

```
1 for i in array:
2     print(i)
```

The grammar.txt file contains the following code:

```
13 = =;
14 BOOLEAN and BO
15 BOOLEAN | VALU
16 | /;
17
18 ' ' ' TEXT ' '
19 OPERATORS VALUE
20 ) : | for VARIA
21
22
23
24 N : START;
25 ABLE as VARIABLE
```

The terminal output shows the command `py main.py "test1.py"` being executed, resulting in the output `Accepted!`.

The screenshot shows the VS Code editor with the following files open: CFG2CNF.py, cnf.txt, cyk_parser.py, test1.py, main.py, grammar.txt, and CF. The test1.py file contains the following code:

```
1 for i in "string":
2     print(i)
```

The grammar.txt file contains the following code:

```
13 = =;
14 BOOLEAN and BO
15 BOOLEAN | VALU
16 | /;
17
18 ' ' ' TEXT ' '
19 OPERATORS VALUE
20 ) : | for VARIA
21
22
23
24 N : START;
25 ABLE as VARIABLE
```

The terminal output shows the command `py main.py "test1.py"` being executed, resulting in the output `Accepted!`.


```
test1.py > ...
1 | for i in range(2,3,1):
2 |     print(i)
```

```
grammar.txt
13 | = =;
14 | BOOLEAN and BO
15 | BOOLEAN | VALU
16 | /;
17 |
18 | ' ' ' TEXT ' '
19 | OPERATORS VALUE
20 | ) : | for VARIA
21 |
22 |
23 |
24 | N : START;
25 | ABLE as VARIABL
```

```
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser> py main.py "test1.py"
Accepted!
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser>
```

Kasus Syntax Error

```
test1.py > ...
1 | for i is in range(2,3,1):
2 |     print(i)
```

```
grammar.txt
13 | = =;
14 | BOOLEAN and BO
15 | BOOLEAN | VALU
16 | /;
17 |
18 | ' ' ' TEXT ' '
19 | OPERATORS VALUE
20 | ) : | for VARIA
21 |
22 |
23 |
24 | N : START;
25 | ABLE as VARIABL
```

```
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser> py main.py "test1.py"
Syntax Error!
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser>
```

```
test1.py
1 | for in in x == True:
2 |     print(i)
```

```
grammar.txt
13 | = =;
14 | BOOLEAN and BO
15 | BOOLEAN | VALU
16 | /;
17 |
18 | ' ' ' TEXT ' '
19 | OPERATORS VALUE
20 | ) : | for VARIA
21 |
22 |
23 |
24 | N : START;
25 | ABLE as VARIABL
```

```
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser> py main.py "test1.py"
Syntax Error!
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser>
```

```
test1.py > ...
1 | for a in x >= True:
2 |     print(i)
```

```
grammar.txt
13 | = =;
14 | BOOLEAN and BO
15 | BOOLEAN | VALU
16 | /;
17 |
18 | ' ' ' TEXT ' '
19 | OPERATORS VALUE
20 | ) : | for VARIA
21 |
22 |
23 |
24 | N : START;
25 | ABLE as VARIABL
```

```
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser> py main.py "test1.py"
Syntax Error!
PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser>
```

4. Test Case Class, Import, Def

Kasus Accepted

This screenshot shows the VS Code interface with the file explorer on the left containing `CFG2CNF.py`, `cnf.txt`, `cyk_parser.py`, `test1.py`, and `main.py`. The editor displays `test1.py` with the following code:

```
1 class AFAN:
2     print("tapi jomblo")
```

The right sidebar shows the `grammar.txt` file with a list of grammar rules. The bottom terminal panel shows the command `py main.py "test1.py"` being executed, resulting in the output `Accepted!`.

This screenshot shows the VS Code interface with the file explorer on the left containing `CFG2CNF.py`, `cnf.txt`, `cyk_parser.py`, `test1.py`, and `main.py`. The editor displays `test1.py` with the following code:

```
1 from itb import jodoh as imagination
```

The right sidebar shows the `grammar.txt` file with a list of grammar rules. The bottom terminal panel shows the command `py main.py "test1.py"` being executed, resulting in the output `Accepted!`.

This screenshot shows the VS Code interface with the file explorer on the left containing `CFG2CNF.py`, `cnf.txt`, `cyk_parser.py`, `test1.py`, and `main.py`. The editor displays `test1.py` with the following code:

```
1 def afan():
2     print("ngecarry gede sama ranjabi orkom")
```

The right sidebar shows the `grammar.txt` file with a list of grammar rules. The bottom terminal panel shows the command `py main.py "test1.py"` being executed, resulting in the output `Accepted!`.

