Tugas Besar Teori Bahasa dan Automata: Lexical Analyzer dan Parser untuk Teks Bahasa Alami dalam Bahasa Sunda

Diajukan untuk memenuhi salah satu tugas mata kuliah CII2l3 - Teori Bahasa dan Automata

Disusun oleh:

Ryan Oktaviandi Susilo Wibowo	**1301204289**
Muhammad Khalid Habiburahman	**1301204437**
Raihan Atsal Hafizh	**1301204485**

Kelompok 2

IF-44-09

PROGRAM STUDI S1 INFORMATIKA

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

2022

Daftar Isi

BAB I

PENDAHULUAN

1.1 Finite Automata

Finite Automata merupakan mesin otomata dari suatu Bahasa regular. *Finite automata* adalah mesin abstrak berupa sistem model matematika dengan masukan dan keluaran diskrit yang dapat mengenali bahasa paling sederhana atau disebut dengan bahasa reguler dan dapat diimplementasikan secara nyata di mana sistem dapat berada di salah satu dari sejumlah berhingga konfigurasi internal disebut state. *Finite Automata* memiliki jumlah state yang banyaknya berhingga dan dapat berpindah-pindah dari suate *state* ke *state* yang lainnya. *Finite Automata* dibagi menjadi *Deterministic Finite Automata* (DFA) dan *Non-Deterministic Finite Automata* (NFA).

1.2 Context Free Grammar

CFG adalah singkatan dari *Context Free Grammar*. CFG adalah tata bahasa formal yang digunakan untuk menghasilkan semua kemungkinan pola string dalam bahasa formal tertentu. CFG dikenali oleh *push-down automata* (PDA) dengan cara yang sama seperti bahasa reguler yang dikenali oleh finite automata. CFG memiliki tujuan sebagai suatu cara yang dapat menghasilkan suatu untai-untai dalam sebuah bahasa. Dalam mendefinisikan aturan tata bahasa pada CFG memiliki dua jenis simbol yaitu simbol terminal dan *non-terminal*. Simbol terminal merupakan simbol alfabet yang mendasari bahasa yang dipertimbangkan, sedangkan simbol *non-terminal* yang berperilaku seperti variabel yang berkisar pada string terminal. Dalam membuat CFG perlu mendefinisikan objeknya terlebih dahulu, kemudian menjelaskan bagaimana CFG digunakan.

1.3 Lexical Analyzer

Lexical Analyzer adalah fase pertama dari kompiler yang juga dikenal sebagai pemindai. Ini mengubah program *input* tingkat tinggi menjadi urutan Token. Token leksikal adalah urutan karakter yang dapat diperlakukan sebagai unit dalam tata bahasa bahasa pemrograman. *Lexical analyzer* adalah tahapan pertama yang dilakukan pada *compiler*. Proses yang dilakukan pada tahapan ini adalah membaca program sumber karakter per karakter.

1.4 Parser

Parser adalah komponen *compiler* atau juru bahasa yang memecah data menjadi elemen yang lebih kecil untuk memudahkan terjemahan ke bahasa lain. *Parsing* digunakan untuk menurunkan *string* menggunakan aturan produksi tata bahasa. Ini digunakan untuk memeriksa penerimaan string. *Compiler* digunakan untuk memeriksa apakah *string* benar secara sintaksis atau tidak. *Parser* mengambil *input* dan membangun pohon parse.

BAB II

RANCANGAN CFG, FINITE AUTOMATA, dan PARSER TABLE

2.1 Context Free Grammar

Rancangan CFG (*Context Free Grammar*) kami dibuat menggunakan Bahasa Sunda, Bahasa Sunda sendiri adalah bahasa cabang Melayu-Polinesia dalam rumpun bahasa Austronesia yang banyak dipakai di pulau Jawa, terutama di Jawa Barat. CFG ini sendiri terdiri dari:

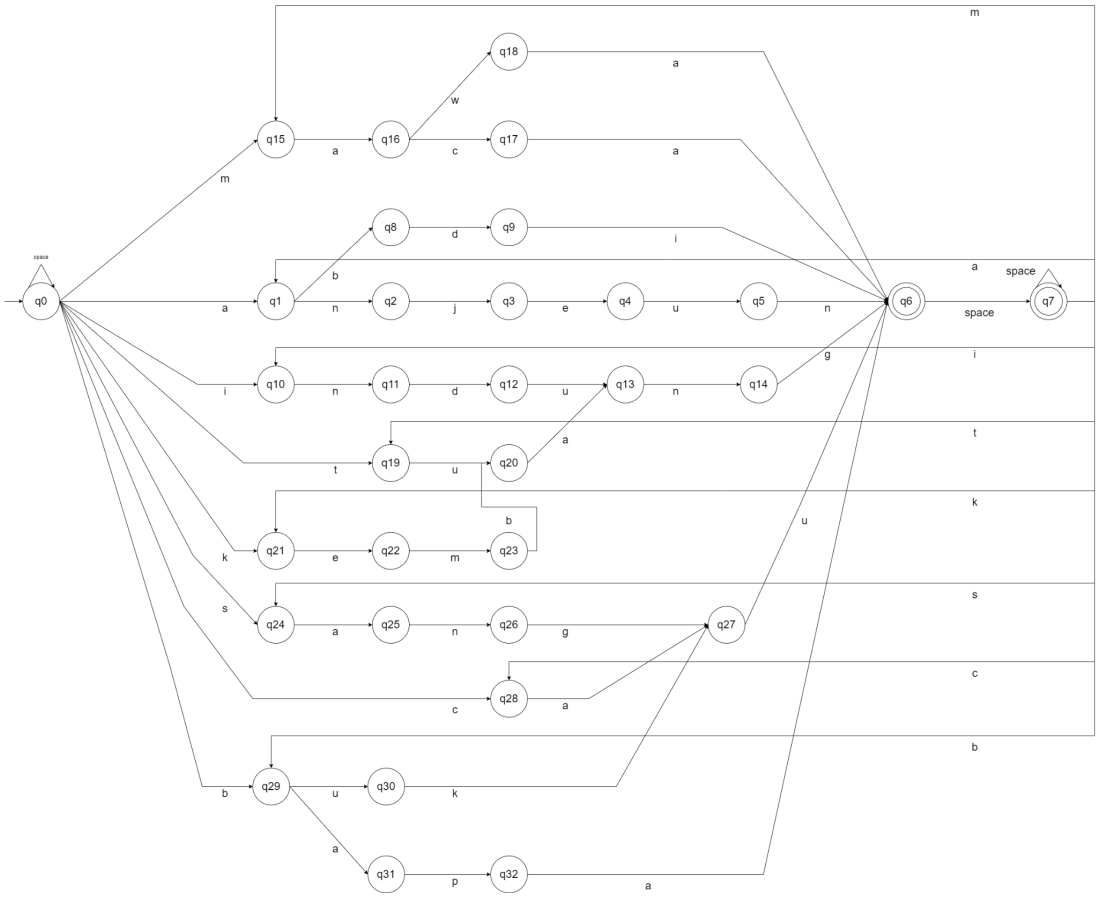
S → {NN VB OB}

NN → abdi | anjeun | indung | bapa

VB → maca | tuang | mawa

OB → buku | cau | sangu | kembang

2.2 Finite Automata



2.3 Parser Table

	abdi	anjeun	indung	bapa	maca	tuang	mawa	buku	cau	sangu	kembang	EOS
S	NN VB OB	NN VB OB	NN VB OB	NN VB OB	error	error	error	NN VB OB	NN VB OB	NN VB OB	NN VB OB	error
NN	abdi	anjeun	indung	bapa	error	error	error	error	error	error	error	error
VB	error	error	error	error	maca	tuang	mawa	error	error	error	error	error
OB	error	error	error	error	error	error	error	buku	cau	sangu	kembang	error

