

**Tugas Besar Teori Bahasa dan Automata:  
Lexical Analyzer dan Parser Sederhana  
untuk Teks Bahasa Alami dalam Bahasa Melayu**



Oleh:  
Risma Amaliyah Mahmudah (1301204087)  
Vania Amadea (1301204365)  
IF4408

Program Studi S1 Informatika  
Fakultas Informatika  
Universitas Telkom  
Bandung

## Daftar Isi

<b>BAB 1</b>	<b>3</b>
<b>BAB 2</b>	<b>4</b>
2.1 Finite Automata	4
2.2 Context Free Grammar	4
2.3 Lexical Analysis	4
2.4 Parser	4
<b>BAB 3</b>	<b>5</b>
<b>BAB 4</b>	<b>7</b>
<b>References</b>	<b>8</b>

## **BAB 1**

### **PENDAHULUAN**

Tata Bahasa Bebas Konteks (*Context Free Grammar* / CFG) adalah sebuah cara membuat string dalam suatu bahasa. Pada saat menurunkan suatu string, simbol-simbol variabel akan mewakili bagian-bagian yang belum yang belum diturunkan dari string tersebut. Bila pada tata bahasa regular, bagian yang belum terturunkan tersebut selalu terjadi pada suatu ujung. *Context Free Grammar* memungkinkan terdapat lebih banyak bagian yang belum diturunkan dan bisa terjadi di mana saja. Jika penurunan sudah lengkap, semua bagian yang belum diturunkan telah diganti oleh string-string yang mungkin saja kosong dari himpunan simbol terminal. Hal ini menjadi dasar dalam pembentukan suatu parser. *Context Free Grammar* sederhana yang dibuat ini menggunakan representasi aturan atau sintaks kalimat dalam sebuah bahasa, bahasa Melayu.

## **BAB 2**

### **KAJIAN PUSTAKA**

#### **2.1 Finite Automata**

Finite automata adalah mesin abstrak berupa sistem model matematika dengan masukan dan keluaran diskrit yang dapat mengenali bahasa paling sederhana (bahasa reguler) dan dapat diimplementasikan secara nyata di mana sistem dapat berada di salah satu dari sejumlah berhingga konfigurasi internal disebut state.

#### **2.2 Context Free Grammar**

CFG atau Context Free Grammar adalah tata bahasa formal di mana setiap aturan produksi adalah dalam bentuk  $A \rightarrow B$  di mana A adalah pemproduksi, dan B adalah hasil produksi. Batasannya hanyalah ruas kiri adalah sebuah simbol variabel. Dan pada ruas kanan bisa berupa terminal, symbol, variable ataupun  $\epsilon$ .

#### **2.3 Lexical Analysis**

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.

#### **2.4 Parser**

Parsing adalah suatu cara memecah-mecah suatu rangkaian masukan (misalnya dari berkas atau keyboard) yang akan menghasilkan suatu pohon uraian (parse tree) yang akan digunakan pada tahap kompilasi berikutnya yaitu analisis semantik. Di dalam komputasi, parser adalah salah satu komponen dalam sebuah interpreter atau compiler yang bertugas memeriksa sintaks secara benar serta membangun struktur data yang tersirat dalam token masukan.

