Tugas Besar IF2123 Formal Language Theory and Automata

Python Code Compiler

Semester I Tahun 2021/2022

Disusun oleh:

Vionie Novencia Thanggestyo 13520006

Jaya Mangalo Soegeng Rahardjo 13520015

Maharani Ayu Putri Irawan 13520019

Institut Teknologi Bandung

Sekolah Teknik Elektro dan Informatika

DAFTAR ISI

Bab I

Dasar Teori

1.2 Context-Free Grammar

Context-Free Grammar (CFG) adalah sebuah aturan untuk membangun Context-Free Language, sebuah language yang memilki fokus dalam produksi menggunakan rekursifitas. Seperti semua *regular language*, CFG dapat digunakan untuk mengecek kebenaran string. Tetapi CFG juga dapat menyelesaikan berbagai masalah yang tidak dapat diselesaikan *regular language*.

CFG pada dasarnya adalah sebuah grammar yang terdiri dari kumpulan aturan seperti berikut:

$$A \rightarrow \alpha$$

Dimana A adalah sebuah simbol non-terminal dan α adalah kumpulan dari satu atau lebih simbol terminal atau non-terminal maupun ε (empty).

CFG secara formal dapat ditulis dengan notasi berikut:

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

G = Context Free Language

V = Himpunan Variabel/Symbol Non-Terminal

T = Himpunan Symbol Terminal

P = Himpunan Aturan Produksi

S = Start State

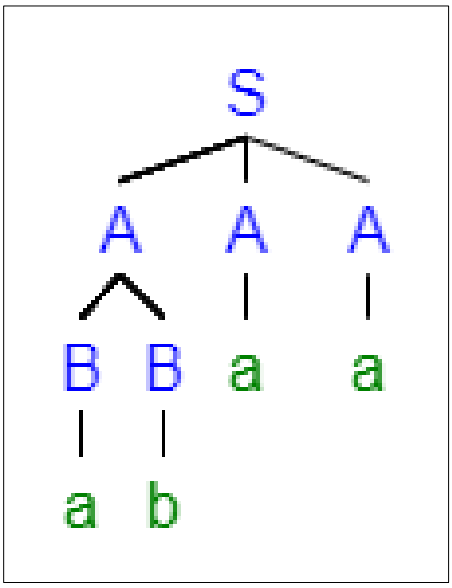
Derivation Trees/Parse Trees

Derivation Trees atau *Parse Trees* adalah sebuah visualisasi dari proses produksi sebuah string dengan sebuah aturan produksi tertentu.

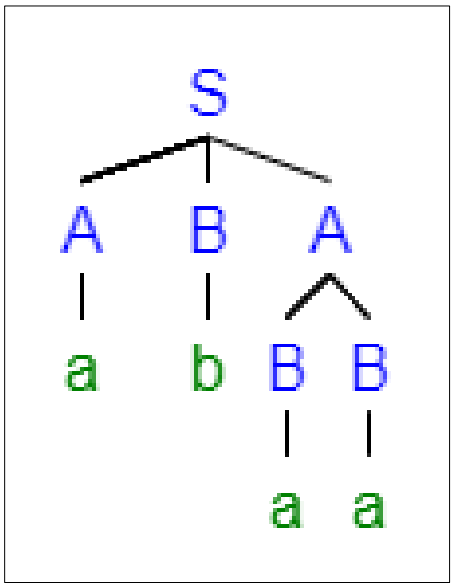
Jika misalnya terdapat sebuah grammar dengan aturan berikut:

- S → AAA | ABA
- A → a |BB
- B → a|b

Jika ingin mendapatkan string “abaa” dengan aturan produksi tersebut, terdapat dua metode yaitu *Left Derivation Trees* atau *Right Derivation Trees*. Left Derivation Trees adalah tree yang dikerjakan dari paling kiri ke kanan. Right Derivation Trees adalah kebalikan dari Left Derivation Trees, pengerjaan dari kanan ke kiri.



Contoh Left Derivation Tree

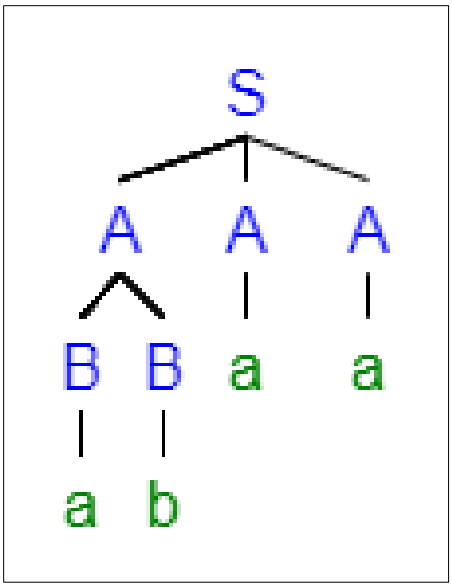


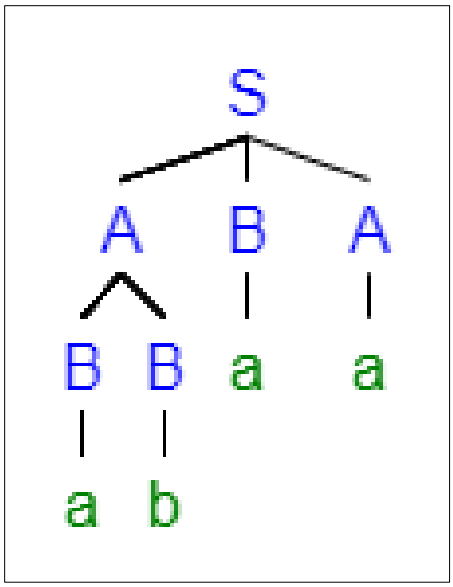
Contoh Right Derivation Tree

Ambiguity

Secara informal, Ambiguity berarti bahwa sebuah aturan produksi memiliki lebih dari satu cara untuk mendapatkan suatu hasil/string yang sama. Untuk membuktikan bahwa sebuah grammar adalah ambigu, perlu dibuktikan kalau terdapat dua atau lebih left (atau right) derivation trees untuk sebuah string yang sama.