BAB III

PROGRAM

3.1 Code program Lexical Analyzer

Code program lexical analyzer digunakan untuk menguji setiap kata yang dimasukkan merupakan kata yang valid atau tidak. Adapun kata-kata yang akan diuji pada *program* ini yaitu ‘abdi, anjeun, indung, bapa, maca, tuang, mawa, buku, cau, sangu, kembang’.

```
import string

'''   Lexical Func   ==========================================================================================='''
def lexical(sentence, check):

    #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', 'q31', 'q32']

    transition_table = {}

    for state in state_list:
        for alphabet in alphabet_list:
            transition_table[(state, alphabet)] = "error"
        transition_table[(state, "#")] = "error"
        transition_table[(state, " ")] = "error"


    #start state
    transition_table["q0", " "] = "q0"

    #finish state
    transition_table[("q6", "#")] = "accept"
    transition_table[("q6", " ")] = "q7"

    transition_table[("q7", "#")] = "accept"
    transition_table[("q7", " ")] = "q7"

    #string abdi
    transition_table[("q0", "a")] = "q1"
    transition_table[("q1", "b")] = "q8"
    transition_table[("q8", "d")] = "q9"
    transition_table[("q9", "i")] = "q6"
    transition_table[("q6", " ")] = "q7"
    transition_table[("q7", "a")] = "q1"
```

```
    #string anjeun
    transition_table[("q0", "a")] = "q1"
    transition_table[("q1", "n")] = "q2"
    transition_table[("q2", "j")] = "q3"
    transition_table[("q3", "e")] = "q4"
    transition_table[("q4", "u")] = "q5"
    transition_table[("q5", "n")] = "q6"
    transition_table[("q6", " ")] = "q7"
    transition_table[("q7", "a")] = "q1"

    #string indung
    transition_table[("q0", "i")] = "q10"
    transition_table[("q10", "n")] = "q11"
    transition_table[("q11", "d")] = "q12"
    transition_table[("q12", "u")] = "q13"
    transition_table[("q13", "n")] = "q14"
    transition_table[("q14", "g")] = "q6"
    transition_table[("q6", " ")] = "q7"
    transition_table[("q7", "i")] = "q10"

    #string bapa
    transition_table[("q0", "b")] = "q29"
    transition_table[("q29", "a")] = "q31"
    transition_table[("q31", "p")] = "q32"
    transition_table[("q32", "a")] = "q6"
    transition_table[("q6", " ")] = "q7"
    transition_table[("q7", "b")] = "q29"

    #string maca
    transition_table[("q0", "m")] = "q15"
    transition_table[("q15", "a")] = "q16"
    transition_table[("q16", "c")] = "q17"
    transition_table[("q17", "a")] = "q6"
    transition_table[("q6", " ")] = "q7"
    transition_table[("q7", "m")] = "q15"
```

```
#string tuang
transition_table[("q0", "t")] = "q19"
transition_table[("q19", "u")] = "q20"
transition_table[("q20", "a")] = "q13"
transition_table[("q13", "n")] = "q14"
transition_table[("q14", "g")] = "q6"
transition_table[("q6", " ")] = "q7"
transition_table[("q7", "t")] = "q19"

#string mawa
transition_table[("q0", "m")] = "q15"
transition_table[("q15", "a")] = "q16"
transition_table[("q16", "w")] = "q18"
transition_table[("q18", "a")] = "q6"
transition_table[("q6", " ")] = "q7"
transition_table[("q7", "m")] = "q15"

#string buku
transition_table[("q0", "b")] = "q29"
transition_table[("q29", "u")] = "q30"
transition_table[("q30", "k")] = "q27"
transition_table[("q27", "u")] = "q6"
transition_table[("q6", " ")] = "q7"
transition_table[("q7", "b")] = "q29"

#string cau
transition_table[("q0", "c")] = "q28"
transition_table[("q28", "a")] = "q27"
transition_table[("q27", "u")] = "q6"
transition_table[("q6", " ")] = "q7"
transition_table[("q7", "c")] = "q28"

#string sangu
transition_table[("q0", "s")] = "q24"
transition_table[("q24", "a")] = "q25"
transition_table[("q25", "n")] = "q26"
transition_table[("q26", "g")] = "q27"
transition_table[("q27", "u")] = "q6"
transition_table[("q6", " ")] = "q7"
transition_table[("q7", "s")] = "q24"
```

```
    #string kembang
    transition_table[("q0", "k")] = "q21"
    transition_table[("q21", "e")] = "q22"
    transition_table[("q22", "m")] = "q23"
    transition_table[("q23", "b")] = "q20"
    transition_table[("q20", "a")] = "q13"
    transition_table[("q13", "n")] = "q14"
    transition_table[("q14", "g")] = "q6"
    transition_table[("q6", " ")] = "q7"
    transition_table[("q7", "k")] = "q21"

    #lexical analysis
    print()
    idx_char = 0
    state = 'q0'
    current_token = ''
    while state != 'accept':
        current_char = input_string[idx_char]
        current_token += current_char
        state = transition_table[(state, current_char)]
        if state == 'q6':
            print('current token        : ', current_token, ', valid')
            current_token = ''
            check = True
        if state == "error":
            print("Input Tidak Valid")
            check = False
            break
        idx_char = idx_char + 1

    #conclusion
    if state == "accept":
        print('\nsemua token diinput  : ', sentence, ', valid')
        check = True

    return check
'''=============================================================================================================='''
```

Pada *program Lexical Analyzer*, dibuat fungsi *lexical* yang memanggil *parameter check* dan *sentence*. Ketika *program* di-run, nanti akan meminta inputan. Jika user memasukan inputan sesuai dengan data yang ada pada daftar simbol *terminal*, maka akan *return* '*True*'. Namun ketika data yang dimasukan tidak terdaftar dalam daftar simbol terminal akan *return* '*False*' ketika di-run.