## BAB 3

### ANALISIS DAN PERANCANGAN

#### 3.1 Context Free Grammar

Deskripsi CFG untuk Bahasa Melayu

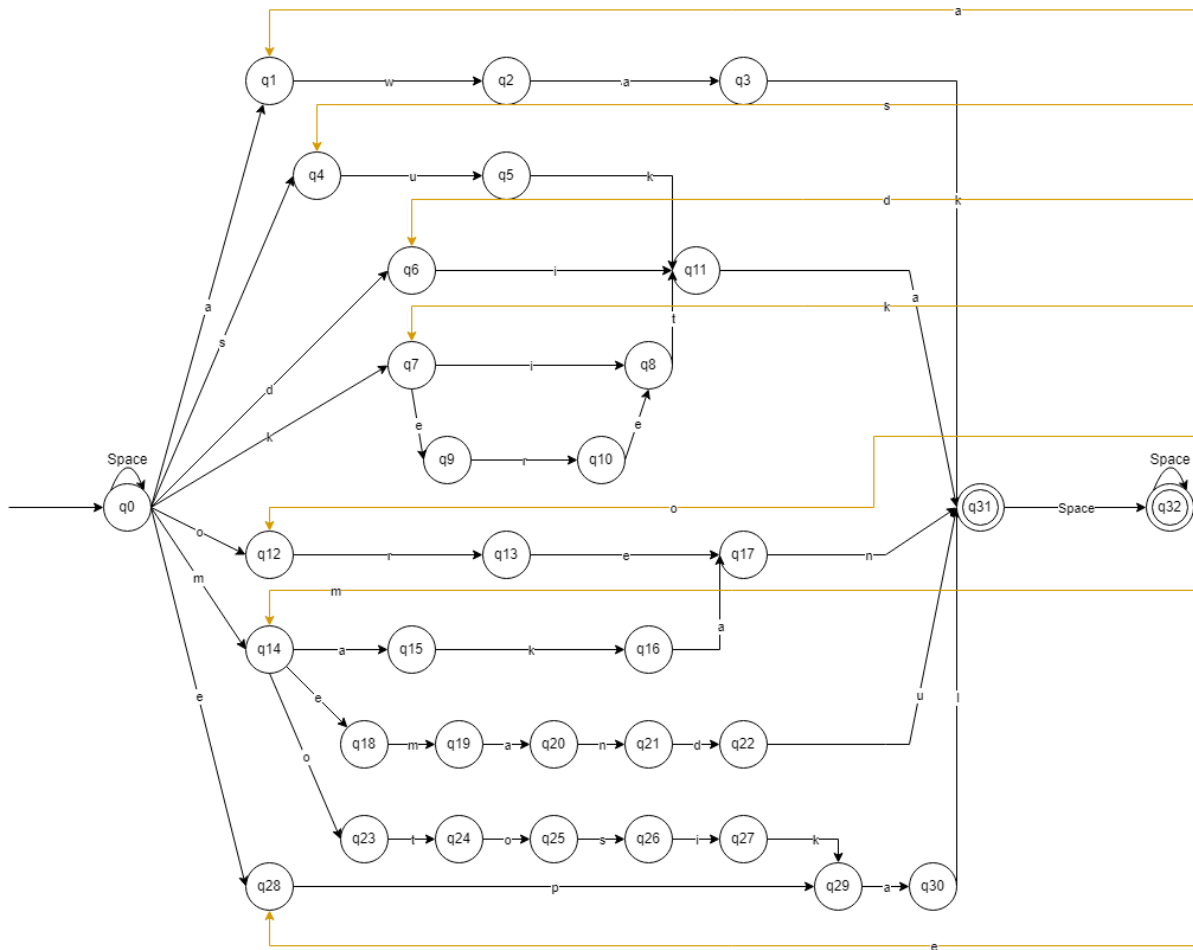
$S = \{SB, VB, OB\}$

$SB \rightarrow awak \mid dia \mid kita$

$VB \rightarrow makan \mid memandu \mid suka$

$OB \rightarrow epal \mid kereta \mid motosikal \mid oren$

#### 3.2 Finite Automata



### 3.3 Parser Table LL (1)

	awak	dia	kita	makan	memandu	suka	epal	kereta	motosikal	oren	EOS
S	SB	SB	SB	error	error	error	SB	SB	SB	SB	error
	VB	VB	VB				VB	VB	VB	VB	
	OB	OB	OB				OB	OB	OB	OB	
SB	awak	dia	kita	error	error	error	error	error	error	oren	error
VB	error	error	error	makan	memandu	suka	error	error	error	oren	error
OB	error	error	error	error	error	error	epal	kereta	motosikal	oren	error

