LAPORAN TUGAS BESAR BAHASA dan AUTOMATA

LEXICAL ANALYSIS dan PARSER

Oleh:

Gian Maxmillian Firdaus (1301190209)

Rahmatia Primadiati (1301194091)

Zakia Syahrini (1301194108)

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY

BANDUNG

2021

KATA PENGANTAR

Puji dan syukur marilah kita panjatkan kepada Tuhan Yang Maha Esa yang telah memberikan nikmat sehat sehingga kami dapat menyelesaikan Laporan Tugas Teori Bahasa dan Automata.

Terima kasih kami ucapkan kepada Ir.Ahmad Suryan, M.T. yang telah membantu kami melalui pemberian materi di kelas. Terima kasih juga saya ucapkan kepada teman-teman seperjuangan yang telah mendukung kami sehingga bisa menyelesaikan tugas ini tepat waktu.

Kami menyadari, bahwa laporan Tugas Teori Bahasa dan Automata yang kami buat ini masih jauh dari kata sempurna dalam penulisannya. Oleh karena itu, kami sangat mengharapkan kritik dan saran yang membangun dari semua pembaca, agar penulis menjadi lebih baik lagi di masa mendatang.

Dan semoga laporan Tugas Teori Bahasa dan Automata ini bisa menambah wawasan para pembaca dan dapat bermanfaat untuk perkembangan dan peningkatan ilmu pengetahuan.

Bandung, 11 Juni 2021

Penulis

BAB I
PENDAHULUAN

1. TEORI DASAR

1.1. Finite automata

Finite Automata adalah mesin automata dari suatu Bahasa regular. 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

Context Free Grammar atau yang dapat disingkat menjadi CFG, merupakan tata bahasa formal bebas konteks yang terdiri dari aturan tata bahasa yang terbatas. 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.

CNF dapat dibuat dari sebuah tata bahasa bebas konteks yang telah mengalami penyederhanaan yaitu penghilangan produksi useless, unit, dan ε.Dengan kata lain, suatu tata bahasa bebas konteks dapat dibuat menjadi bentuk normal Chomsky dengan syarat tata bahasa bebas konteks tersebut:

- Tidak memiliki produksi useless
- Tidak memiliki produksi unit
- Tidak memiliki produksi ε

1.3. Lexical analyzer

Lexical analyzer adalah tahapan pertama yang dilakukan pada compiler. Proses yang dilakukan pada tahapan ini adalah membaca program sumber karakter per karakter. Satu atau lebih (deretan) karakter karakter ini dikelompokkan menjadi suatu kesatuan mengikuti pola kesatuan kelompok karakter (token) yang ditentukan dalam bahasa sumber dan disimpan dalam table simbol, sedangkan karakter yang tidak mengikuti pola akan dilaporkan sebagai token tak dikenal.

1.4. Parser

Parser adalah komponen kompiler atau juru bahasa yang memecah data menjadi elemen yang lebih kecil untuk memudahkan terjemahan ke bahasa lain. Parser mengambil input dalam bentuk urutan token atau instruksi program dan biasanya membangun struktur data dalam bentuk pohon parse atau pohon sintaksis abstrak.

BAB II
CFG, FINITE AUTOMATA, dan PARSER TABLE

2. Rancangan CFG, Finite Automata, dan parser table

2.1. Context Free Grammar

CFG Menggunakan bahasa inggris yaitu terdiri dari:

S → {NN VB OB}

NN → brother | sister | you

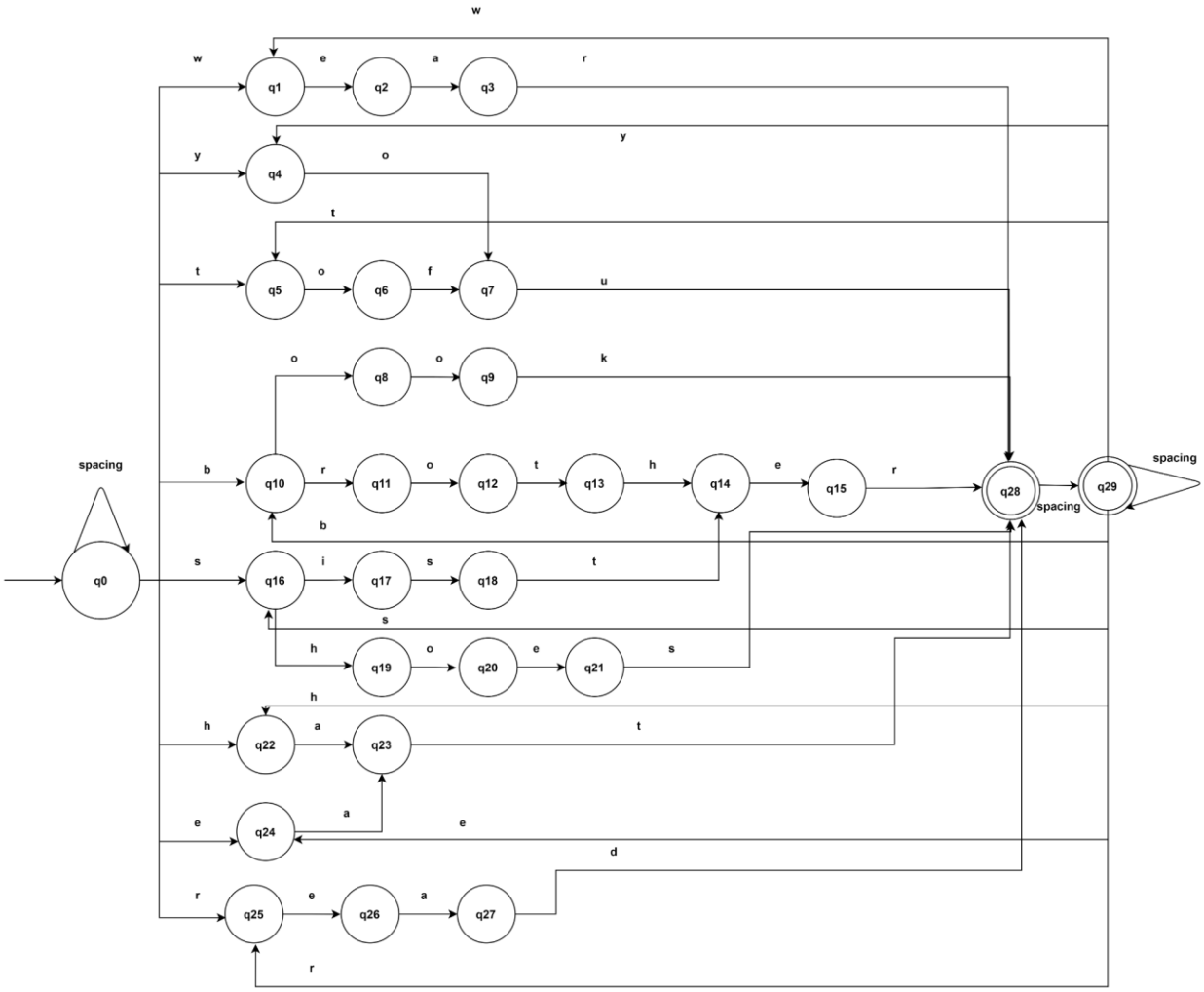
VB → read | eat | wear

OB → book | shoes | tofu | hat

non-terminal symbol : S (starting symbol), noun, verb

Terminal symbol : brother, sister, you, book, shoes, tofu, hat, read, eat, wear

2.2. Finite Automata



2.3. PARSER TABLE LL (1)

	brother	sister	you	read	eat	wear	book	shoes	tofu	hat	EOS
S	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	brother	sister	you	error	error	error	error	error	error	error	error
VB	error	error	error	read	eat	wear	error	error	error	error	error
OB	error	error	error	error	error	error	book	shoes	tofu	hat	error

BAB III
PROGRAM

3. Lexical Analyzer

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 brother, sister, you, book, shoes, tofu, hat, read, eat, wear.