3.2 Pengujian Program Lexical Analyzer Dengan 3 Kata Valid Berdasarkan Daftar Simbol Terminal

Pada percobaan kali ini dimulai dengan memasukan isi *sentence* dengan kata yang telah terdaftar pada simbol *terminal*. Contoh yang digunakan yaitu 'abdi tuang sangu', 'bapa maca buku', 'indung mawa kembang'.

- Abdi tuang sangu

```
TUGAS BESAR TEORI BAHASA DAN AUTOMATA | KELOMPOK 2 | IF-44-09
============================================================
||                          terminal                        ||
||----------------------------------------------------------||
|| NN      : abdi, anjeun, indung, bapa                     ||
|| VB      : maca, tuang, mawa                              ||
|| OB      : buku, cau, sangu, kembang                      ||
============================================================

input masukan        : abdi tuang sangu

current token        :  abdi , valid
current token        :   tuang , valid
current token        :   sangu , valid

semua token diinput  :  abdi tuang sangu , valid
```

- Bapa maca buku

```
TUGAS BESAR TEORI BAHASA DAN AUTOMATA | KELOMPOK 2 | IF-44-09
============================================================
||                          terminal                        ||
||----------------------------------------------------------||
|| NN      : abdi, anjeun, indung, bapa                     ||
|| VB      : maca, tuang, mawa                              ||
|| OB      : buku, cau, sangu, kembang                      ||
============================================================

input masukan        : bapa maca buku

current token        :  bapa , valid
current token        :   maca , valid
current token        :   buku , valid

semua token diinput  :  bapa maca buku , valid
```

- Indung mawa kembang

```
TUGAS BESAR TEORI BAHASA DAN AUTOMATA | KELOMPOK 2 | IF-44-09
============================================================
||                         terminal                         ||
||----------------------------------------------------------||
|| NN      : abdi, anjeun, indung, bapa                     ||
|| VB      : maca, tuang, mawa                              ||
|| OB      : buku, cau, sangu, kembang                      ||
============================================================

input masukan        : indung mawa kembang

current token        :  indung , valid
current token        :   mawa , valid
current token        :   kembang , valid

semua token diinput  :  indung mawa kembang , valid
```

3.3 Pengujian Program Lexical Analyzer Dengan 3 Kata Tidak Valid Berdasarkan Daftar Simbol Terminal

- I eat banana

```
TUGAS BESAR TEORI BAHASA DAN AUTOMATA | KELOMPOK 2 | IF-44-09
============================================================
||                         terminal                         ||
||----------------------------------------------------------||
|| NN      : abdi, anjeun, indung, bapa                     ||
|| VB      : maca, tuang, mawa                              ||
|| OB      : buku, cau, sangu, kembang                      ||
============================================================
input masukan        : I eat banana
```

- You fly forward

```
============================================================
||                         terminal                         ||
||----------------------------------------------------------||
|| NN      : abdi, anjeun, indung, bapa                     ||
|| VB      : maca, tuang, mawa                              ||
|| OB      : buku, cau, sangu, kembang                      ||
============================================================
input masukan        : You fly foward

Input Tidak Valid
```

- Udin Take Shower

```
============================================================
||                        terminal                        ||
||--------------------------------------------------------||
|| NN      : abdi, anjeun, indung, bapa                   ||
|| VB      : maca, tuang, mawa                            ||
|| OB      : buku, cau, sangu, kembang                    ||
============================================================
input masukan        : Udin Take Shower

Input Tidak Valid
```

