PARSER BAHASA PYTHON

LAPORAN TUGAS BESAR

Diajukan Untuk Memenuhi Tugas Besar Teori Bahasa Formal dan Otomata

Kelompok:

GEPREK CRISBAR

HILYA FADHILAH IMANIA	13520024
MUHAMMAD RISQI FIRDAUS	13520043
FIKRI KHOIRON FADHILA	13520056

TEKNIK INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2021

BAB I
DASAR TEORI

1. LEXER PREPROCESSING

Lexical analyzer adalah fase pertama pada compiler. Lexer memodifikasi sebuah program melalui pemrosesan awal. Lexer memecah syntax menjadi deretan token dengan menghapus spasi atau comment pada source code.

Lexer memanfaatkan regex untuk menyeleksi deretan source agar sesuai dengan bentuk non-terminal sesuai yang diekspektasikan.

Pembentukan nonterminal pada lexer mengikuti pasangan set pada array rules yang diberikan. Selanjutnya lexer akan diproses mengikuti grammar yang ditentukan.

Pada tugas kali ini, lexer dibuat dengan finite automat. Program dibuat dengan memanfaatkan class atau object pada python. String atau file dibaca layaknya mesin kata, di mana pembacaan per karakter kemudian dicocok dan dengan enumeration token yang sudah disiapkan sejak awal.

2. CONTEXT-FREE GRAMMAR

Dalam teori bahasa formal, CFG adalah grammar formal yang memproduksi aturan dengan bentuk

$$A \rightarrow a$$

Language yagn diproduksi CFG disebut denan CFL, context free language. CFG pada linguistic dapat digunakan untuk mendeskripsikan struktur kalimat dan kata dalam bahasa natural. Setiap dari grammar regular ada CFG. Namun, tidak semua CFG reguler.

Penggunaan CFG pada kompiler bermaksud untuk mempermudah serta mengurangi ambiguitas dalam parsing. Parsing sendiri merupakan proses pemrosesan string berdasrkan aturan CFG.

Sebagian besar syntax pad abahasa pemrograman didefinisikan dalam CFG. Pada keberjalanannya CFG menggunakan pohon penurunan untuk menggambarkan simbol variabel menjadi simbol terminal.

3. CHOMSKY NORMAL-FORM

Chomsky normal form (CNF) adalah bentuk (*form*) tersimplifikasi dari CFG. Sebuah grammar dikatakan berbentuk CNF jika semua aturan produksinya berbentuk salah satu di antara:

$A \rightarrow BC$

$A \rightarrow a$

$S \rightarrow \epsilon$

dengan A, B, C adalah simbol nonterminal (B dan C bukan simbol start), a adalah simbol terminal, S adalah simbol start, dan ϵ adalah empty string. Bentuk ketiga hanya valid apabila empty string juga valid dalam grammar tersebut.

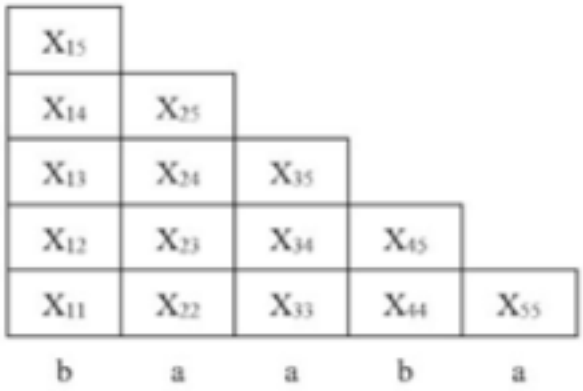
Langkah-langkah pengubahan CFG umum ke CNF adalah sebagai berikut:

1. Apabila simbol start muncul di right-hand side pada salah satu rule, buat produksi baru sebagai simbol start
2. Simplifikasi CFG dengan urutan
    a. Hapus dan ganti null productions
    b. Hapus dan ganti unit productions
    c. Hapus useless productions
3. Dekomposisi aturan yang mengandung campuran terminal dan nonterminal, atau lebih dari satu terminal
4. Dekomposisi aturan yang mengandung lebih dari dua nonterminal

4. COCKE-YOUNGER KASAMI

Cocke-Younger-Kasami (CYK) Algorithm adalah sebuah algoritma yang digunakan untuk membuktikan apakah sebuah word *w* di-generate oleh grammar context free atau tidak. Algoritma CYK dikembangkan oleh John Cocke, Daniel Younger, dan Tadao Kasami. Untuk dapat menggunakan algoritma ini dibutuhkan grammar context free G dalam bentuk Chomsky normal form, dimana word *w* adalah sebagai input, dan outputnya adalah sebuah pembuktian apakah word *w* merupakan bahasa dari grammar G atau bukan.

CYK-Algorithm dapat diilustrasikan dalam bentuk tabel sebagai berikut.



Tabel diatas merupakan tabel dengan pangjang string masukan 5. Dari tabel tersebut, X_{15} adalah akar, sehingga bila akar tersebut mengandung start symbol, maka string tersebut merupakan bagian dari grammar G.

Misalkan suatu CFG didefinisikan sebagai

G = ({S, A, B,C}, {a, b}, P, S)

G mempunyai aturan produksi:

S -> AB | BC

A -> BA | a

B -> CC | b

C -> AB | a

String masukan ‘baaba; diterima oleh grammar tersebut, karena akar mengandung start symbol S.

5. BAHASA PEMROGRAMAN PYTHON

Python adalah bahasa pemrograman tingkat tinggi yang ditafsirkan, berorientasi objek, dengan semantik dinamis. Pemrograman tingkat tinggi yang dibangun dalam struktur data, dikombinasikan dengan pengetikan dinamis dan pengikatan dinamis, membuatnya sangat menarik untuk pengembangan aplikasi secara cepat, serta digunakan sebagai bahasa scripting untuk menghubungkan komponen yang ada bersama-sama. Sintaksis Python yang sederhana dan mudah dipelajari menekankan keterbacaan dan karenanya mengurangi biaya pemeliharaan program. Python mendukung modul dan paket, yang mendorong modularitas program dan penggunaan kembali kode. Interpreter Python dan pustaka standar yang luas tersedia dalam bentuk sumber atau biner tanpa biaya untuk semua platform utama, dan dapat didistribusikan

secara bebas (https://www.python.org/about/apps/). Python juga dapat dikolaborasikan dengan beberapa bahasa pemrograman seperti Java, C++, Javascript.

BAB II
ANALISIS PERSOALAN DAN DEKOMPOSISI

2.1 CFG Production

Berikut adalah context free grammar uang telah kami buat terkait membuat parser bahasa python

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

NONTERMINAL

No.	Simbol	Keterangan
	S	Start symbol
	STMTS	Statements
	STMT	
	ASSIGN	Assignment
	ASSIGNCHAIN	
	ASSIGNOP	
	EXP	General expression
	DOTEXP	
	DOTEXPR	
	PAREXP	
	PAREXP1	
	PARLOGICEXP	
	PARMATHEXP	
	LOGICEXP	
	RELOP	
	MATHEXP	
	MATHOP1	
	MATHOP2	
	FUNCEXP	
	ARR	Array declaration

	ARRLIST	
	ELMT	Elemen array / dictionary
	DICT	Dictionary declaration
	DICTLIST	
	DICT1	
	TUPLE	Tuple
	ARGS	List expressions
	ARGS1	
	LIT	Literal
	REAL	Floating point
	IMPORT	Import
	FROMPKG	
	PKG	
	COND	Conditional
	CONDTAIL	
	LOOP	Loop
	WITH	With
	RAISE	Raise
	RET	Return
	DEF	Function declaration
	DEFARGS	
	DEFARGS1	
	DEFARG	
	ARGTYPE	
	DEFRETT	
	CLASS	Class declaration
	INHERIT	
	COMMA	Simbol dengan whitespace

	COLON	
	LC	
	RC	
	LB	
	RB	
	LP	
	RP	
	NL	
	NL1	

2.2 CNF (Chomsky Normal Form)

Berikut ini adalah hasil Context-Free Grammar yang sudah menjadi bentukk Chomsky Normal Form yang telah kami dapatkan dari converter CFG ke CNF.