Menggunakan contoh grammar sebelumnya, terdapat beberapa cara untuk mendapatkan string “abaa” menggunakan left derivation tree. Salah duanya adalah:





Karena terdapat dua atau lebih cara untuk mendapatkan string “abaa”, terbukti bahwa grammar yang telah didefinisikan sebelumnya adalah “ambigu”.

1.2 Chomsky Normal Form

Chomsky Normal Form (CNF) adalah sebuah bentuk lain dari CFG, dimana semua CNF adalah sebuah CFG dan semua CFG dapat dikonversi menjadi CNF. CNF tersebut adalah sebuah CFG dengan beberapa aturan tambahan:

- Tidak boleh ada produksi epsilon
- Tidak boleh ada produksi unit
- Tidak boleh ada produksi useless
- Tidak boleh ada produksi yang hasilnya gabungan dari simbol termnal dan non-terminal

Karena aturan tersebut, Chomsky Normal Form hanya memiliki 2 tipe produksi, yaitu

$$\mathbf{A \rightarrow BC \text{ atau } A \rightarrow a}$$

A, B, C = non-terminal symbols

a = terminal symbols

Transformasi Context-Free Grammar to Chomsky Normal Form

Untuk mengubah sebuah Context-Free Grammar menjadi Chomsky Normal Form, terdapat beberapa tahap:

1. Jika start simbol S muncul di sebelah kanan, buat start simbol baru dengan produksi S' → S.
2. Hilangkan produksi Null (ε).
3. Hilangkan produksi unit.
4. Jika terdapat produksi A→ B_1.....B_n, dengan n > 2, rubahlah menjadi A → B_1C_1 & C_1 → B_2C_2 , ulangi sampai setiap tiap produksi hanya menghasilkan 2 non-terminal.
5. Jika ada produksi yang hasilnya gabungan terminal dan non terminal, ex: A→ Bc. Rubahlah menjadi A→ BC, C → c.

Misalkan ada CFG dengan produksi:

- **S** → ASA | aB
- **A** → B | S
- **B** → b | ε

Transformasi tahap 1 (Perubahan simbol Start):

- **S'** → S
- **S** → ASA | aB
- **A** → B | S
- **B** → b | ε

Transformasi tahap 2 (Penghilangan produksi Null):

Remove B → ε :

- **S'** → S
- **S** → ASA | aB | a
- **A** → B | S | ε
- **B** → b

Remove A → ε :

- **S'** → S
- **S** → ASA | AS | SA | S | aB | a
- **A** → B | S
- **B** → b

Transformasi tahap 3 (Penghilangan produksi Unit):

Remove S → S :

- **S'** → S
- **S** → ASA | AS | SA | aB | a
- **A** → B | S
- **B** → b

Remove S' → S :

- **S'** → ASA | AS | SA | aB | a
- **S** → ASA | AS | SA | aB | a
- **A** → B | S
- **B** → b

Remove A → S

- **S'** → ASA | AS | SA | aB | a
- **S** → ASA | AS | SA | aB | a
- **A** → B | ASA | AS | SA | aB | a
- **B** → b

Remove A → B

- **S'** → ASA | AS | SA | aB | a
- **S** → ASA | AS | SA | aB | a
- **A** → b | ASA | AS | SA | aB | a
- **B** → b

Transformasi tahap 4 (Penghilangan produksi lebih dari 3 non-terminal):

Change SA menjadi X

- **S'** → AX | AS | SA | aB | a
- **S** → AX | AS | SA | aB | a
- **A** → b | AX | AS | SA | aB | a
- **X** → SA
- **B** → b

Transformasi tahap 5 (Pemisahan terminal dengan non-terminal):

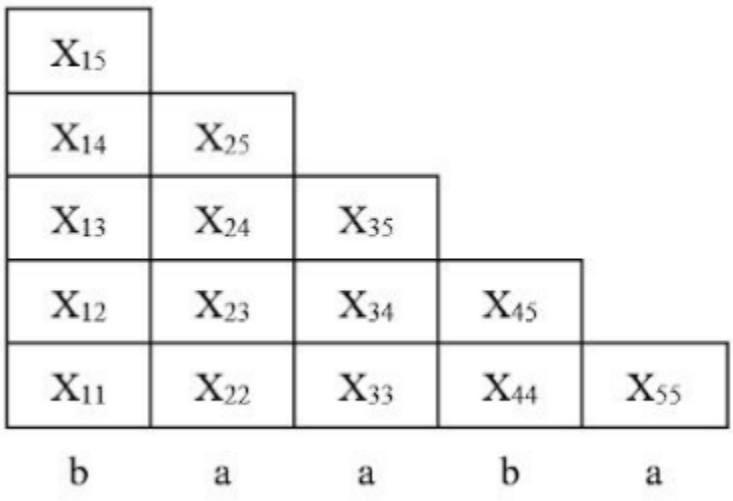
Change aB menjadi PB, P → a

- **S’** → AX | AS | SA | PB | a
- **S** → AX | AS | SA | PB | a
- **A** → b | AX | AS | SA | PB | a
- **X** → SA
- **B** → b
- P → a

1.3 Cocke-Younger Kasami

Cocke Younger Kasami (CYK) adalah sebuah algoritma yang digunakan terhadap *Context-Free Grammar* yang sudah ditransformasikan menjadi *Chomsky Normal Form*. Algoritma CYK digunakan untuk mengecek kevaliditas dari sebuah string terhadap sebuah *grammar* tertentu.

Algortima CYK dieksekusikan dengan membangun tabel dimana isi tabel tersebut adalah akar(state untuk mulai) untuk membangun string.



