LAPORAN TUGAS BESAR
TEORI BAHASA DAN AUTOMATA

I Wayan Ardi Satya Putra - 1301201397

Intan Fauzia Anwar - 1301204539

Muammar Fajar Rahmadani - 1301204129

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

2022

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Manusia merupakan makhluk sosial yang membutuhkan manusia lainnya untuk saling berkomunikasi. Oleh karena itu dibutuhkan suatu bahasa yang dapat dimengerti. Dan bahasa yang digunakan oleh manusia adalah bahasa alami.

Arti dari bahasa alami itu sendiri adalah bahasa yang dapat dikemukakan, ditulis atau diisyaratkan (secara visual atau isyarat lain) oleh manusia untuk berkomunikasi.

Terdapat suatu bahasa yang tidak terdapat pembatasan tata bahasa dalam hasil produksinya. Bahasa ini diciptakan oleh para ilmuwan yang terinspirasi dengan adanya bahasa alami, para ilmuwan ini mengembangkan bahasa pemrograman dengan memberikan grammar kedalam bahasa tersebut secara formal, grammar ini diciptakan secara bebas-konteks dan pada akhirnya disebut dengan *Context Free Grammar*.

BAB II

PENYELESAIAN

2.1 Context Free Grammar

Pada tugas kali ini, Mahasiswa diminta untuk mendefinisikan suatu *Context Free Grammar* (CFG) yang dapat merepresentasikan bahasa alami atau bahasa yang digunakan oleh manusia ke dalam aturan bahasa sederhana. Dan kelompok kami memilih menggunakan bahasa Banjar dengan struktur S-V-O (*subject-verb-object*).

Untuk membuat program yang sesuai spesifikasi yang dibutuhkan, kami menggunakan *Context Free Grammar* untuk mempermudah dalam membuat aturan-aturan yang diperlukan. Pada program kali ini, kami menggunakan batasan yang dapat digunakan, dimana batasannya seperti tabel dibawah ini:

Subject	Verb	Object
Ulun, Ikam, Amang, Wadai, Iwak, Ading, Abah	Maigut, Mamakan, Manatak	Ulun, Ikam, Amang, Wadai, Iwak, Ading, Abah

G = {V, T, P, S}

V = {NN, VB}

T = {Ulun, Ikam, Amang, Wadai, Iwak, Ading, Abah}

S = {S}

Berikut adalah contoh CGF:

S → NN VB NN

NN → Ulun | Ikam | Amang | Wadai | Iwak | Ading | Abah

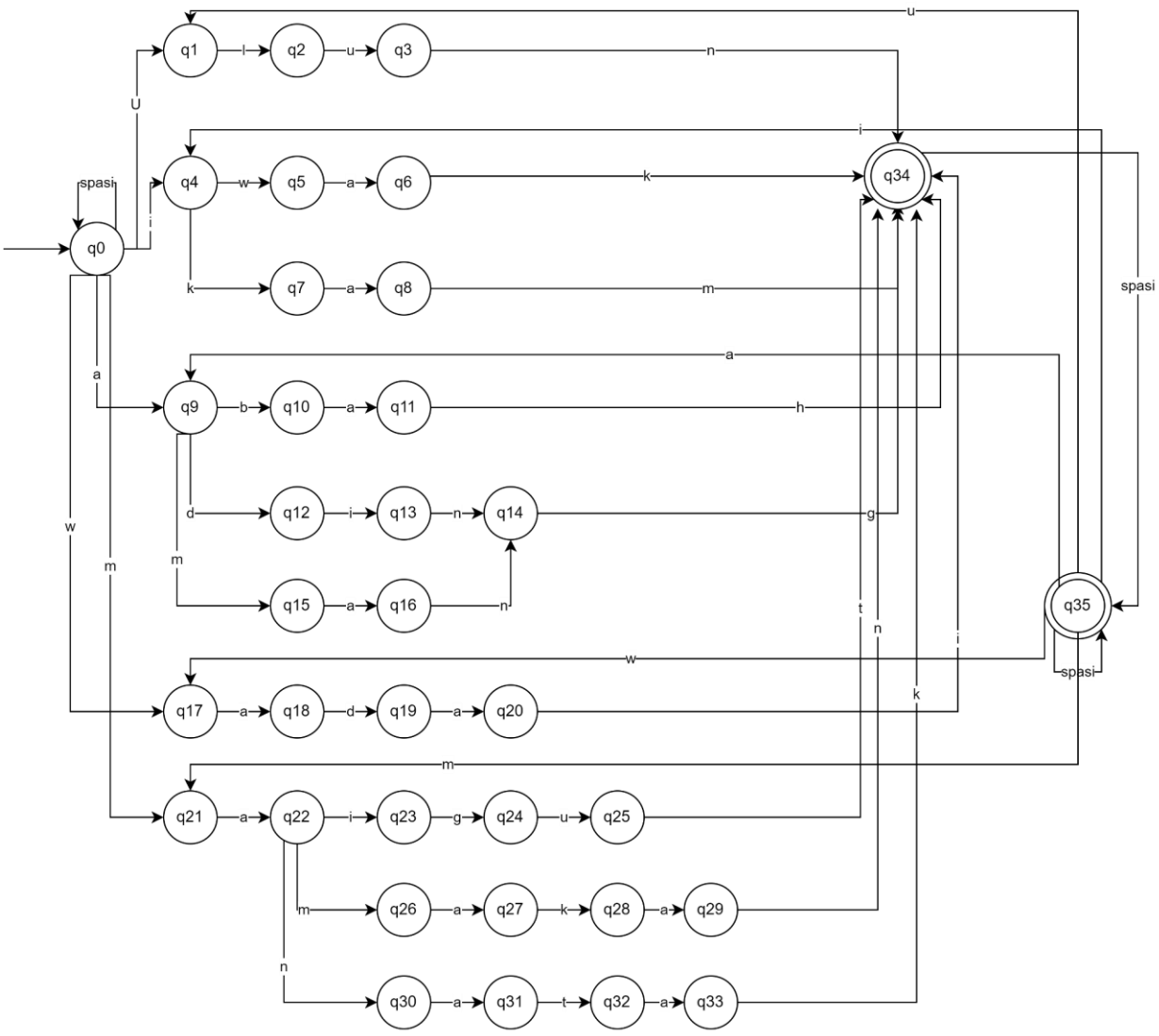
VB → Maigut | Mamakan | Manatak

2.2 Rancangan Finite Automata

Pada rancangan finite automa ini terdapat 35 state dengan q0 sebagai state awal dan 2 accepted state yaitu q34 dan q35. Finite automata ini digunakan sebagai rancangan untuk memvalidasi kata-kata dari tiap huruf dari kata yang sudah ditentukan.

a. Diagram Transisi

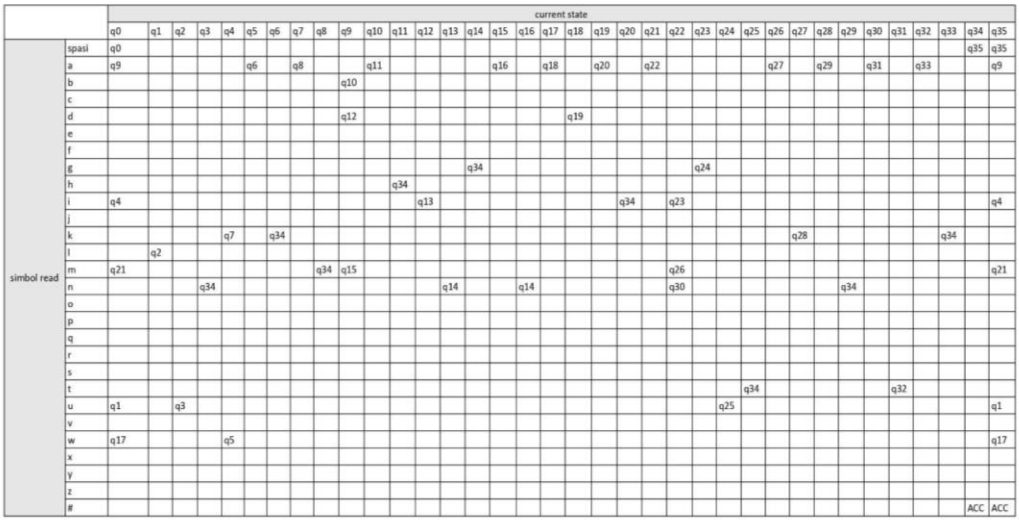
Berikut ini adalah rancangan finite automata yang sudah kami buat untuk setiap katanya.