S0	NL	S0_S23	S0_S63	PAREXP1_S14	COLON_T1
S	NL1	S0_S24	S0_S64	PAREXP1_S15	LC_T1
STMTS	S0_T1	S0_S25	S0_S65	PAREXP1_S16	RC_T1
STMT	S0_T2	S0_S26	S0_S66	RELOP_T1	LB_T1
ASSIGNCHAIN	S0_T3	S0_S27	S0_S67	RELOP_T2	RB_T1
ASSIGNOP	S0_T4	S0_S28	S0_S68	RELOP_T3	LP_T1
EXP	S0_T5	S0_S29	S0_S69	RELOP_T4	RP_T1
DOTEXPR	S0_T6	S0_S30	S0_S70	RELOP_T5	NL_T1
PAREXP1	S0_T7	S0_S31	S0_S71	RELOP_T6	NL1_T1
RELOP	S0_T8	S0_S32	S0_S72	RELOP_T7	
MATHOP1	S0_T9	S0_S33	S0_S73	MATHOP2_T1	
MATHOP2	S0_T10	S0_S34	ASSIGNOP_T1	MATHOP2_T2	
FUNCEXP	S0_T11	S0_S35	ASSIGNOP_T2	MATHOP2_T3	
ARR	S0_T12	S0_S36	ASSIGNOP_T3	MATHOP2_T4	
ARRLIST	S0_T13	S0_S37	ASSIGNOP_T4	DICTLIST_T1	
DICT	S0_T14	S0_S38	ASSIGNOP_T5	DICTLIST_S1	
DICTLIST	S0_T15	S0_S39	ASSIGNOP_T6	DICTLIST_S2	
DICT1	S0_T16	S0_S40	ASSIGNOP_T7	ARGS_S1	
TUPLE	S0_S1	S0_S41	ASSIGNOP_T8	FROMPKG_T1	
ARGS	S0_S2	S0_S42	ASSIGNOP_T9	FROMPKG_T2	
ARGS1	S0_S3	S0_S43	ASSIGNOP_T10	FROMPKG_S1	
LIT	S0_S4	S0_S44	ASSIGNOP_T11	FROMPKG_S2	
IMPORT	S0_S5	S0_S45	ASSIGNOP_S1	CONDTAIL_T1	
FROMPKG	S0_S6	S0_S46	ASSIGNOP_S2	CONDTAIL_T2	
PKG	S0_S7	S0_S47	ASSIGNOP_S3	CONDTAIL_S1	

COND	S0_S8	S0_S48	ASSIGNOP_S4	CONDTAIL_S2
CONDTAIL	S0_S9	S0_S49	PAREXP1_T1	CONDTAIL_S3
RAISE	S0_S10	S0_S50	PAREXP1_S1	CONDTAIL_S4
DEFARGS	S0_S11	S0_S51	PAREXP1_S2	CONDTAIL_S5
DEFARGS1	S0_S12	S0_S52	PAREXP1_S3	CONDTAIL_S6
DEFRETT	S0_S13	S0_S53	PAREXP1_S4	CONDTAIL_S7
INHERIT	S0_S14	S0_S54	PAREXP1_S5	CONDTAIL_S8
COMMA	S0_S15	S0_S55	PAREXP1_S6	DEFARGS_T1
COLON	S0_S16	S0_S56	PAREXP1_S7	DEFARGS_S1
LC	S0_S17	S0_S57	PAREXP1_S8	DEFRETT_T1
RC	S0_S18	S0_S58	PAREXP1_S9	DEFRETT_T2
LB	S0_S19	S0_S59	PAREXP1_S10	DEFRETT_T3
RB	S0_S20	S0_S60	PAREXP1_S11	DEFRETT_S1
LP	S0_S21	S0_S61	PAREXP1_S12	INHERIT_S1
RP	S0_S22	S0_S62	PAREXP1_S13	COMMA_T1