3.4 Code Program Parser

Pada *program* ini, kami menggabungkan *program lexical analyzer* yang sebelumnya sudah kami buat dengan *program parser* menjadi satu program. *Program* ini terdapat empat *non-terminal* yaitu S-NN-VB-OB untuk menjalankan *program*, dimana NN untuk kata subjek, VB untuk kata kerja, dan OB untuk kata objek. Adapun kata-kata atau *terminal* yang akan diuji pada *program* ini yaitu ‘abdi, anjeun, indung, bapa, maca, tuang, mawa, buku, cau, sangu, kembang’.

```
'''   Parser Func   ============================================================================================================='''
def parser(sentence, check):
    if check:
        print("\n============================================================")
        print("                          Parser                            ")
        print("------------------------------------------------------------")

        tokens = sentence.lower().split()
        tokens.append('EOS')

        non_terminals = ['S', 'NN', 'VB', 'OB']
        test = ['NN', 'VB', 'OB']
        tError = ['error']
        terminals = ['abdi', 'anjeun', 'indung', 'bapa', 'maca', 'tuang', 'mawa', 'buku', 'cau', 'sangu', 'kembang']

        parse_table = {}

        parse_table[('S', 'abdi')] = test
        parse_table[('S', 'anjeun')] = test
        parse_table[('S', 'indung')] = test
        parse_table[('S', 'bapa')] = test
        parse_table[('S', 'maca')] = tError
        parse_table[('S', 'tuang')] = tError
        parse_table[('S', 'mawa')] = tError
        parse_table[('S', 'buku')] = test
        parse_table[('S', 'cau')] = test
        parse_table[('S', 'sangu')] = test
        parse_table[('S', 'kembang')] = test
        parse_table[('S', 'EOS')] = tError
```

```
parse_table[('NN', 'abdi')] = ['abdi']
parse_table[('NN', 'anjeun')] = ['anjeun']
parse_table[('NN', 'indung')] = ['indung']
parse_table[('NN', 'bapa')] = ['bapa']
parse_table[('NN', 'maca')] = tError
parse_table[('NN', 'tuang')] = tError
parse_table[('NN', 'mawa')] = tError
parse_table[('NN', 'buku')] = tError
parse_table[('NN', 'cau')] = tError
parse_table[('NN', 'sangu')] = tError
parse_table[('NN', 'kembang')] = tError
parse_table[('NN', 'EOS')] = tError

parse_table[('VB', 'abdi')] = tError
parse_table[('VB', 'anjeun')] = tError
parse_table[('VB', 'indung')] = tError
parse_table[('VB', 'bapa')] = tError
parse_table[('VB', 'maca')] = ['maca']
parse_table[('VB', 'tuang')] = ['tuang']
parse_table[('VB', 'mawa')] = ['mawa']
parse_table[('VB', 'buku')] = tError
parse_table[('VB', 'cau')] = tError
parse_table[('VB', 'sangu')] = tError
parse_table[('VB', 'kembang')] = tError
parse_table[('VB', 'EOS')] = tError

parse_table[('OB', 'abdi')] = tError
parse_table[('OB', 'anjeun')] = tError
parse_table[('OB', 'indung')] = tError
parse_table[('OB', 'bapa')] = tError
parse_table[('OB', 'maca')] = tError
parse_table[('OB', 'tuang')] = tError
parse_table[('OB', 'mawa')] = tError
parse_table[('OB', 'buku')] = ['buku']
parse_table[('OB', 'cau')] = ['cau']
parse_table[('OB', 'sangu')] = ['sangu']
parse_table[('OB', 'kembang')] = ['kembang']
parse_table[('OB', 'EOS')] = tError
```

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

index_token = 0
symbol = tokens[index_token]

while(len(stack) > 0):
    top = stack[ len(stack) - 1 ]

    print('top = ', top)
    print('symbol = ', symbol)

    if top in terminals:
        print('top adalah simbol terminal')

        if top == symbol:
            stack.pop()
            index_token = index_token + 1
            symbol = tokens[index_token]

            if symbol == "EOS":
                stack.pop()
                print('isi stack:', stack)

        else:
            print('error')
            break

    elif top in non_terminals:
        print('top adalah symbol non-terminal')

        if parse_table[(top, symbol)][0] != 'error':
            stack.pop()
            symbol_to_be_pushed = parse_table[(top, symbol)]

            for i in range(len(symbol_to_be_pushed)-1, -1, -1):
                stack.append(symbol_to_be_pushed[i])
```

```
                    for i in range(len(symbol_to_be_pushed)-1, -1, -1):
                        stack.append(symbol_to_be_pushed[i])

                else:
                    print('error')
                    break

            else:
                print('error')
                break

            print('isi stack: ', stack)
            print("-----------------------------------------------------------------")

        print("-----------------------------------------------------------------")
        if symbol == 'EOS' and len(stack) == 0:
            print('input string ', '"', sentence, '"', 'diterima, sesuai grammar')

        else:
            print('error, input string:', '"', sentence, '"', ', tidak diterima, tidak sesuai grammar')
    else:
        pass

    print("=================================================================\n")

    return parser
'''=============================================================================================================================='''
```

3.5 Pengujian Parser dengan Kata yang Sesuai Grammar

Pada percobaan kali ini dimulai dengan memasukan isi *sentence* dengan kata yang telah terdaftar pada simbol *terminal* dan *grammar* yang sesuai. Contoh yang digunakan yaitu ‘abdi tuang sangu’, ‘bapa maca buku’, ‘indung mawa kembang’.

- Abdi tuang sangu