File bisa dilihat [disini]

b. Tabel Transisi

Berikut ini adalah tabel transisi yang sudah kami buat untuk setiap katanya.



		current state																														
		q0	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12	q13	q14	q15	q16	q17	q18	q19	q20	q21	q22	q23	q24	q25	q26	q27	q28	q29	q30	q31	q32	q33	q34	q35
simbol read	spasi	q0																																		q35	q35
	a	q9					q6		q8			q11					q16		q18		q20		q22						q27		q29		q31		q33			q9
	b										q10																											
	c																																					
	d										q12									q19																		
	e																																					
	f																																					
	g															q34									q24														
	h												q34																									
	i	q4													q13							q34		q23													q4	
	j																																					
	k					q7		q34																					q28						q34			
	l		q2																																			
	m	q21								q34	q15													q26														q21
	n				q34										q14			q14						q30								q34						
	o																																					
	p																																					
	q																																					
	r																																					
	s																																					
	t																										q34							q32				
	u	q1		q3																						q25											q1	
	v																																					
	w	q17				q5																																q17	
	x																																					
	y																																					
	z																																					
	#																																				ACC	ACC

File bisa dilihat disini

2.3 Tabel Parser

Parse adalah serangkaian perintah pada program dan dipisahkan menjadi komponen yang lebih mudah diproses, yang dianalisis untuk sintaks yang benar dan kemudian dilampirkan ke tag yang menentukan setiap komponen.

Parsing adalah memecah sebuah kalimat atau kelompok kata menjadi komponen yang terpisah, termasuk definisi fungsi atau bentuk setiap bagian yang dapat lebih mudah untuk dimengerti. Pada tugas kali ini, kami akan melakukan parsing dengan menggunakan parse tabel.
Berikut adalah parse table dari kata-kata yang sudah ditentukan sebelumnya oleh kelompok kami:

	Ulun	**Ikam**	**Iwak**	**Abah**	**Ading**	**Amang**	**Wadai**	**Maigut**	**Manatak**	**Mamakan**	**EOS**
S	**NN** **VB** **NN**	**NN** **VB** **NN**	**NN** **VB** **NN**	**NN** **VB** **NN**	**NN** **VB** **NN**	**NN** **VB** **NN**	**NN** **VB** **NN**	**error**	**error**	**error**	**error**

NN	**ulun**	**ikam**	**iwak**	**abah**	**ading**	**amang**	**wadai**	**error**	**error**	**error**	**error**
VB	**error**	**error**	**error**	**error**	**error**	**error**	**error**	**maigut**	**manatak**	**mamakan**	

BAB III
PROGRAM

3.1 Lexical analyzer

Lexical analyzer merupakan program yang digunakan untuk memvalidasi setiap kata yang telah ditentukan. Kelompok kami menggunakan bahasa python sebagai bahasa pemrograman yang akan digunakan untuk membuat program lexical analyzer.