2.3 FA Lexer

Parsing tiap karakter pada file input menjadi token dilakukan dengan menggunakan prinsip finite automata namun dengan modifikasi sesuai kebutuhan. Token-token yang dihasilkan inilah nantinya yang akan digunakan oleh parser CYK untuk dibandingkan. Oleh karena itu token juga menjadi terminal dalam grammar. Token-token terbagi menjadi sebagai berikut

Nama token	**Value**	**Keterangan**
Keywords		
TRUE	"true"	Literal
FALSE	"false"	
NONE	"none"	
AND	"and"	
IS	"is"	
NOT	"not"	
IN	"in"	
WITH	"with"	
AS	"as"	

BREAK	"break"	
PASS	"pass"	
CLASS	"class"	
CONTINUE	"cont"	
DEF	"def"	
IF	"if"	
ELIF	"elif"	
ELSE	"else"	
FOR	"for"	
WHILE	"while"	
FROM	"from"	
IMPORT	"import"	
RAISE	"raise"	
RETURN	"return"	
Simbol		
LP	"lp"	Left parenthesis (
RP	"rp"	)
LB	"lb"	Left bracket [
RB	"rb"	]
LC	"lc"	Left curly brace {
RC	"rc"	}
COLON	"colon"	:
DOT	"dot"	,
COMMA	"comma"	,
SHARP	"sharp"	#

TILDE	"tilde"	~
MULT	"mult"	*
DIV	"div"	/
MOD	"mod"	%
PLUS	"plus"	+
MIN	"min"	-
AMP	"amp"	&
BNOT	"bnot"	Binary not ^
BOR	"bor"	Binary or |
EXC	"exc"	Exclamation mark !
GT	"gt"	Greater than >
LT	"lt"	Less than <
EQ	"eq"	Equal =
Lainnya		
NUM	"int"	Int
XBO	"xbo"	Hex, binary, octal
STR	"str"	String
ID	"id"	Identifier
NL	"nl"	Newline
UNDEF	"undef"	Undefined
ILLEGAL	“illegal”	

BAB III
IMPLEMENTASI DAN PENGUJIAN

SPESIFIKASI TEKNIS PROGRAM

1. File cfg2cnf.py
    a. File cfg2cnff.py sebuah class yang betujuan untuk mengubah grammar cfg menjadi cnf untuk melakukan pembacaan. Berikut isi dari file tersebut:

NO.	Fungsi/Prosedur	Fungsi
1	class Cfg2Cnf()	Object utamam pad aifle cfg2cnf.py
2.	def _split_rule()	Memecah rule pada bagian terminal dan non terminal
3	def _replace_nullable	Menghapus nullable pada rule.
4	def write	Menuliskan grammar yang terkonversi ke dalam sebuah program.
5	def convert()	Mengkonfersi grammar pada cfg ke cnf.
6	def __init__(fileName, startSym)	Menginisiasi sebuah objek pengubah cfg to cnf
7	def _remove_useless()	Menghapus production yang useless
8	def _delete_symbol()	Menghapus simbol pada production, variabel dan datar termianl
9	def _traverse()	Mentraversal grammar dari production tertentu
10	def _is_repeating()	Mengecek apakah dalam grammar atau rule terdapat production berulang
11	def _is_terminal()	Mengecek apakah suatu production merupakan terminal atau bukan
12	def _extend_unique()	Menambahkan suatu val jika val trsebut tak ada dalam list
13	def _append_unique()	Menambahkan terminal unique pada list of production

14	def _remove_all()	Menghapus semua production

2. File cykParser.py
    a. File cykParser.py berisi procedure yang terkait dengan algoritma CYK. Berikut ini adalah isi dari file tersebut.