C: > Users > Lenovo > Downloads > Lexical Analyzer Kelompok 1 test.py > lexical

```python
print("TUGAS BESAR BAHASA AUTOMATA | KELOMPOK 1 | IF4310")
print("========LEXICAL ANALYZER=========")

import string

#input example
#sentence = 'brother wear hat'
#input_string = sentence.lower()+'#'


def lexical(sentence):

    #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']

    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[("q28", "#")] = "accept"
    transition_table[("q28", " ")] = "q29"

    transition_table[("q29", "#")] = "accept"
    transition_table[("q29", " ")] = "q29"

    #string brother
    transition_table[("q0", "b")] = "q10"
    transition_table[("q10", "r")] = "q11"
    transition_table[("q11", "o")] = "q12"
    transition_table[("q12", "t")] = "q13"
    transition_table[("q13", "h")] = "q14"
    transition_table[("q14", "e")] = "q15"
    transition_table[("q15", "r")] = "q28"
    transition_table[("q28", " ")] = "q29"
    transition_table[("q29", "b")] = "q10"

    #string sister
    transition_table[("q0", "s")] = "q16"
    transition_table[("q16", "i")] = "q17"
    transition_table[("q17", "s")] = "q18"
    transition_table[("q18", "t")] = "q14"
    transition_table[("q14", "e")] = "q15"
    transition_table[("q15", "r")] = "q28"
    transition_table[("q28", " ")] = "q29"
    transition_table[("q29", "s")] = "q16"

```

```
    #string you
    transition_table[("q0", "y")] = "q4"
    transition_table[("q4", "o")] = "q7"
    transition_table[("q7", "u")] = "q28"
    transition_table[("q28", " ")] = "q29"
    transition_table[("q29", "y")] = "q4"

    #string read
    transition_table[("q0", "r")] = "q25"
    transition_table[("q25", "e")] = "q26"
    transition_table[("q26", "a")] = "q27"
    transition_table[("q27", "d")] = "q28"
    transition_table[("q28", " ")] = "q29"
    transition_table[("q29", "r")] = "q25"

    #string eat
    transition_table[("q0", "e")] = "q24"
    transition_table[("q24", "a")] = "q23"
    transition_table[("q23", "t")] = "q28"
    transition_table[("q28", " ")] = "q29"
    transition_table[("q29", "e")] = "q24"

    #string wear
    transition_table[("q0", "w")] = "q1"
    transition_table[("q1", "e")] = "q2"
    transition_table[("q2", "a")] = "q3"
    transition_table[("q3", "r")] = "q28"
```

```
    transition_table[("q28", " ")] = "q29"
    transition_table[("q29", "w")] = "q1"

    #string book
    transition_table[("q0", "b")] = "q10"
    transition_table[("q10", "o")] = "q8"
    transition_table[("q8", "o")] = "q9"
    transition_table[("q9", "k")] = "q28"
    transition_table[("q28", " ")] = "q29"
    transition_table[("q29", "b")] = "q10"

    #string shoes
    transition_table[("q0", "s")] = "q16"
    transition_table[("q16", "h")] = "q19"
    transition_table[("q19", "o")] = "q20"
    transition_table[("q20", "e")] = "q21"
    transition_table[("q21", "s")] = "q28"
    transition_table[("q28", " ")] = "q29"
    transition_table[("q29", "s")] = "q16"

    #string tofu
    transition_table[("q0", "t")] = "q5"
    transition_table[("q5", "o")] = "q6"
    transition_table[("q6", "f")] = "q7"
    transition_table[("q7", "u")] = "q28"
    transition_table[("q28", " ")] = "q29"
    transition_table[("q29", "t")] = "q5"