```
================================================================
                          Parser
----------------------------------------------------------------
top =  S
symbol =  abdi
top adalah symbol non-terminal
isi stack:  ['#', 'OB', 'VB', 'NN']
----------------------------------------------------------------
top =  NN
symbol =  abdi
top adalah symbol non-terminal
isi stack:  ['#', 'OB', 'VB', 'abdi']
----------------------------------------------------------------
top =  abdi
symbol =  abdi
top adalah simbol terminal
isi stack:  ['#', 'OB', 'VB']
----------------------------------------------------------------
top =  VB
symbol =  tuang
top adalah symbol non-terminal
isi stack:  ['#', 'OB', 'tuang']
----------------------------------------------------------------
top =  tuang
symbol =  tuang
top adalah simbol terminal
isi stack:  ['#', 'OB']
----------------------------------------------------------------
top =  OB
symbol =  sangu
top adalah symbol non-terminal
isi stack:  ['#', 'sangu']
----------------------------------------------------------------
```

```
top =  sangu
symbol =  sangu
top adalah simbol terminal
isi stack: []
isi stack:  []
------------------------------------------------------------
------------------------------------------------------------
input string  " abdi tuang sangu " diterima, sesuai grammar
============================================================
```

- Bapa maca buku

```
============================================================
                          Parser
------------------------------------------------------------
top =  S
symbol =  bapa
top adalah symbol non-terminal
isi stack:  ['#', 'OB', 'VB', 'NN']
------------------------------------------------------------
top =  NN
symbol =  bapa
top adalah symbol non-terminal
isi stack:  ['#', 'OB', 'VB', 'bapa']
------------------------------------------------------------
top =  bapa
symbol =  bapa
top adalah simbol terminal
isi stack:  ['#', 'OB', 'VB']
------------------------------------------------------------
top =  VB
symbol =  maca
top adalah symbol non-terminal
isi stack:  ['#', 'OB', 'maca']
------------------------------------------------------------
top =  maca
symbol =  maca
top adalah simbol terminal
isi stack:  ['#', 'OB']
------------------------------------------------------------
top =  OB
symbol =  buku
top adalah symbol non-terminal
isi stack:  ['#', 'buku']
------------------------------------------------------------
```

```
top =  OB
symbol =  buku
top adalah symbol non-terminal
isi stack:  ['#', 'buku']
------------------------------------------------------------
top =  buku
symbol =  buku
top adalah simbol terminal
isi stack: []
isi stack:  []
------------------------------------------------------------
------------------------------------------------------------
input string  " bapa maca buku " diterima, sesuai grammar
============================================================
```

- Indung mawa kembang

```
============================================================
                          Parser
------------------------------------------------------------
top =  S
symbol =  indung
top adalah symbol non-terminal
isi stack:  ['#', 'OB', 'VB', 'NN']
------------------------------------------------------------
top =  NN
symbol =  indung
top adalah symbol non-terminal
isi stack:  ['#', 'OB', 'VB', 'indung']
------------------------------------------------------------
top =  indung
symbol =  indung
top adalah simbol terminal
isi stack:  ['#', 'OB', 'VB']
------------------------------------------------------------
top =  VB
symbol =  mawa
top adalah symbol non-terminal
isi stack:  ['#', 'OB', 'mawa']
------------------------------------------------------------
top =  mawa
symbol =  mawa
top adalah simbol terminal
isi stack:  ['#', 'OB']
------------------------------------------------------------
top =  OB
symbol =  kembang
top adalah symbol non-terminal
isi stack:  ['#', 'kembang']
------------------------------------------------------------
```

```
top =  kembang
symbol =  kembang
top adalah simbol terminal
isi stack: []
isi stack:  []
------------------------------------------------------------
------------------------------------------------------------
input string  " indung mawa kembang " diterima, sesuai grammar
============================================================
```

3.6. Pengujian Parser dengan Kata yang tidak Sesuai Grammar

- Buku maca abdi

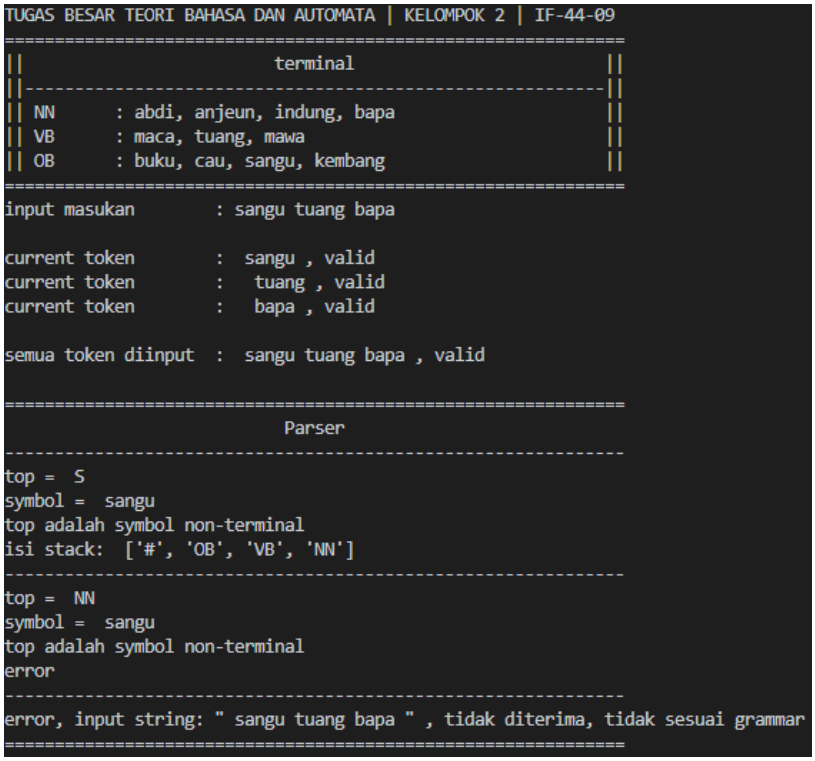
```
============================================================
||                        terminal                        ||
||--------------------------------------------------------||
|| NN      : abdi, anjeun, indung, bapa                   ||
|| VB      : maca, tuang, mawa                            ||
|| OB      : buku, cau, sangu, kembang                    ||
============================================================
input masukan        : buku maca abdi