a. Code Program

Berikut adalah program code python lexical analyzer berdasarkan Finite Automata yang telah kami buat sebelumnya :

```
#lexical Analyzer

from pickle import TRUE
import string

#input str
#print("-------|  Tubes Teori Bahasa & Automata - Kelompok 11 - IF4412  |-------")
#print(" I Wayan Ardi Satya Putra - Intan Fauzia Anwar - Muamar Fajar Rahmadani ")
#print("       1301201397              1301204539              1301204129\n")
#print("Terminal Kata : ulun | ikam | amang | wadai | iwak | ading")
#print("                abah | maigut | mamakan | manatak ")
#kalimat = input("Masukan Kata yang di cari: ")
def lexical(sentence):
    input_string = sentence.lower() + "#"

    #initialization
    alphabet_list = list(string.ascii_lowercase)
    state_list = [
    "q0", "q1", "q2", "q3", "q4", "q5", "q6", "q7", "q8", "q9", "q10", "q11", "q12", "q13", "q14",
"q15", "q16",
    "q17", "q18", "q19", "q20", "q21", "q22", "q23", "q24", "q25", "q26", "q27", "q28", "q29",
"q30", "q31", "q32",
    "q33", "q34","q35"
```

```
]

transition_table = {}

for i in state_list:
    for alphabet in alphabet_list:
        transition_table[(i, alphabet)] = "ERROR"
    transition_table[(i, "#")] = "ERROR"
    transition_table[(i, " ")] = "ERROR"

# CFG
# S → NN VB NN
# NN → Ulun | Ikam | Amang | Wadai | Iwak | Ading | Abah
# VB → Maigut | Mamakan | Manatak

# For starting node (q0)
    transition_table[("q0", " ")] = "q0"

# Finish state
    transition_table[("q34", "#")] = "ACCEPT"
    transition_table[("q34", " ")] = "q35"

    transition_table[("q35", "#")] = "ACCEPT"
    transition_table[("q35", " ")] = "q35"

# string "Ulun"
    transition_table[("q35", "u")] = "q1"
    transition_table[("q0", "u")] = "q1"
    transition_table[("q1", "l")] = "q2"
    transition_table[("q2", "u")] = "q3"
    transition_table[("q3", "n")] = "q34"

# string "Ikam"
transition_table[("q35", "i")] = "q4"
```

```
    transition_table[("q0", "i")] = "q4"
    transition_table[("q4", "k")] = "q7"
    transition_table[("q7", "a")] = "q8"
    transition_table[("q8", "m")] = "q34"

    # string "Amang"
    transition_table[("q35", "a")] = "q9"
    transition_table[("q0", "a")] = "q9"
    transition_table[("q9", "m")] = "q15"
    transition_table[("q15", "a")] = "q16"
    transition_table[("q16", "n")] = "q14"
    transition_table[("q14", "g")] = "q34"

    # string "Wadai"
    transition_table[("q35", "w")] = "q17"
    transition_table[("q0", "w")] = "q17"
    transition_table[("q17", "a")] = "q18"
    transition_table[("q18", "d")] = "q19"
    transition_table[("q19", "a")] = "q20"
    transition_table[("q20", "i")] = "q34"

    # string "Iwak"
    transition_table[("q35", "i")] = "q4"
    transition_table[("q0", "i")] = "q4"
    transition_table[("q4", "w")] = "q5"
    transition_table[("q5", "a")] = "q6"
    transition_table[("q6", "k")] = "q34"

    # string "Ading"
    transition_table[("q35", "a")] = "q9"
    transition_table[("q0", "a")] = "q9"
    transition_table[("q9", "d")] = "q12"
    transition_table[("q12", "i")] = "q13"
    transition_table[("q13", "n")] = "q14"
```

```
    transition_table[("q14", "g")] = "q34"

    # string "Abah"
    transition_table[("q35", "a")] = "q9"
    transition_table[("q0", "a")] = "q9"
    transition_table[("q9", "b")] = "q10"
    transition_table[("q10", "a")] = "q11"
    transition_table[("q11", "h")] = "q34"

    # string "Maigut"
    transition_table[("q35", "m")] = "q21"
    transition_table[("q0", "m")] = "q21"
    transition_table[("q21", "a")] = "q22"
    transition_table[("q22", "i")] = "q23"
    transition_table[("q23", "g")] = "q24"
    transition_table[("q24", "u")] = "q25"
    transition_table[("q25", "t")] = "q34"

    # string "Mamakan"
    transition_table[("q35", "m")] = "q21"
    transition_table[("q0", "m")] = "q21"
    transition_table[("q21", "a")] = "q22"
    transition_table[("q22", "m")] = "q26"
    transition_table[("q26", "a")] = "q27"
    transition_table[("q27", "k")] = "q28"
    transition_table[("q28", "a")] = "q29"
    transition_table[("q29", "n")] = "q34"

    # string "Manatak"
    transition_table[("q35", "m")] = "q21"
    transition_table[("q0", "m")] = "q21"
    transition_table[("q21", "a")] = "q22"
    transition_table[("q22", "n")] = "q30"
    transition_table[("q30", "a")] = "q31"
```

```
transition_table[("q31", "t")] = "q32"
transition_table[("q32", "a")] = "q33"
transition_table[("q33", "k")] = "q34"

# lexical Analysis
idx_char = 0
state = "q0"
current_token = ""
while state != "ACCEPT":
    current_char = input_string[idx_char]
    current_token += current_char
    print(state, current_char)
    state = transition_table[(state, current_char)]
    if state == "q34":
        print("current token: {} is valid".format(current_token))
        current_token = ""
    if state == "ERROR":
        print("error")
        break
    idx_char += 1

# Conclusion
if state == "ACCEPT":
    print("semua token yang di input: {} valid".format(input_string))
    return True
```