```

```
    #string hat
    transition_table[("q0", "h")] = "q22"
    transition_table[("q22", "a")] = "q23"
    transition_table[("q23", "t")] = "q28"
    transition_table[("q28", " ")] = "q29"
    transition_table[("q29", "h")] = "q22"

    #lexical analysis
    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=='q28':
            print('current token: ', current_token, ', valid')
            current_token = ''
        if state =="error":
            print("error")
            break
        idx_char = idx_char + 1

    #conclusion
    if state == "accept":
        print('semua token diinput: ', sentence, ', valid')

    return lexical

print("terminal: brother, sister, you, read, eat, wear, book, shoes, tofu, hat \n ")
sentence = input("input here: ")
input_string = sentence.lower()+'#'
lexical(sentence)
```

Pada program Lexical Analyzer, dibuat fungsi lexical yang memanggil fungsi sentence. Ketika program di-run, nanti akan meminta inputan. Jika user memasukan inputan sesuai dengan data yang ada pada dafatar simbol terminal, maka akan keluar ‘valid’. Namun ketika data yang dimasukan tidak terdaftar dalam daftar simbol terminal akan keluar ‘error’ ketika di-*run*.

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

```
PS C:\Users\Lenovo> & C:/Users/Lenovo/AppData/Local/Programs/Python/Python39/python.
exe "c:/Users/Lenovo/Downloads/Lexical Analyzer Kelompok 1 test (1).py"
TUGAS BESAR BAHASA AUTOMATA | KELOMPOK 1 | IF4310
========LEXICAL ANALYZER=========
terminal: brother, sister, you, read, eat, wear, book, shoes, tofu, hat

input here: you wear shoes
```

Pada percobaan kali ini dimulai dengan memasukan isi sentence dengan kata yang telah terdaftar pada symbol terminal. Contoh yang digunakan yaitu ‘you wear shoes’.

```
PS C:\Users\Lenovo> & C:/Users/Lenovo/AppData/Local/Programs/Python/Python39/python.
exe "c:/Users/Lenovo/Downloads/Lexical Analyzer Kelompok 1 test (1).py"
TUGAS BESAR BAHASA AUTOMATA | KELOMPOK 1 | IF4310
========LEXICAL ANALYZER=========
terminal: brother, sister, you, read, eat, wear, book, shoes, tofu, hat

input here: you wear shoes
current token:  you , valid
current token:   wear , valid
current token:   shoes , valid
semua token diinput:  you wear shoes , valid
```

Setelah di-*run* hasilnya adalah current token adalah valid, karena kata yang dimasukan pada sentence ada pada daftar simbol terminal.

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

```
PS C:\Users\Lenovo> & C:/Users/Lenovo/AppData/Local/Programs/Python/Python39/python.
exe "c:/Users/Lenovo/Downloads/Lexical Analyzer Kelompok 1 test (1).py"
TUGAS BESAR BAHASA AUTOMATA | KELOMPOK 1 | IF4310
========LEXICAL ANALYZER=========
terminal: brother, sister, you, read, eat, wear, book, shoes, tofu, hat

input here: kamu memakai jaket
```

Pada percobaan kali ini dimulai dengan memasukan isi sentence dengan kata yang tidak terdaftar pada symbol terminal. Contoh yang digunakan yaitu ‘kamu memakai jaket’.’

```
PS C:\Users\Lenovo> & C:/Users/Lenovo/AppData/Local/Programs/Python/Python39/python.
exe "c:/Users/Lenovo/Downloads/Lexical Analyzer Kelompok 1 test (1).py"
TUGAS BESAR BAHASA AUTOMATA | KELOMPOK 1 | IF4310
========LEXICAL ANALYZER=========
terminal: brother, sister, you, read, eat, wear, book, shoes, tofu, hat

input here: kamu memakai jaket
error
PS C:\Users\Lenovo>
```

Setelah di-*run* hasilnya adalah error, karena kata yang dimasukan pada sentence tidak ada pada daftar symbol terminal.