X11 menyatakan akar yang dapat membangun string ‘b’,

X22 menyatakan akar yang dapat membangun string ‘a’,

X12 menyatakan akar yang dapat membangun string ‘ba’,

X24 menyatakan akar yang dapat membangun string ‘aab’, dan seterusnya.

Agar sebuah string dapat dinyatakan valid, perlu memiliki akar yang mengandung Start state. Jika tidak memiliki akar, maka string tersebut tidak bisa dibangung dengan CFG tersebut.

Terdapat contoh CFG/CNF dan tabel CYK yang sudah diisi.

CFG:

S → AB | BC

S → BA | a

A → CC | b

B → AB | a

String yang ingin dicek kevaliditasnya: ‘baaba’

Tabel:

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

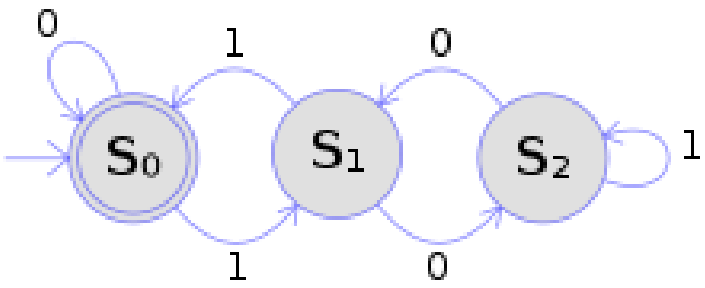
Dalam tabel tersebut, B adalah akar yang dapat membangun string ‘b’, A dan C adalah akar yang dapat membangun string ‘a’. Lalu A dan S adalah akar yang dapat membangun string ‘ba’. Pada akar terakhirnya, terdapat S, A ,dan C. Yang diperhatikan adalah S karena S merupakan start state dari CFG. Karena akar puncak mengandung start state S, maka CFG tersebut **valid** dalam membangun string ‘baaba’.

1.4 Finite Automata

Finite Automata (FA) adalah sebuah mesin automata dari *Regular Language.* FA dan regular language adalah cara paling dasar dan sederhana dalam automata. FA bekerja dengan prinsip dengan menggunakan state yang sudah di-*predefine* terlebih dahulunya dan berpindah-pindah sesuai dengan input yang aturan pindahannya juga sudah di-*define*.

Finite automata bekerja dengan mengecek apabila sebuah string dapat diterima oleh finite automata tersebut dengan membagi tiap isi string menjadi karakter dimana setiap karakter tersebut adalah input yang akan mengganti current state. Jika setelah meng-input karakter terakhir dan hasilnya jatuh di *accepting state*, maka string tersebut diaccept, JIka tidak jatuh di *accepting state*, maka string tersebut akan ditolak.

Contoh FA:

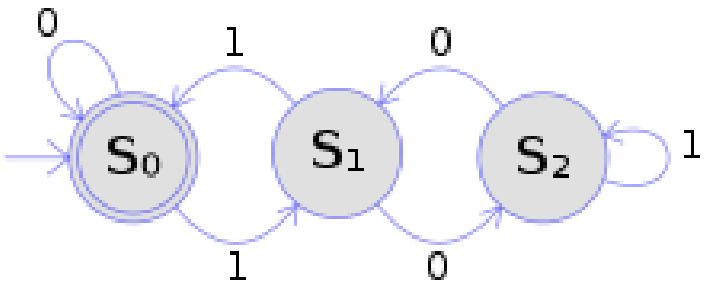


Jika ingin mengecek apakah string “01” dapat diterima oleh FA tersebut, dapat dilakukan pertama dengan mengambil ‘0’ dan memasukkanya sebagai input ke FA. Start state So jika menerima input 0 akan pindah ke S1, lalu jika ia menerima input 1 ia akan pindah ke So. Karena string habis dan So adalah accepting state maka string ‘01’ akan diterima oleh FA tersebut.

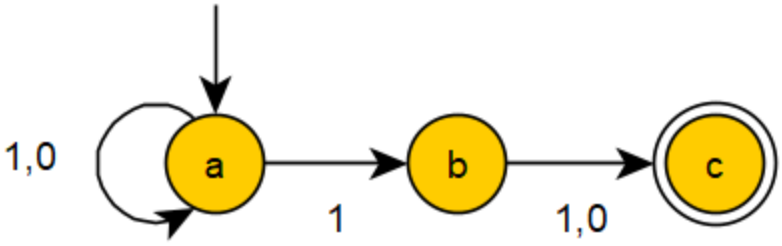
Determinism and Non-Determinism

Sebuah Finite Automata disebut Deterministic Finite Automata(DFA) jika setiap state memiliki tepat 1 transisi untuk setiap input. Jika FA tersebut tidak memenuhi syarat sebelumnya, maka Finite Automata tersebut disebut Non-Deterministic Finite Automata(NFA).

Contoh DFA



Contoh NFA



1.5 Python

Python adalah sebuah “high-level general-purpose programming language”, python dibuat oleh Guido van Rossum pada tahun 1991 dengan fokus ke “keterbacaan” yang tampak jelas dengan cara penulisan koding yang fokus ke struktur indentasi.

Python karena kemampuan multi-purposenya, terutama sebagai object-orientated programming, functional programming, adalah sebuah bahasa pemrograman yang sangat populer pada masa sekarang (2021).

Seperti yang disebut terlebih-dahulu tentang fokus python ke “keterbacaan”, python memiliki design yang bersih, dimana program-program bahasa lain memerlukan end of line(;) dan/atau kurung kurawal({}) untuk sebuah loop/fungsi, python menggantikan kedua tersebut dengan newline dan indentasi. Karena itu, python sangat udah untuk dibaca (dan dibuat).