Kasus Syntax Error

This screenshot shows the VS Code interface with the file explorer on the left containing `CFG2CNF.py`, `cnf.txt`, `cyk_parser.py`, `test1.py`, and `main.py`. The editor displays `test1.py` with the following code:

```
1 def afan[imba]:
2     print("ngecarry gede sama ranjabi orkom")
```

The right sidebar shows the `grammar.txt` file with a list of grammar rules. The bottom terminal panel shows the command `py main.py "test1.py"` being executed, resulting in the output `Syntax Error!`.

CFG2CNF.py cnf.txt cyk_parser.py test1.py 4, M x main.py grammar.txt CF< ...

test1.py

1 from itb : import jodoh : as imagination

grammar.txt

20 FOR -> for VARI

21 BREAK -> break;

22 PASS -> pass;

23 CONTINUE -> con

24 WHILE -> while

25 IMPORT -> impor

26 FROM -> from VA

27 CLASS -> class

28 LOOPVARIABLE ->

29 DEF -> def VARI

30 IFONELINE -> VA

31 IF -> if BOOLEA

32 ELIF -> elif BO

PROBLEMS 4 OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser> py main.py "test1.py"

Syntax Error!

PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser>

powershell Python

CFG2CNF.py cnf.txt cyk_parser.py test1.py M x main.py grammar.txt CF< ...

test1.py > AFAN

1 class AFAN():

2 | print("tapi jomblo")

grammar.txt

20 FOR -> for VARI

21 BREAK -> break;

22 PASS -> pass;

23 CONTINUE -> con

24 WHILE -> while

25 IMPORT -> impor

26 FROM -> from VA

27 CLASS -> class

28 LOOPVARIABLE ->

29 DEF -> def VARI

30 IFONELINE -> VA

31 IF -> if BOOLEA

32 ELIF -> elif BO

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser> py main.py "test1.py"

Syntax Error!

PS C:\Users\Gede Prasadha\Documents\GitHub\TBFO-Parser>

powershell Python

Bab IV

Penutup

4.1 Kesimpulan

Program parser yang dibuat dalam rangka menyelesaikan Tugas Besar IF2124 Teori Bahasa Formal dan Otomata dapat berjalan dengan baik pada mayoritas uji coba yang diberikannya. Hanya saja, terdapat sedikit bug dalam memproses komen atau multiline string yang mengandung kata yang juga termasuk terminal symbol untuk CFG. Hal ini mungkin disebabkan oleh bagaimana proses pengenalan semua simbol dan teks oleh pemroses file dan juga grammar CFG yang dibuat.

4.2 Saran

Untuk pembuatan program parser kedepannya dapat dilakukan beberapa hal agar program lebih akurat dan efektif.

1. File diproses terlebih dahulu per token menggunakan sebuah lexer agar mengurangi kasus-kasus salah identifikasi struktur
2. FA dapat lebih diimplementasikan secara hard-code daripada menggunakan regex untuk mengenali variabel yang sesuai
3. Perlu dilakukan percobaan dan debugging yang sangat teliti untuk memastikan bahwa grammar CFG yang dibuat sudah mencakupi segala testcase yang seharusnya diterima dan ditolak oleh Python

4.3 Refleksi

Dalam mengerjakan tugas besar ini, tidak sedikit kendala yang kami hadapi terutama dalam memastikan bahwa pemroses file teks yang dihasilkan, grammar CFG, dan algoritma CYK yang dapat berfungsi dengan baik untuk menentukan kebenaran dari sintaks teks program tersebut. Meskipun kami terjun dengan teori dasar yang mencukupi untuk mengerjakan tugas besar ini, namun banyak waktu dari kami terkuras dalam mengimplementasikan teori yang sudah diajari di kelas tersebut menjadi bentuk kode yang benar dan efisien.

Bab V

Lampiran

Repository Github:

<https://github.com/blueguy42/TBFO-Parser>

Pembagian Tugas:

Nama	Pembagian Tugas
13520002 MUHAMMAD FIKRI RANJABI	Laporan, CYK Algorithm
13520004 GEDE PRASIDHA BHAWARNAWA	Laporan, Grammar
13520023 AHMAD ALFANI HANDOYO	Laporan, Parser