b. Input dan Output dari code program

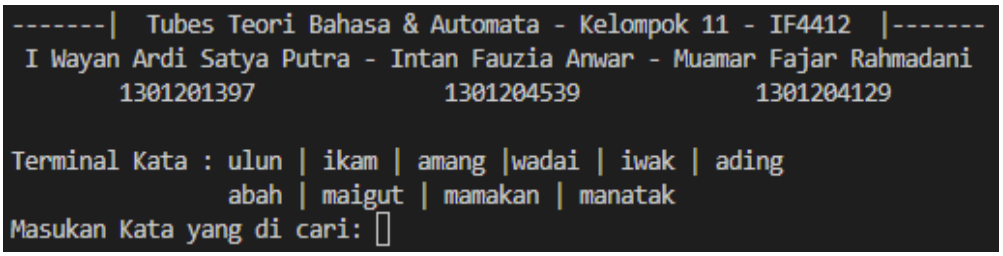
```
-------|  Tubes Teori Bahasa & Automata - Kelompok 11 - IF4412  |-------
 I Wayan Ardi Satya Putra - Intan Fauzia Anwar - Muamar Fajar Rahmadani
        1301201397              1301204539             1301204129

Terminal Kata : ulun | ikam | amang | wadai | iwak | ading
                abah | maigut | mamakan | manatak
Masukan Kata yang di cari: ulun ikam amang wadai iwak ading abah maigut mamakan manatak
q0 u
q1 l
q2 u
q3 n
current token: ulun is valid
q34
q35 i
q4 k
q7 a
q8 m
current token:  ikam is valid
q34
q35 a
q9 m
q15 a
q16 n
q14 g
current token:  amang is valid
q34
q35 w
q17 a
q18 d
q19 a
q20 i
current token:  wadai is valid
q34
q35 i
q4 w
q5 a
q6 k
current token:  iwak is valid
```

Petunjuk Untuk Menjalankan source code :

1. Lakukan instalasi bahasa pemrograman python
2. Download source code dari link di bawah ini: https://colab.research.google.com/drive/1xI2iZlIWgIObsPpaO72EhIDCYxzdYFv5?usp=sharing
3. Jalankan program tersebut dengan menggunakan python

4. Pada tahap seperti pada gambar di bawah ini, lakukan pengisian kata-kata yang ingin diperiksa oleh lexical analyzer yang telah dibuat oleh kami



5. Jika output yang diberikan program tersebut terdapat ‘error’ maka kata atau kalimat yang anda masukkan tidak valid dengan lexical analyzer yang telah kami rancang.
6. Lalu run program tersebut

3.2 Program Parser

Pada program parser ini dilakukan untuk memvalidasi susunan kalimat yang sesuai dengan context free grammar yang sudah dibuat. Program ini akan memberikan keluaran valid dan grammar yang dihasilkan benar apabila susunan inputan kata adalah NN-VB-NN, dimana NN adalah kata benda dan VB adalah kata kerja. Apabila tidak sesuai dengan urutan tersebut program akan memberikan keluaran tidak valid dan grammar tidak diterima. Kami menggunakan Javascript untuk digunakan pada program parser kami.