4. Parser

4.1. Code Program Parser

Disini 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 brother, sister, you, book, shoes, tofu, hat, read, eat, wear.

```
#Gian Maxmillian Firdaus (1301190209),Rahmatia Primadiati (1301194091),Zakia Syahrini (1301194108)(IF4310)
def parser(sentence):
    print("==========PARSER========== \n")
    print("=========KELOMPOK 1======= \n")

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

    non_terminals = ['S', 'NN', 'VB', 'OB']
    terminals = ['brother', 'sister', 'you', 'book', 'shoes', 'tofu', 'hat', 'read', 'eat', 'wear']

    parse_table = {}

    parse_table[('S', 'brother')] = ['NN', 'VB', 'OB']
    parse_table[('S', 'sister')] = ['NN', 'VB', 'OB']
    parse_table[('S', 'you')] = ['NN', 'VB', 'OB']
    parse_table[('S', 'read')] = ['error']
    parse_table[('S', 'eat')] = ['error']
    parse_table[('S', 'wear')] = ['error']
    parse_table[('S', 'book')] = ['NN', 'VB', 'OB']
    parse_table[('S', 'shoes')] = ['NN', 'VB', 'OB']
    parse_table[('S', 'tofu')] = ['NN', 'VB', 'OB']
    parse_table[('S', 'hat')] = ['NN', 'VB', 'OB']
    parse_table[('S', 'EOS')] = ['error']
```

```
166     parse_table[('NN', 'brother')] = ['brother']
167     parse_table[('NN', 'sister')] = ['sister']
168     parse_table[('NN', 'you')] = ['you']
169     parse_table[('NN', 'read')] = ['error']
170     parse_table[('NN', 'eat')] = ['error']
171     parse_table[('NN', 'wear')] = ['error']
172     parse_table[('NN', 'book')] = ['error']
173     parse_table[('NN', 'shoes')] = ['error']
174     parse_table[('NN', 'tofu')] = ['error']
175     parse_table[('NN', 'hat')] = ['error']
176     parse_table[('NN', 'EOS')] = ['error']
177
178     parse_table[('VB', 'brother')] = ['error']
179     parse_table[('VB', 'sister')] = ['error']
180     parse_table[('VB', 'you')] = ['error']
181     parse_table[('VB', 'read')] = ['read']
182     parse_table[('VB', 'eat')] = ['eat']
183     parse_table[('VB', 'wear')] = ['wear']
184     parse_table[('VB', 'book')] = ['error']
185     parse_table[('VB', 'shoes')] = ['error']
186     parse_table[('VB', 'tofu')] = ['error']
187     parse_table[('VB', 'hat')] = ['error']
188     parse_table[('VB', 'EOS')] = ['error']
```

```
    parse_table[('OB', 'brother')] = ['error']
    parse_table[('OB', 'sister')] = ['error']
    parse_table[('OB', 'you')] = ['error']
    parse_table[('OB', 'read')] = ['read']
    parse_table[('OB', 'eat')] = ['eat']
    parse_table[('OB', 'wear')] = ['wear']
    parse_table[('OB', 'book')] = ['book']
    parse_table[('OB', 'shoes')] = ['shoes']
    parse_table[('OB', 'tofu')] = ['tofu']
    parse_table[('OB', 'hat')] = ['hat']
    parse_table[('OB', 'EOS')] = ['error']


    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])
            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')

    return parser

print("terminal: brother, sister, you, read, eat, wear, book, shoes, tofu, hat \n ")
sentence = input("input masukan: ")
input_string = sentence.lower()+'#'
lexical(sentence)
parser(sentence)
```

Berikut adalah hasil pengujian program gabungan dari *lexical analysis* dan *parser* yang Sudah kami buat. Terdapat tiga kalimat input yang akan diuji apakah diterima atau tidak yang ditentukan dengan grammar.