Seiring dengan keterbacaan, hampir semua fungsi dan operator di python menggunakan bahasa inggris, misalnya jika di C ‘&&’ dapat diartikan sebagai ‘and’ di python, penamaan inggris yang jelas tersebut sangat membantu dalam keterbacaan dan pembuatan program.

Bab II

Implementasi dan Pengujian

2.1 Dekomposisi Program

2.1.1 CFG

Context-Free Grammar yang dibuat untuk mengecek validitas program python adalah:

G = (V, T, P, S)

V (Variables / Non-Terminal Symbols)

ALGORITHM	**STRING**	**ELIFSTATEMENT**	**IMPORTSTATEMENT**
ISIWHILE	**DICTIONARY**	**ELSESTATEMENT**	**WITHSTATEMENT**
ALGOLOOP	**BOOLEAN**	**RAISESTATEMENT**	**COMMENT**
DECLARE	**OPERATOR**	**ERROR**	**WITH**
ISIVARIABEL	**OTHER_OPERATOR**	**FORSTATEMENT**	**IMPORT**
ISIVARNUMBER	**RELATION**	**FUNCTIONCALL**	**FROM**
FLOAT	**VAR3**	**RANGESTATEMENT**	**FOR**
NEGATIVE	**VAR4**	**INLINERANGE**	**RANGE**
CONDITION	**VAR5**	**INLINEFOR**	**CLASS**
PRINTSTATEMENT	**VAR6**	**FROMSTATEMNET**	**RAISE**
INPUTSTATEMENT	**DEFSTATEMENT**	**VAR**	**BK**
STRINGS	**RETURNSTATEMENT**	**BREAK**	**TK**
SAMADENGAN	**CLASSSTATEMENT**	**NONE**	**STRIP**
WHILE	**IFSTATEMENT**	**AND**	**IF**
TITIKDUA	**RETURN**	**OR**	**ELIF**
NUMBER	**IS**	**NOT**	**ELSE**
KOMA	**IN**	**AS**	**PRINT**
TPS	**BPD**	**CONTINUE**	**INPUT**

BKS	**TPD**	**TKS**	**PASS**
DEF	**BPS**	**BKK**	**TKK**

T (Terminal Symbols)

-	*	/	%	=	>	<
!	[]	( )	'	"	#	:
,	&	^	variable	number	string	str
int	float	char	True	False	none	and
or	not	as	break	continue	class	def
if	elif	else	raise	for	in	from
import	is	pass	return	while	with	input
print						

P (Productions)

Productions	Result
ALGORITHM	**ALGORITHM ALGORITHM | ISIWHILE | DECLARE | FORSTATEMENT | DEFSTATEMENT | CLASSSTATEMENT | IFSTATEMENT | IMPORT | FROM | WITHSTATEMENT | FUNCTIONCALL | PRINTSTATEMENT | INPUTSTATEMENT**
ISIWHILE	**WHILE CONDITION TITIKDUA ALGOLOOP | WHILE BK BOOLEAN TK TITIKDUA ALGOLOOP | WHILE BK TK TITIKDUA ALGOLOOP**
ALGOLOOP	**ALGOLOOP ALGOLOOP | BREAK | CONTINUE | DECLARE | IFSTATEMENT | ISIWHILE | FORSTATEMENT | PRINTSTATEMENT | INPUTSTATEMENT**
DECLARE	**VAR SAMADENGAN ISIVARIABEL | VAR OTHER_OPERATOR ISIVARNUMBER | VAR SAMADENGAN BKS TKS | VAR SAMADENGAN BKS INLINE_FOR TKS | VAR SAMADENGAN BKS VAR5 TKS | DICTIONARY**
ISIVARIABEL	**VAR | STRINGS | ISIVARNUMBER | BK ISIVARIABEL TK**
ISIVARNUMBER	**FLOAT | NUMBER | NEGATIVE**
FLOAT	**NUMBER TITIK NUMBER**
NEGATIVE	**STRIP NUMBER | STRIP FLOAT**

CONDITION	**BK CONDITION TK | BOOLEAN | CONDITION AND CONDITION | CONDITION OR CONDITION | NOT CONDITION | CONDITION RELATION CONDITION | ISIVARIABEL RELATION ISIVARIABEL**
PRINTSTATEMENT	**PRINT BK VAR TK | PRINT BK STRINGS TK**
INPUTSTATEMENT	**INPUT BK TK | INPUT BK STRINGS TK | TYPE BK INPUTSTATEMENT TK**
STRINGS	**BPD STRING TPD | BPS STRING TPS**
STRING	**string | string STRING**
DICTIONARY	**VAR SAMADENGAN BKK TKK | VAR SAMADENGAN BKK VAR3 TITIKDUA VAR3 TKK**
BOOLEAN	**True | False**
OPERATOR	**+ | - | * | / | / / | % | * ***
OTHER_OPERATOR	*** = | + = | - = | / = | % = | / / = | * * = | & = | ^ = | > > = | < < =**
RELATION	**> | > = | < | < = | ! = | = =**
VAR3	**VAR | NUMBER | FLOAT | NEGATIVE**
VAR4	**VAR3 | VAR3 OPERATOR VAR3**
VAR5	**VAR | VAR KOMA VAR**
VAR6	**VAR3 | STRINGS**
DEFSTATEMENT	**DEF VAR BK VAR3 TK TITIKDUA ALGORITHM | DEF VAR BK VAR3 TK TITIKDUA ALGORITHM RETURNSTATEMENT**
RETURNSTATEMENT	**RETURN BOOLEAN | RETURN VAR6 | RETURN VAR4 | RETURN**
CLASSSTATEMENT	**CLASS VAR TITIKDUA ALGORITHM**
IFSTATEMENT	**IF BK BOOLEAN TK TITIKDUA ALGORITHM | IF BK VAR RELATION VAR4 TK TITIKDUA ALGORITHM | IFSTATEMENT ELIFSTATEMENT | IFSTATEMENT ELSESTATEMENT | IFSTATEMENT BREAK | IFSTATEMENT CONTINUE | IFSTATEMENT RAISESTATEMENT | IFSTATEMENT PASS**
ELIFSTATEMENT	**ELIF BK BOOLEAN TK TITIKDUA ALGORITHM | ELIF BK VAR RELATION VAR4 TK TITIKDUA ALGORITHM | ELIFSTATEMENT ELIFSTATEMENT | ELIFSTATEMENT ELSESTATEMENT**
ELSESTATEMENT	**ELSE TITIKDUA ALGORITHM**
RAISESTATEMENT	**RAISE ERROR BK STRING TK**
ERROR	**ValueError | ZeroDivisionError | ImportError | NameError | TypeError**