a. Code Program

Berikut ini adalah code program untuk program parser.

```
#sentence = input('input kalimat : ')
#from tkinter import FALSE, TRUE

def parser(sentence):
  tokens = sentence.lower().split()
  tokens.append('EOS')

  #symbol
  non_terminals = ['S','NN','VB']

terminals=['ulun','ikam','amang','wadai','iwak','ading','abah','maigut','mamakan','manat
ak']

  #parse table
  parse_table ={}

  parse_table[('S', 'ulun')] = ['NN','VB','NN']
  parse_table[('S', 'ikam')] = ['NN','VB','NN']
  parse_table[('S', 'amang')] = ['NN','VB','NN']
  parse_table[('S', 'wadai')] = ['NN','VB','NN']
  parse_table[('S', 'iwak')] = ['NN','VB','NN']
  parse_table[('S', 'ading')] = ['NN','VB','NN']
  parse_table[('S', 'abah')] = ['NN','VB','NN']
  parse_table[('S', 'maigut')] = ['error']
  parse_table[('S', 'mamakan')] = ['error']
```

```
parse_table[('S', 'manatak')] = ['error']
parse_table[('S', 'EOS')] = ['error']

parse_table[('NN', 'ulun')] = ['ulun']
parse_table[('NN', 'ikam')] = ['ikam']
parse_table[('NN', 'amang')] = ['amang']
parse_table[('NN', 'wadai')] = ['wadai']
parse_table[('NN', 'iwak')] = ['iwak']
parse_table[('NN', 'ading')] = ['ading']
parse_table[('NN', 'abah')] = ['abah']
parse_table[('NN', 'maigut')] = ['error']
parse_table[('NN', 'mamakan')] = ['error']
parse_table[('NN', 'manatak')] = ['error']
parse_table[('NN', 'EOS')] = ['error']

parse_table[('VB', 'ulun')] = ['error']
parse_table[('VB', 'ikam')] = ['error']
parse_table[('VB', 'amang')] = ['error']
parse_table[('VB', 'wadai')] = ['error']
parse_table[('VB', 'iwak')] = ['error']
parse_table[('VB', 'ading')] = ['error']
parse_table[('VB', 'abah')] = ['error']
parse_table[('VB', 'maigut')] = ['maigut']
parse_table[('VB', 'mamakan')] = ['mamakan']
parse_table[('VB', 'manatak')] = ['manatak']
parse_table[('NN', 'EOS')] = ['error']

# stack initialization
stack = []
stack.append('#')
stack.append('S')

# input reading initialization
idx_token = 0
```

```
symbol = tokens[idx_token]

#
while (len(stack) > 0):
  top = stack [len(stack) - 1]
  if top in terminals:
    if top == symbol:
      stack.pop()
      idx_token = idx_token + 1
      symbol = tokens[idx_token]
      if symbol == 'EOS':
        stack.pop()
    else:
      print('error')
      break;
  elif top in non_terminals:
    if parse_table[(top, symbol)][0] != 'error':
        stack.pop()
        symbols_to_be_pushed = parse_table[(top, symbol)]
        for i in range(len(symbols_to_be_pushed)-1,-1,-1):
            stack.append(symbols_to_be_pushed[i])
    else:
        print('error')
        break;
  else:
    print('error')
    break;
print()

# conclusion
print()
if symbol == 'EOS' and len(stack) == 0:
    print('Input string ', sentence, ' diterima, sesuai Grammar')
    return True
```

```
else:
    print('Error, input string: ', sentence, ', tidak diterima, tidak sesuai Grammar')
    return False
```

```
PS C:\Belajar Python\python> & "C:/Program Files/Python310/python.exe" "c:/Belajar Python/python/new.py"
input kalimat : ulun maigut ikam


Input string  ulun maigut ikam  diterima, sesuai Grammar
PS C:\Belajar Python\python> & "C:/Program Files/Python310/python.exe" "c:/Belajar Python/python/new.py"
input kalimat : ikam mamakan iwak


Input string  ikam mamakan iwak  diterima, sesuai Grammar
PS C:\Belajar Python\python> & "C:/Program Files/Python310/python.exe" "c:/Belajar Python/python/new.py"
input kalimat : ikam ulun mamakan
error


Error, input string:  ikam ulun mamakan , tidak diterima, tidak sesuai Grammar
```

b. Input dan Output dari code program

Petunjuk Untuk Menjalankan source code :

1. Lakukan instalasi bahasa pemrograman python
2. Download source code dari link di bawah ini: https://colab.research.google.com/drive/1Cmx4aFNEne2gDyTRddkJbrCFktbfN4aD#scrollTo=-bfyWkuiMQA6
3. Jalankan program tersebut dengan menggunakan python
4. Pada tahap seperti pada gambar di bawah ini, lakukan pengisian kata-kata yang ingin diperiksa oleh parser yang telah dibuat oleh kami
5. Jika output yang diberikan program tersebut terdapat ‘error’ maka kata atau kalimat yang anda masukkan tidak valid dengan parser yang telah kami rancang.
6. Lalu run program tersebut

3.3 Program WEB

Berikut adalah link web dari program Lexical Analyzer dan Parser yang telah kelompok kami buat

https://lexical-analyzer-parser-kelompok11.netlify.app

BAB IV
KESIMPULAN

Kesimpulan penyelesaian dari tugas besar yang telah Kami buat dapat disimpulkan bahwa, dengan Bahasa Banjar yang memiliki struktur S-V-O (subject-verb-object) dapat dibuat *context free grammar, finite automata, lexical analyzer, parse table, dan program parser*. Dan dari 5 hal tersebut dapat menunjukkan validasi dari susunan kata dalam Bahasa Banjar