4.2. Pengujian Program Lexical Analyzer dan Parser dengan Kata yang Sesuai Grammar

4.2.1. brother wear hat

```
TUGAS BESAR BAHASA AUTOMATA | KELOMPOK 1 | IF4310
========LEXICAL ANALYZER=========
terminal: brother, sister, you, read, eat, wear, book, shoes,
tofu, hat

input masukan: brother wear hat
current token:  brother , valid
current token:   wear , valid
current token:   hat , valid
semua token diinput:  brother wear hat , valid
==========PARSER==========

=========KELOMPOK 1=======

top =  S
symbol =  brother
top adalah symbol non-terminal
isi stack:  ['#', 'OB', 'VB', 'NN']

top =  NN
symbol =  brother
top adalah symbol non-terminal
isi stack:  ['#', 'OB', 'VB', 'brother']

top =  brother
symbol =  brother
top adalah simbol terminal
isi stack:  ['#', 'OB', 'VB']
```

```
top =  VB
symbol =  wear
top adalah symbol non-terminal
isi stack:  ['#', 'OB', 'wear']

top =  wear
symbol =  wear
top adalah simbol terminal
isi stack:  ['#', 'OB']

top =  OB
symbol =  hat
top adalah symbol non-terminal
isi stack:  ['#', 'hat']

top =  hat
symbol =  hat
top adalah simbol terminal
isi stack: []
isi stack:  []


input string  " brother wear hat " diterima, sesuai grammar
```

4.2.2. sister read book

```
TUGAS BESAR BAHASA AUTOMATA | KELOMPOK 1 | IF4310
========LEXICAL ANALYZER=========
terminal: brother, sister, you, read, eat, wear, book, shoes, tofu, hat

input masukan: sister read book
current token:  sister , valid
current token:   read , valid
current token:   book , valid
semua token diinput:  sister read book , valid
==========PARSER==========

=========KELOMPOK 1=======

top =  S
symbol =  sister
top adalah symbol non-terminal
isi stack:  ['#', 'OB', 'VB', 'NN']

top =  NN
symbol =  sister
top adalah symbol non-terminal
isi stack:  ['#', 'OB', 'VB', 'sister']

top =  sister
symbol =  sister
top adalah simbol terminal
isi stack:  ['#', 'OB', 'VB']
```

```
top =  VB
symbol =  read
top adalah symbol non-terminal
isi stack:  ['#', 'OB', 'read']

top =  read
symbol =  read
top adalah simbol terminal
isi stack:  ['#', 'OB']

top =  OB
symbol =  book
top adalah symbol non-terminal
isi stack:  ['#', 'book']

top =  book
symbol =  book
top adalah simbol terminal
isi stack: []
isi stack:  []


input string  " sister read book " diterima, sesuai grammar
PS C:\Users\Lenovo>
```

4.2.3. you read book

```
TUGAS BESAR BAHASA AUTOMATA | KELOMPOK 1 | IF4310
========LEXICAL ANALYZER=========
terminal: brother, sister, you, read, eat, wear, book, shoes, t
ofu, hat

input masukan: you read book
current token:  you , valid
current token:   read , valid
current token:   book , valid
semua token diinput:  you read book , valid
==========PARSER==========

=========KELOMPOK 1=======

top =  S
symbol =  you
top adalah symbol non-terminal
isi stack:  ['#', 'OB', 'VB', 'NN']

top =  NN
symbol =  you
top adalah symbol non-terminal
isi stack:  ['#', 'OB', 'VB', 'you']

top =  you
symbol =  you
top adalah simbol terminal
isi stack:  ['#', 'OB', 'VB']
```

```
top =  VB
symbol =  read
top adalah symbol non-terminal
isi stack:  ['#', 'OB', 'read']

top =  read
symbol =  read
top adalah simbol terminal
isi stack:  ['#', 'OB']

top =  OB
symbol =  book
top adalah symbol non-terminal
isi stack:  ['#', 'book']

top =  book
symbol =  book
top adalah simbol terminal
isi stack: []
isi stack:  []


input string  " you read book " diterima, sesuai grammar
PS C:\Users\Lenovo>
```

Dari hasil pengujian program gabungan untuk lexical analysis dan parser yang digunakan untuk menguji kalimat input yang telah ditentukan. Tiga input kalimat yang telah sesuai dengan grammar ketika diuji maka input string tersebut diterima.