NO.	Fungsi/Prosedur	Fungsi
1	def getCNF(pathCFG)	Mengambil data dari CNF yang sudah dibuat kemudian menjadikannya sebagai dictionary dalam CYK Algorithm dengan KEY berupa Hasil dan VALUE berupa Produksi
2	Def cykParser(input)	Melakukan algoritma CYK untuk suatu input yang sudah dilakukan pre-processing menggunakan lexer, dan kemudian membuat sebuah tabel CYK dan akan mengembalikan sebuah statement apakah input merupakan syntax python yang benar sesuai grammar.

3. File Lexerr.py

File Lexerr.py merupakan file yang berisi objek untuk melalukan lexin atau preprocessing atau tokenizer pada file pyhton. Lexerr berisi algoritma yang mememecah tiap baris menjadi komponen-komponen array terminal. Berikut isi dari file Lexerr.py

NO.	Fungsi/Prosedur	Fungsi
1	def is_alpha(ch:str)	Mengcek apakah sebuah character merupakan character huruf atau simbol underscore [a-zA-Z_]
`2	Def is_digit (ch:str)	Mengcek apakah sebuah character merupakan character angka "\d”

3	def id_hex(ch)	Mengecek apakah sebuah character termasuk angka dalam hexadesimal
4	def is_oct(ch:str)	Mengecek apakah sebuah character termasuk angka dalam octadesimal
5	def is_bin(ch:str)	Mengecek apakah karakter ch merupakan karakter biner
6	def is_space(ch:str)	Mengecek apakah karakter ch merupakan sebuah blankspace
7	def parse_char(self, char:str, start: LexInput)	Membaca sebuah input string kemudian mengembalikan nilai transable dari character yang dibaca
8	def delta(self,state, input)	Mengecek apakah input merupakan transtable, jika tidak maka akan dipanggil sebagai illegal
9	Def lex(self, string:str)	Melalkukan lexing terhadap state yang ada pada string masukan.

HASIL PENGUJIAN

1. Hasil Finite Automata

Pada tahap awal, finite automata digunakan untuk membaca file python dan mengubahnya ke dalam terminal. Tiap karakter dari file dibaca, tiap pembacaan dimasukkan pada stack-stakc tertentu. Pembacaan yang tepat akan memasukkan komponen program ke sebuah stack yang pasti, jika stack pembacaan tak tersedia, maka hasil pembacaan akan dimasukkan dalam stack illegal.

Pembacaan dibagi menjadi dua sistem, yakni keywords tertentu, serta char tertentu. Char yang sesuai denga symbol tertentu akan dimasukkan ke dalam bentuk symbol tersebut (token), sedangkan string yang match dengan keywords tertentu (syntax) akan dimasukkan atau dikelompokkan pada stack tersebut. Susunan character yang tak illegal, tetapi tak match dengan keywords akan dibaca sebagai identifier (variabel).

2. Hasil CFG

Pada program ini, grammar atau CFG disimpan dalam bentuk .txt. CFG diparsing menjadi right hand side and left hand side yang kemudian dipecah menjadi bagian non terminal serta terminal. Pemecahan dari CFG akan disimpan dalam .txt berjudul CNF. Seperti yang dijelaskan pada bagian teori dasar, CNF ini lah yang digunakan komputer sebagai panduan dalam menjalankan algoritma CYK.
Sebelum menjalankan CYK, CNF disimpan dengan pasangan key value dalam bentuk dictionary. Di mana lefthand side menjadi value dan right hand side menjadi key. CYK akan mencocokkan pasangan key value tadi dengan bentukan terminal pada tabel CYK.