FORSTATEMENT	**FOR VAR IN RANGE TITIKDUA ALGORITHM | FOR VAR IN VAR TITIKDUA ALGORITHM | FOR VAR IN STRING TITIKDUA ALGORITHM**
FUNCTIONCALL	**VAR BK VAR3 TK | VAR BK TK | TYPE BK VAR3 TK | FUNCTIONCALL BK FUNCTIONCALL TK;**
RANGESTATEMENT	**RANGE BK CAR3 TK | RANGE BK NUMBER KOMA NUMBER KOMA NUMBER TK | RANGE BK NUMBER KOMA NUMBER TK;**
INLINERANGE	**BK VAR3 TK**
INLINEFOR	**VAR FOR VAR IN INLINERANGE | VAR FOR VAR IN STRING**
FROMSTATEMNET	**FROM VAR IMPORT**
IMPORTSTATEMENT	**IMPORT VAR | IMPORT VAR AS VAR**
WITHSTATEMENT	**WITH VAR BK VAR TK AS VAR TITIKDUA ALGORITHM**
COMMENT	**BPD BPD BPD STRING TPD TPD TPD | BPS BPS BPS STRING TPS TPS TPS**
WITH	**with**
IMPORT	**import**
FROM	**from**
FOR	**for**
RANGE	**range**
CLASS	**class**
RAISE	**raise**
BK	**(**
TK	**)**
VAR	**variable**
BREAK	**break**
NONE	**None**
AND	**and**
OR	**or**
NOT	**not**
AS	**as**
CONTINUE	**continue**
SAMADENGAN	**=**
WHILE	**while**
TITIKDUA	**:**
NUMBER	**number**
KOMA	**,**
STRIP	**-**
IF	**if**
ELIF	**elif**

ELSE	**else**
PRINT	**print**
INPUT	**input**
PASS	**pass**
DEF	**def**
RETURN	**return**
IS	**is**
IN	**in**
BPD	**"**
TPD	**"**
BPS	**'**
TPS	**'**
BKS	**[**
TKS	**]**
BKK	**{**
TKK	**}**

S (Start State)

Start State → ALGORITHM

2.1.2 CNF

Chomsky Normal Form setelah konversi Context Free Grammar adalah berikut:

G = (V, T, P, S)

V (Variables / Non-Terminal Symbols)

ALGORITHM	**OPERATOR**	**O12**	**F26**
ISIWHILE	**Z01**	**O13**	**F27**
A01	**OTHER_OPERATOR**	**ERROR**	**COMMENT**
A02	**A11**	**FORSTATEMENT**	**G21**
B01	**B11**	**P11**	**G22**
B02	**Z6**	**P12**	**G23**
B03	**Y6**	**P13**	**G24**
B04	**C11**	**P14**	**G25**
C01	**D11**	**Q11**	**H21**
C02	**RELATION**	**Q12**	**H22**
C03	**X6**	**Q13**	**H23**
ALGOLOOP	**VAR4**	**Q14**	**H24**
DECLARE	**E11**	**R11**	**H25**
D01	**VAR5**	**R12**	**WITH**
E01	**F11**	**R13**	**IMPORT**
F01	**DEFSTATEMENT**	**R14**	**FROM**

F02	**G11**	**FUNCTIONCALL**	**FOR**
G01	**G12**	**S11**	**RANGE**
G02	**G13**	**S12**	**CLASS**
G03	**G14**	**T11**	**RAISE**
H01	**G15**	**T12**	**BK**
H02	**H11**	**U11**	**TK**
H03	**H12**	**U12**	**VAR**
ISIVARIABEL	**H13**	**W11**	**BREAK**
I01	**H14**	**W12**	**NONE**
FLOAT	**H15**	**RANGESTATEMENT**	**AND**
J01	**H16**	**X11**	**OR**
NEGATIVE	**RETURNSTATEMENT**	**X12**	**NOT**
CONDITION	**CLASSSTATEMENT**	**Y11**	**AS**
K01	**I11**	**Y12**	**CONTINUE**
L01	**I12**	**Y13**	**SAMADENGAN**
M01	**IFSTATEMENT**	**Y14**	**WHILE**
N01	**J11**	**Y15**	**TITIKDUA**
O01	**J12**	**Y16**	**NUMBER**
PRINTSTATEMENT	**J13**	**Z11**	**KOMA**
P01	**J14**	**Z12**	**STRIP**
P02	**K11**	**Z13**	**IF**
Q01	**K12**	**Z14**	**ELIF**
Q02	**K13**	**INLINERANGE**	**ELSE**
INPUTSTATEMENT	**K14**	**A21**	**PRINT**
R01	**K15**	**INLINEFOR**	**INPUT**
S01	**K16**	**B21**	**PASS**
S02	**ELIFSTATEMENT**	**B22**	**DEF**
T01	**L11**	**B23**	**RETURN**
T02	**L12**	**C21**	**IS**
STRINGS	**L13**	**C22**	**IN**
U01	**L14**	**C23**	**BPD**
W01	**L15**	**FROMSTATEMNET**	**TPD**
STRING	**M11**	**D21**	**BPS**
DICTIONARY	**M12**	**IMPORTSTATEMENT**	**TPS**
X01	**M13**	**E21**	**BKS**
X02	**M14**	**E22**	**TKS**
Y01	**M15**	**WITHSTATEMENT**	**BKK**
Y02	**M16**	**F21**	**TKK**
Y03	**ELSESTATEMENT**	**F22**	**S0**
Y04	**N11**	**F23**	**ISIVARNUMBER**
Y05	**RAISESTATEMENT**	**F24**	**VAR3**
BOOLEAN	**O11**	**F25**	**VAR6**