4.3. Pengujian Program Lexical Analyzer dan Parser dengan Kata yang tidak Sesuai Grammar

4.3.1. Tofu eat sister

```
TUGAS BESAR BAHASA AUTOMATA | KELOMPOK 1 | IF4310
========LEXICAL ANALYZER=========
terminal: brother, sister, you, read, eat, wear, book, shoes, tofu, hat

input masukan: tofu eat sister
current token:  tofu , valid
current token:   eat , valid
current token:   sister , valid
semua token diinput:  tofu eat sister , valid
==========PARSER==========

=========KELOMPOK 1=======

top =  S
symbol =  tofu
top adalah symbol non-terminal
isi stack:  ['#', 'OB', 'VB', 'NN']

top =  NN
symbol =  tofu
top adalah symbol non-terminal
error

error, input string: " tofu eat sister " , tidak diterima, tidak sesuai grammar
```

4.3.2. Book wear you

```
TUGAS BESAR BAHASA AUTOMATA | KELOMPOK 1 | IF4310
========LEXICAL ANALYZER=========
terminal: brother, sister, you, read, eat, wear, book, shoes, tofu, hat

input masukan: book wear you
current token:  book , valid
current token:   wear , valid
current token:   you , valid
semua token diinput:  book wear you , valid
==========PARSER==========

=========KELOMPOK 1=======

top =  S
symbol =  book
top adalah symbol non-terminal
isi stack:  ['#', 'OB', 'VB', 'NN']

top =  NN
symbol =  book
top adalah symbol non-terminal
error

error, input string: " book wear you " , tidak diterima, tidak sesuai grammar
```

4.3.3. sister read book wear book

```
TUGAS BESAR BAHASA AUTOMATA | KELOMPOK 1 | IF4310
========LEXICAL ANALYZER=========
terminal: brother, sister, you, read, eat, wear, book, shoes, tofu, hat

input masukan: sister wear book read book
current token:  sister , valid
current token:   wear , valid
current token:   book , valid
current token:   read , valid
current token:   book , valid
semua token diinput:  sister wear book read book , valid
==========PARSER==========

=========KELOMPOK 1=======

top =  S
symbol =  sister
top adalah symbol non-terminal
isi stack:  ['#', 'OB', 'VB', 'NN']

top =  NN
symbol =  sister
top adalah symbol non-terminal
isi stack:  ['#', 'OB', 'VB', 'sister']

top =  sister
symbol =  sister
top adalah simbol terminal
isi stack:  ['#', 'OB', 'VB']
```

```
top =  VB
symbol =  read
top adalah symbol non-terminal
isi stack:  ['#', 'OB', 'read']

top =  read
symbol =  read
top adalah simbol terminal
isi stack:  ['#', 'OB']

top =  OB
symbol =  book
top adalah symbol non-terminal
isi stack:  ['#', 'book']

top =  book
symbol =  book
top adalah simbol terminal
isi stack:  ['#']

top =  #
symbol =  wear
error

error, input string: " sister read book wear book " , tidak diterima, tidak sesuai grammar
```

Dari hasil pengujian diatas, dika input yang dimasukan tidak sesuai dengan grammar yang telah ditentukan (seperti susunan struktur yang terbaik), maka hasil outputnya adalah error atau tidak diterima.

BAB IV
PENUTUP

4.1. KESIMPULAN

Parse adalah proses untuk data atau informasi berarti memecahnya menjadi bagian-bagian komponen sehingga sintaksnya dapat dianalisis, dikategorikan dan dipahami.Tujuan parser adalah untuk melibatkan pencarian pohon parse 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.

4.2. CARA MENJALAN PROGRAM LEXICAL ANALYZER dan PARSER

1. Buka program lexical analyzer atau parser yang sudah kelompok kami buat dengan aplikasi seperti visual studio code, dll.
   (program kami juga bisa langsung input masukan tanpa harus membuka program terlebih dahulu hanya saja, setelah di run dan mengeluarkan output akan otomatis ter-close)
2. Run program
3. Inputkan masukan dengan kata yang sudah kami sediakan di terminal
4. Lalu tekan enter
5. Akan muncul hasil valid atau tidak validnya kata yang user inputkan.

Daftar Pustaka

https://socs.binus.ac.id/2019/12/21/teknik-kompilasi-perbedaan-dfa-dan-nfa/

https://id.icyscience.com/parser