3. Pengerjaan Tugas Besar

Tugas besar ini dikerjakan pada repository: https://github.com/mrfirdauss-20/Tubes-TBFO
Pembagian tugas anggota kelompok:

1. Hilya Fadhilah Imani (13520024)
   -> Implementasi Lexer dengan FA
   -> cfg2cnf.py
   -> Grammar python (cfg.txt)
   -> Debug
2. Muhammad Risqi Firdaus (13520043)
   -> rules lexer
   -> main.py
   -> Laporan
   -> Debug
3. Fikri Khoiron Fadhila (13520056)
   -> cykParser.py
   -> Laporan
   -> Debug

BAB IV

HASIL PENGUJIAN

4.1 Hasil Pengujian

Berikut hasil pengujian terhadap program kami,

a. Input dan Output Sederhana

Pada pengujian pertama, program diuji dengan operasi sederhana. Program diuji dengan source code berisi input output serta operasi matematika sederhana, sebagai berikut.

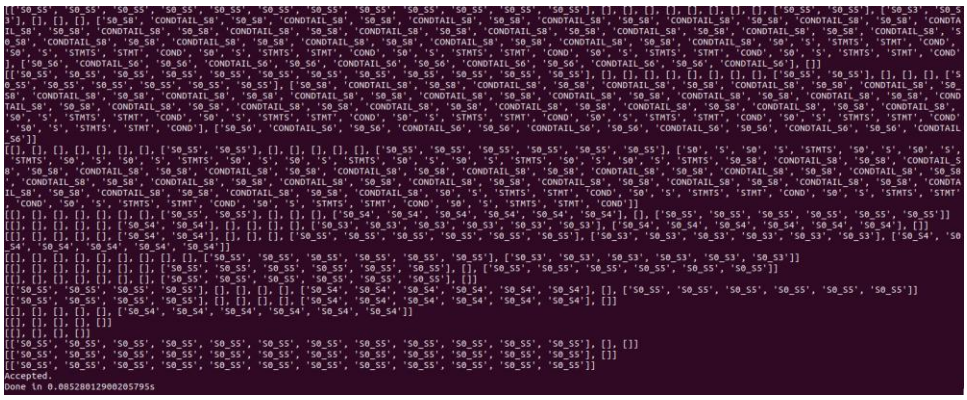
```
"""from collections import namedtuple
from enum import Enum

class X():
    a = 2
    def b():
        if(a):
            return id(self.name)
        else:
            d = 1

    def c():
        print("a")"""
```

```
Masukkkan path file python: tests/tes.py
[['S0', 'S', 'STMTS', 'STMT', 'EXP', 'PAREXP1', 'ARRLIST', 'ARGS', 'ARGS1', 'LIT
', 'DICTLIST_T1']]
Accepted.
Done in 0.004876842001976911s
```

b. Fungsi serta Conditional dan Interasi

Pada pengujian keduua, program diuji dengan syntax berupa iterasi, konsional, multiline comment serta fungsi sederhana, sebagai berikut