current token        :  buku , valid
current token        :   maca , valid
current token        :   abdi , valid

semua token diinput  :  buku maca abdi , valid

============================================================
                          Parser
------------------------------------------------------------
top =  S
symbol =  buku
top adalah symbol non-terminal
isi stack:  ['#', 'OB', 'VB', 'NN']
------------------------------------------------------------
top =  NN
symbol =  buku
top adalah symbol non-terminal
error
------------------------------------------------------------
error, input string: " buku maca abdi " , tidak diterima, tidak sesuai grammar
============================================================
```

- Sangu tuang bapa



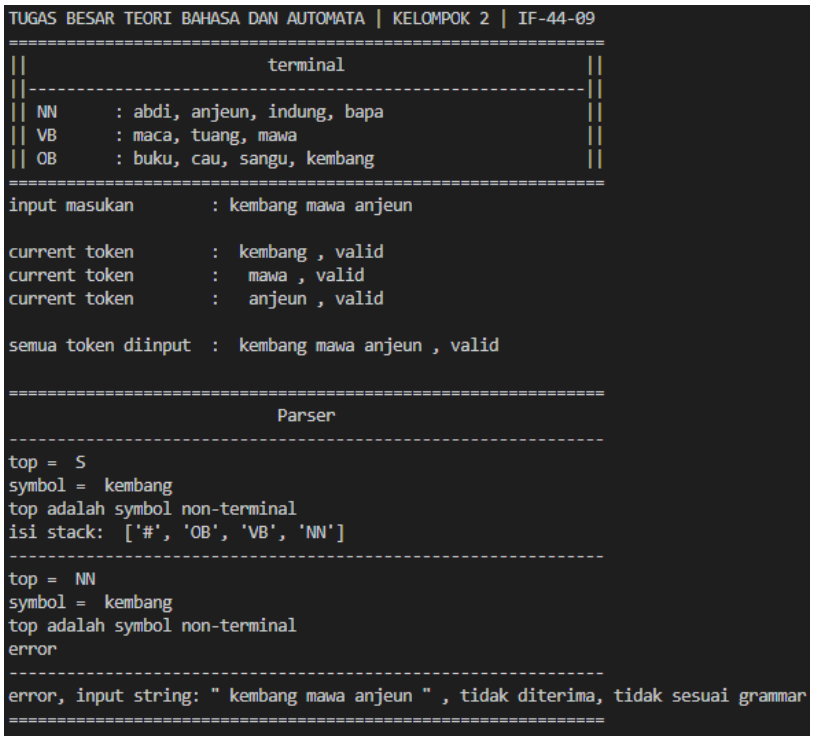
```
TUGAS BESAR TEORI BAHASA DAN AUTOMATA | KELOMPOK 2 | IF-44-09
============================================================
||                         terminal                       ||
||--------------------------------------------------------||
|| NN      : abdi, anjeun, indung, bapa                   ||
|| VB      : maca, tuang, mawa                            ||
|| OB      : buku, cau, sangu, kembang                    ||
============================================================
input masukan        : sangu tuang bapa

current token        :  sangu , valid
current token        :   tuang , valid
current token        :   bapa , valid

semua token diinput  :  sangu tuang bapa , valid

============================================================
                          Parser
------------------------------------------------------------
top =  S
symbol =  sangu
top adalah symbol non-terminal
isi stack:  ['#', 'OB', 'VB', 'NN']
------------------------------------------------------------
top =  NN
symbol =  sangu
top adalah symbol non-terminal
error
------------------------------------------------------------
error, input string: " sangu tuang bapa " , tidak diterima, tidak sesuai grammar
============================================================
```

- Kembang mawa anjeun