T (Terminal Symbols)

-	*	/	%	=	>

!	[]	( )	'	"	#
,	&	^	variable	number	string
int	float	char	True	False	none
or	not	as	break	continue	class
if	elif	else	raise	for	in

P (Productions)

Productions	Result
ALGORITHM	**ALGORITHM ALGORITHM | WHILE A01 | WHILE B01 | WHILE C01 | VAR D01 | VAR E01 | VAR F01 | VAR G01 | VAR H01 | FOR P11 | FOR Q11 | FOR R11 | DEF G11 | DEF H11 | CLASS I11 | IF J11 | IF K11 | IFSTATEMENT ELIFSTATEMENT | IFSTATEMENT ELSESTATEMENT | IFSTATEMENT BREAK | IFSTATEMENT CONTINUE | IFSTATEMENT RAISESTATEMENT | IFSTATEMENT PASS | import | from | WITH F21 | VAR S11 | VAR T11 | TYPE U11 | FUNCTIONCALL W11 | PRINT P01 | PRINT Q01 | INPUT R01 | INPUT S01 | TYPE T01 | VAR X01 | VAR Y01**
ISIWHILE	**WHILE A01 | WHILE B01 | WHILE C01**
A01	**CONDITION A02**
A02	**TITIKDUA ALGOLOOP**
B01	**BK B02**
B02	**BOOLEAN B03**
B03	**TK B04**
B04	**TITIKDUA ALGOLOOP**
C01	**BK C02**
C02	**TK C03**
C03	**TITIKDUA ALGOLOOP**
ALGOLOOP	**ALGOLOOP ALGOLOOP | break | continue | VAR D01 | VAR E01 | VAR F01 | VAR G01 | VAR H01 | IF J11 | IF K11 | IFSTATEMENT ELIFSTATEMENT | IFSTATEMENT ELSESTATEMENT | IFSTATEMENT BREAK | IFSTATEMENT CONTINUE | IFSTATEMENT RAISESTATEMENT | IFSTATEMENT PASS | WHILE A01 | WHILE B01 | WHILE C01 | FOR P11 | FOR Q11 | FOR R11 | PRINT P01 | PRINT Q01 | INPUT R01 | INPUT S01 | TYPE T01**

	| VAR X01 | VAR Y01
DECLARE	**VAR D01 | VAR E01 | VAR F01 | VAR G01 | VAR H01 | VAR X01 | VAR Y01**
D01	**SAMADENGAN ISIVARIABEL**
E01	**OTHER_OPERATOR ISIVARNUMBER**
F01	**SAMADENGAN F02**
F02	**BKS TKS**
G01	**SAMADENGAN G02**
G02	**BKS G03**
G03	**INLINE_FOR TKS**
H01	**SAMADENGAN H02**
H02	**BKS H03**
H03	**VAR5 TKS**
ISIVARIABEL	**BK I01 | variable | BPD U01 | BPS W01 | NUMBER J01 | number | STRIP NUMBER | STRIP FLOAT**
I01	**ISIVARIABEL TK**
FLOAT	**NUMBER J01**
J01	**TITIK NUMBER**
NEGATIVE	**STRIP NUMBER | STRIP FLOAT**
CONDITION	**BK K01 | CONDITION L01 | CONDITION M01 | NOT CONDITION | CONDITION N01 | ISIVARIABEL O01 | True | False**
K01	**CONDITION TK**
L01	**AND CONDITION**
M01	**OR CONDITION**
N01	**RELATION CONDITION**
O01	**RELATION ISIVARIABEL**
PRINTSTATEMENT	**PRINT P01 | PRINT Q01**
P01	**BK P02**
P02	**VAR TK**
Q01	**BK Q02**
Q02	**STRINGS TK**
INPUTSTATEMENT	**INPUT R01 | INPUT S01 | TYPE T01**
R01	**BK TK**
S01	**BK S02**
S02	**STRINGS TK**
T01	**BK T02**
T02	**INPUTSTATEMENT TK**
STRINGS	**BPD U01 | BPS W01**
U01	**STRING TPD**
W01	**STRING TPS**
STRING	**string | STRING STRING**

DICTIONARY	**VAR X01 | VAR Y01**
X01	**SAMADENGAN X02**
X02	**BKK TKK**
Y01	**SAMADENGAN Y02**
Y02	**BKK Y03**
Y03	**VAR3 Y04**
Y04	**TITIKDUA Y05**
Y05	**VAR3 TKK**
BOOLEAN	**True | False**
OPERATOR	**+ | - | * | / | Z01 | % | OPERATOR OPERATOR**
Z01	**OPERATOR OPERATOR**
OTHER_OPERATOR	**OPERATOR SAMADENGAN | OPERATOR SAMADENGAN | STRIP SAMADENGAN | OPERATOR SAMADENGAN | OPERATOR SAMADENGAN | OPERATOR A11 | OPERATOR B11 | Z6 SAMADENGAN | Y6 SAMADENGAN | RELATION C11 | RELATION D11**
A11	**OPERATOR SAMADENGAN**
B11	**OPERATOR SAMADENGAN**
Z6	**&**
Y6	**^**
C11	**RELATION SAMADENGAN**
D11	**RELATION SAMADENGAN**
RELATION	**> | RELATION SAMADENGAN | < | RELATION SAMADENGAN | X6 SAMADENGAN | SAMADENGAN SAMADENGAN**
X6	**!**
VAR4	**VAR3 E11 | variable | number | NUMBER J01 | STRIP NUMBER | STRIP FLOAT**
E11	**OPERATOR VAR3**
VAR5	**VAR F11 | variable**
F11	**KOMA VAR**
DEFSTATEMENT	**DEF G11 | DEF H11**
G11	**VAR G12**
G12	**BK G13**
G13	**VAR3 G14**
G14	**TK G15**
G15	**TITIKDUA ALGORITHM**
H11	**VAR H12**
H12	**BK H13**
H13	**VAR3 H14**
H14	**TK H15**