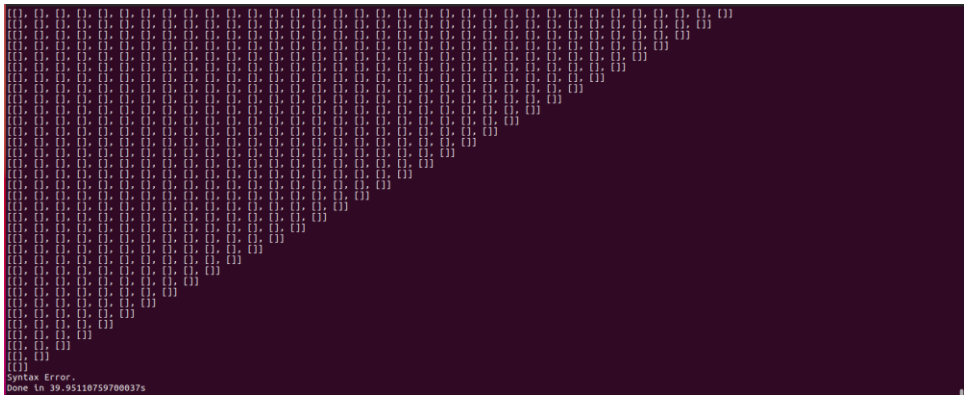
```python
def do_something(x):
    ''' This is a sample multiline comment
    '''
    while True:
        continue
    if x == 0:
        return 0
    elif x + 4 == 1:
        if True:
            return 3
        else:
            return 2
    elif x == 32:
        return 4
    else:
        return "Doodoo"
```



c. Literal dan method

Pada pengujian terakhir, program diuji dengan method serta literal lain seperti import dan pemanggilan class, sebagai berikut.

```
import sys
import time as t


if len(sys.argv) > 1:
    modelPath = str(sys.argv[1])
else:
    modelPath = 'cnf.txt'

cykParser.getCNF(modelPath)

inputFile = input("Masukkkan path file python: ")
if inputFile:
    t1 = time.perf_counter()

    lxr = Lexer.Lexer()
    lxr.lex_file(inputFile)
    t2 = time.perf_counter()
    print("Done in {t2 - t1}s", end="\n\n")

class LexState(Enum):
    START = auto()
    ZERO = auto()
    HEX = auto()
    BIN = auto()
    OCT = auto()
    STR = auto()
    STR3 = auto()
    SQUOTE = auto()
    SQUOTE1 = auto()
    SQUOTE2 = auto()
```



Pada program yang kami kembangkan, masih terdapat kekurangan seperti pada kasus di atas. Di mana, seharusnya program di atas dapat memberikan nilai balikan accepted, tetapi justru syntax error. Selain itu, program di atas juga masih jauh dari kata efisien, terlihat dari run time yang menyentuh 39 detik.

BAB V
PENUTUP

5.1 Simpulan

Dari tugas besar berjudul Parser Bahasa Python maka terbukti tugas yang kami buat berjalan, meskipun masih terdapat banyak kekurangan. Dari tugas ini, dapat ditarik kesimpulan, bahwa setiap proses sangat penting dan berpengaruh pada keberjalan program, baik preprocessing hingga implementasi.

Preprocessing yang buruk akan menyebabkan proses parsing terhambat akibat dari banyaknya noise pada text yang dianalisis. Pembacaan parsing akan error jika noise pada terminal terjadi cukup fatal.

Grammar harus dikonvert menjadi pasangan key value yang lebih dipahami komputer, sehingga syarat butuh dari program ini ialah program yang dapat memecah grammar secara universal.

Algoritma CYK merupakan alogrita berjenis dynamic programming. Sehingga perlu diperhatikan kompleksitasnya.

5.2 Saran

Berikut adalah saran-saran yang kami himpun mengenai tubes besar ini.

1. Banyak menggali mengenai implementasi regular expression serta materi lain pada mata kuliah TBFO.
2. Mempunyai jiwa pembelajar yang kuat dan tangguh dalam mempelajari hal-hal baru.
3. Tingkatkan komunikasi antar anggota kelompok.
4. Jangan mudah menyerah, mulailah dari hal-hal kecil terlebih dahulu.

BAB VI

DAFTAR PUSTAKA

1. https://www.xarg.org/tools/cyk-algorithm/
2. https://www.geeksforgeeks.org/cocke-younger-kasami-cyk-algorithm/
3. https://www.geeksforgeeks.org/converting-context-free-grammar-chomsky-normal-form/
4. https://python.plainenglish.io/lets-build-an-interpreter-in-python-from-scratch-833e9929bbb8