```
TUGAS BESAR TEORI BAHASA DAN AUTOMATA | KELOMPOK 2 | IF-44-09
============================================================
||                         terminal                       ||
||--------------------------------------------------------||
|| NN      : abdi, anjeun, indung, bapa                   ||
|| VB      : maca, tuang, mawa                            ||
|| OB      : buku, cau, sangu, kembang                    ||
============================================================
input masukan        : kembang mawa anjeun

current token        :  kembang , valid
current token        :   mawa , valid
current token        :   anjeun , valid

semua token diinput  :  kembang mawa anjeun , valid

============================================================
                          Parser
------------------------------------------------------------
top =  S
symbol =  kembang
top adalah symbol non-terminal
isi stack:  ['#', 'OB', 'VB', 'NN']
------------------------------------------------------------
top =  NN
symbol =  kembang
top adalah symbol non-terminal
error
------------------------------------------------------------
error, input string: " kembang mawa anjeun " , tidak diterima, tidak sesuai grammar
============================================================
```

3.7 Main Program

Main program berguna untuk memanggil fungsi *lexical* dan *parser* sesuai dengan parameter yang telah dibuat. *Main program* akan menjalankan semua fungsi yang telah dibuat yang nantinya akan menghasilkan *output* sesuai dengan aturan.

```
'''   Main Program   ==========================================================================================='''
check = False
print("\nTUGAS BESAR TEORI BAHASA DAN AUTOMATA | KELOMPOK 2 | IF-44-09")

while check != True:
    print("============================================================")
    print("||                          terminal                      ||")
    print("||--------------------------------------------------------||")
    print("|| NN      : abdi, anjeun, indung, bapa                   ||")
    print("|| VB      : maca, tuang, mawa                            ||")
    print("|| OB      : buku, cau, sangu, kembang                    ||")
    print("============================================================")
    sentence = input("input masukan        : ")

    input_string = sentence.lower() + '#'
    check = lexical(sentence, check)

parser(sentence, check)
'''==========================================================================================================='''
```

BAB IV

PENUTUP

4.1 Kesimpulan

Parser adalah proses untuk data atau informasi yang berarti memecahnya menjadi bagian-bagian komponen sehingga sintaksnya dapat dianalisis, dikategorikan dan dipahami.Tujuan *parser* adalah untuk melibatkan pencarian pohon *parser* untuk menemukan derivasi paling kiri dari input stream dengan menggunakan ekspansi *top-down* dan melibatkan penulisan ulang *input* kembali ke simbol awal. Cara kerja *parser* salah satunya adalah *Lexical Analytic* yang digunakan untuk menghasilkan *token* dari aliran karakter *string input*, yang dipecah menjadi komponen kecil untuk membentuk ekspresi yang bermakna. Dari hasil pengerjaan kami, kami dapat menentukan bahwa menggunakan *grammar* (tata bahasa) dengan struktur SB-VB-OB membuat *program* berjalan dengan baik, oleh karena itu jika struktur bahasa tersebut terbalik atau tertukar maka akan mempengaruhi ketidaksesuaian *grammar*.

Referensi

BAB 2 LANDASAN TEORI 2.1 Finite Automata. (n.d.). [online] Available at: http://library.binus.ac.id/eColls/eThesisdoc/Bab2/2011-2-00004-MTIF%20Bab2001.pdf [Accessed 24 May 2022].

Kompas Cyber Media (2011). Tentang Bahasa Sunda. [online] KOMPAS.com. Available at: https://edukasi.kompas.com/read/2011/07/01/04001079/Tentang.Bahasa.Sunda.%C2%A0diakses%2017%20April%202018 [Accessed 24 May 2022].

www.javatpoint.com. (n.d.). Automata Context-free Grammar | CFG - Javatpoint. [online] Available at: https://www.javatpoint.com/automata-context-free-grammar#:~:text=CFG%20stands%20for%20context%2Dfree [Accessed 24 May 2022].

www.tutorialspoint.com. (n.d.). Pushdown Automata & Parsing. [online] Available at: https://www.tutorialspoint.com/automata_theory/pda_and_parsing.htm [Accessed 24 May 2022].

Context-Free Grammars. (n.d.). [online] Available at: https://www.cs.unc.edu/~plaisted/comp455/slides/cfg3.1.pdf [Accessed 24 May 2022].