H15	**TITIKDUA H16**
H16	**ALGORITHM RETURNSTATEMENT**
RETURNSTATEMENT	**RETURN BOOLEAN | RETURN VAR6 | RETURN VAR4 | return**
CLASSSTATEMENT	**CLASS I11**
I11	**VAR I12**
I12	**TITIKDUA ALGORITHM**
IFSTATEMENT	**IF J11 | IF K11 | IFSTATEMENT ELIFSTATEMENT | IFSTATEMENT ELSESTATEMENT | IFSTATEMENT BREAK | IFSTATEMENT CONTINUE | IFSTATEMENT RAISESTATEMENT | IFSTATEMENT PASS**
J11	**BK J12**
J12	**BOOLEAN J13**
J13	**TK J14**
J14	**TITIKDUA ALGORITHM**
K11	**BK K12**
K12	**VAR K13**
K13	**RELATION K14**
K14	**VAR4 K15**
K15	**TK K16**
K16	**TITIKDUA ALGORITHM**
ELIFSTATEMENT	**L11 | ELIF M11 | ELIFSTATEMENT ELIFSTATEMENT | ELIFSTATEMENT ELSESTATEMENT**
L11	**ELIF L12**
L12	**BK L13**
L13	**BOOLEAN L14**
L14	**TK L15**
L15	**TITIKDUA ALGORITHM**
M11	**BK M12**
M12	**VAR M13**
M13	**RELATION M14**
M14	**VAR4 M15**
M15	**TK M16**
M16	**TITIKDUA ALGORITHM**
ELSESTATEMENT	**ELSE N11**
N11	**TITIKDUA ALGORITHM**
RAISESTATEMENT	**RAISE O11**
O11	**ERROR O12**
O12	**BK O13**
O13	**STRING TK**

ERROR	**ValueError | ZeroDivisionError | ImportError | NameError | TypeError**
FORSTATEMENT	**FOR P11 | FOR Q11 | FOR R11**
P11	**VAR P12**
P12	**IN P13**
P13	**RANGE P14**
P14	**TITIKDUA ALGORITHM**
Q11	**VAR Q12**
Q12	**IN Q13**
Q13	**VAR Q14**
Q14	**TITIKDUA ALGORITHM**
R11	**VAR R12**
R12	**IN R13**
R13	**STRING R14**
R14	**TITIKDUA ALGORITHM**
FUNCTIONCALL	**VAR S11 | VAR T11 | TYPE U11 | FUNCTIONCALL W11**
S11	**BK S12**
S12	**VAR3 TK**
T11	**BK T12**
T12	**TK**
U11	**BK U12**
U12	**VAR3 TK**
W11	**BK W12**
W12	**FUNCTIONCALL TK**
RANGESTATEMENT	**RANGE X11 | RANGE Y11 | RANGE Z11**
X11	**BK X12**
X12	**CAR3 TK**
Y11	**BK Y12**
Y12	**NUMBER Y13**
Y13	**KOMA Y14**
Y14	**NUMBER Y15**
Y15	**KOMA Y16**
Y16	**NUMBER TK**
Z11	**BK Z12**
Z12	**NUMBER Z13**
Z13	**KOMA Z14**
Z14	**NUMBER TK**
INLINERANGE	**BK A21**
A21	**VAR3 TK**
INLINEFOR	**VAR B21 | VAR C21**

B21	**FOR B22**
B22	**VAR B23**
B23	**IN INLINERANGE**
C21	**FOR C22**
C22	**VAR C23**
C23	**IN STRING**
FROMSTATEMNET	**FROM D21**
D21	**VAR IMPORT**
IMPORTSTATEMENT	**IMPORT VAR | IMPORT E21**
E21	**VAR E22**
E22	**AS VAR**
WITHSTATEMENT	**WITH F21**
F21	**VAR F22**
F22	**BK F23**
F23	**VAR F24**
F24	**TK F25**
F25	**AS F26**
F26	**VAR F27**
F27	**TITIKDUA ALGORITHM**
COMMENT	**BPD G21 | BPS H21**
G21	**BPD G22**
G22	**BPD G23**
G23	**STRING G24**
G24	**TPD G25**
G25	**TPD TPD**
H21	**BPS H22**
H22	**BPS H23**
H23	**STRING H24**
H24	**TPS H25**
H25	**TPS TPS**
WITH	**with**
IMPORT	**import**
FROM	**from**
FOR	**for**
RANGE	**range**
CLASS	**class**
RAISE	**raise**
BK	**(**
TK	**)**
VAR	**variable**
BREAK	**break**

NONE	**None**
AND	**and**
OR	**or**
NOT	**not**
AS	**as**
CONTINUE	**continue**
SAMADENGAN	**=**
WHILE	**while**
TITIKDUA	**:**
NUMBER	**number**
KOMA	**,**
STRIP	**-**
IF	**if**
ELIF	**elif**
ELSE	**else**
PRINT	**print**
INPUT	**input**
PASS	**pass**
DEF	**def**
RETURN	**return**
IS	**is**
IN	**in**
BPD	**"**
TPD	**"**
BPS	**'**
TPS	**'**
BKS	**[**
TKS	**]**
BKK	**{**
TKK	**}**
S0	**ALGORITHM ALGORITHM | WHILE A01 | WHILE B01 | WHILE C01 | VAR D01 | VAR E01 | VAR F01 | VAR G01 | VAR H01 | VAR X01 | VAR Y01 | FOR P11 | FOR Q11 | FOR R11 | DEF G11 | DEF H11 | CLASS I11 | IF J11 | IF K11 | IFSTATEMENT ELIFSTATEMENT | IFSTATEMENT ELSESTATEMENT | IFSTATEMENT BREAK | IFSTATEMENT CONTINUE | IFSTATEMENT RAISESTATEMENT | IFSTATEMENT PASS | import | from | WITH F21 | VAR S11 | VAR T11 | TYPE U11 | FUNCTIONCALL W11 | PRINT P01 | PRINT Q01 | INPUT R01 | INPUT S01 | TYPE T01**

ISIVARNUMBER	**NUMBER J01 | number | STRIP NUMBER | STRIP FLOAT**
VAR3	**variable | number | NUMBER J01 | STRIP NUMBER | STRIP FLOAT**
VAR6	**BPD U01 | BPS W01 | variable | number | NUMBER J01 | STRIP NUMBER | STRIP FLOAT**

S (Start State)

Start State → ALGORITHM

2.2 Implementasi Program

2.1.1 CFG2CNF.py

CFG2CNF.py adalah program yang digunakan untuk mengkonversi CFG menjadi CNF. Program kami direfernsikan dari https://github.com/adelmassimo/CFG2CNF dengan modifikasi . CFG2CNF memiliki beberapa fungsi berikut:

No.	Fungsi/Prosedur	Description
1	isUnitary	Return true jika rules Unitary
2	isSimple	Return true jika rules Simple
3	START	Add S0->S rule
4	TERM	Remove rules containing both terms and variables, like A->Bc, replacing by A->BZ and Z->c
5	BIN	Eliminate non unitary rules
6	DEL	Delete non terminal rules
7	unit_routine	Check if rules are unitary or simple
8	UNIT	Delete unit production rules

2.1.2 CheckVarName.py

CheckVarName.py berfungsi untuk mengecek secara FA yang dihardcode apabila sebuah string memenuhi ketentuan untuk menjadi sebuah nama variabel.

No.	Fungsi/Prosedur	Description
1	CheckAtoz	Mengecek apabila input(char) adalah karakter dari a...z atau A...Z. Return True if yes and False if no.
2	Check0to9	Mengecek apabila input(char) adalah karakter dari 0..9.

		Return True if yes and False if no.
3	CheckVariableName	Mengecek apabila input (string) adalah nama variabel yang valid. Aturan variabel valid : terdiri dari a..z,A..Z,0..9 atau '_' dan dimulai dengan huruf. Return True if yes and False if no.

2.1.3 cyk.py

cyk.py adalah program utama yang menggunakan algoritma CYK untuk mengecek kebenaran compile sebuah program. Program membaca rules cnf, mengkonvertnya menjadi rules yang bisa digunakan program, membaca input lalu mengoutput apakah valid atau tidak.

No.	Fungsi/Prosedur	Description
1	read_cnf	Membaca CNF dan menyimpannya.
2	convert_cnf	Mengkonversi CNF yang sudah dibaca menjadi aturan yang bisa digunakan program.
3	unswap_conver_cnf	Konversion cnf ver2.
4	separator	Fungsi bantuan untuk megubah operator '+' misalnya menjadi ' + '
5	read_inp	Membaca input lalu membersihkannya
6	cyk	Mengecek menggunakan algoritma cyk apakah file hasil read input sesuai/dapat diterima oleh kamus cnf yang diload.

2.1.4 helper.py

helper.py berisi fungsi-fungsi yang tidak termasuk dari main program tetapi dapat membantu mengkonstruksi CFG maupun CNF.

No.	Fungsi/Prosedur	Description
1	union	Mengabungkan dua list.
2	loadMode1	Meload CFG lalu dibagikan berdasarkan Terminal, Production, dan Variabel.
3	cleanProduction	Membersihkan Production lalu di-return sebagai list.
4	cleanAlphabet	Membersihkan Terminal dan Variables lalu di-return sebagai list.
5	seekAndDestroy	Menghilangkan useless variable.
6	setupDict	Menelusuri unti variabel.
7	rewrite	Membaca file CFG lalu dirubah menjadi format yang bisa dibaca yaitu dalam array
8	dict2Set	Memasukkan dictionary ke suatu list atau set.
9	pprintrules	Mengoutput rules yang ada di list.
10	prettyForm	Menggabungkan produksi yang memiliki lebih dari 1 hasil.

2.2 Pengujian Program

Pengujian inputAcc.py:

Pengujian inputAcc.py gagal, ini kemungkinan dikarenakan oleh ketidak-telitian di pembuatan CFG atau CNF.

Bab III

Penutup

3.1 Kesimpulan

Program yang kami buat walaupun menggunakan konsep CFG, CNF ,dan CYK, masih salah dalam eksekusinya. Masalah ini kami temukan terdapat di ketidaktelitian sedikit yang terdapat di CFG dan CNF.

3.2 Saran

Saran untuk mahasiswa selanjutnya yang akan melakukan tugas besar ini:

1. Mempelajari secara dalam dan mengerti konsep CFG, CNF, dan CYK.
2. Teliti dalam membuat CFG dan CNF.

Bab IV

Lampiran

4.1 Link Repo

- **https://github.com/JayaMangalo/TBFO-PVZ**

4.2 Pembagian Tugas

NIM	NAMA	Pembagian Tugas
13520006	Vionie Novencia Thanggestyo	CFG, CYK
13520015	Jaya Mangalo Soegeng Rahardjo	FA, Laporan
13520019	Maharani Ayu Putri Irawan	CNF, CYK

REFERENSI

https://github.com/adelmassimo/CFG2CNF diakses pada 23 November 2021

https://www.xarg.org/tools/cyk-algorithm/ diakses pada 23 November 2021