## BAB 4

### KODE PROGRAM DAN HASIL

#### 4.1 Lexical Analysis

Berikut adalah kodingan untuk *lexical analyzer* untuk menguji kata yang dimasukkan user. Adapun kata yang valid dalam *lexical analyzer* ini, yaitu awak, dia, kita, makan, memandu, suka, epal, kereta, motosikal, dan oren. Untuk kata awak, dia, dan kita merupakan kata *subject*. Untuk kata makan, memandu, dan suka merupakan kata kerja (*verb*). Untuk kata epal, kereta, motosikal, dan oren merupakan kata *object*.

```
JS lexical_analyzerjs > ...
1 var masukkan = document.getElementById('masukkan_kalimat');
2 var submit = document.getElementById('btn-analyze');
3 var hasil = document.getElementById('result');
4 var clear = document.getElementById('btn-clear');
5 var loading = document.getElementById('loading');
6
7 //input example
8 //var input_string;
9 //var sentence;
10 //sentence = 'awak memandu motosikal';
11 //input_string = sentence.lower () + '#';
12
13 //initialization
14 var state_list, transition_table;
15 var alphabet_list = [];
16 for(var i = 32; i <= 126; i++) {
17     alphabet_list.push(String.fromCharCode( i ))
18 }
19 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",
20 transition_table = {};
21
22
23 for(var state in state_list) {
24     for(alphabet in alphabet_list) {
25         transition_table[[state_list[state], alphabet_list[alphabet]]] = 'error'
26     }
27     transition_table[[state_list[state], '#']] = 'error'
28     transition_table[[state_list[state], ' ']] = 'error'
29 }
30
31 //spaces before input string
32 transition_table [['q0', ' ']] = 'q0'
33 //final state
34 transition_table[['q31', ' ']] = 'q32'
35 transition_table[['q31', '#']] = 'accept'
36
37 transition_table[['q32', ' ']] = 'q32'
```

```
JS lexical_analyzerjs > ...
38 transition_table[['q32', '#']] = 'accept'
39
40 //update the transition table for the following token: awak
41 transition_table[['q0', 'a']] = 'q1'
42 transition_table[['q1', 'w']] = 'q2'
43 transition_table[['q2', 'a']] = 'q3'
44 transition_table[['q3', 'k']] = 'q31'
45 transition_table[['q31', ' ']] = 'q32'
46
47 //update the transition table for the following token: suka
48 transition_table[['q0', 's']] = 'q4'
49 transition_table[['q4', 'u']] = 'q5'
50 transition_table[['q5', 'k']] = 'q11'
51 transition_table[['q11', 'a']] = 'q31'
52 transition_table[['q31', ' ']] = 'q32'
53
54 //update the transition table for the following token: dia
55 transition_table[['q0', 'd']] = 'q6'
56 transition_table[['q6', 'i']] = 'q11'
57 transition_table[['q11', 'a']] = 'q31'
58 transition_table[['q31', ' ']] = 'q32'
59
60 //update the transition table for the following token: kita
61 transition_table[['q0', 'k']] = 'q7'
62 transition_table[['q7', 'i']] = 'q8'
63 transition_table[['q8', 't']] = 'q11'
64 transition_table[['q11', 'a']] = 'q31'
65 transition_table[['q31', ' ']] = 'q32'
66
67 //update the transition table for the following token: kereta
68 transition_table[['q0', 'k']] = 'q7'
69 transition_table[['q7', 'e']] = 'q9'
70 transition_table[['q9', 'r']] = 'q10'
71 transition_table[['q10', 'e']] = 'q8'
72 transition_table[['q8', 't']] = 'q11'
73 transition_table[['q11', 'a']] = 'q31'
74 transition_table[['q31', ' ']] = 'q32'
```

```

JS lexical_analyzer.js > ...
75
76 //update the transition table for the following token: oren
77 transition_table[['q0', 'o']] = 'q12'
78 transition_table[['q12', 'r']] = 'q13'
79 transition_table[['q13', 'e']] = 'q17'
80 transition_table[['q17', 'n']] = 'q31'
81 transition_table[['q31', ' ']] = 'q32'
82
83 //update the transition table for the following token: makan
84 transition_table[['q0', 'm']] = 'q14'
85 transition_table[['q14', 'a']] = 'q15'
86 transition_table[['q15', 'k']] = 'q16'
87 transition_table[['q16', 'a']] = 'q17'
88 transition_table[['q17', 'n']] = 'q31'
89 transition_table[['q31', ' ']] = 'q32'
90
91 //update the transition table for the following token: memandu
92 transition_table[['q0', 'm']] = 'q14'
93 transition_table[['q14', 'e']] = 'q18'
94 transition_table[['q18', 'm']] = 'q19'
95 transition_table[['q19', 'a']] = 'q20'
96 transition_table[['q20', 'n']] = 'q21'
97 transition_table[['q21', 'd']] = 'q22'
98 transition_table[['q22', 'u']] = 'q31'
99 transition_table[['q31', ' ']] = 'q32'
100
101 //update the transition table for the following token: motosikal
102 transition_table[['q0', 'm']] = 'q14'
103 transition_table[['q14', 'o']] = 'q23'
104 transition_table[['q23', 't']] = 'q24'
105 transition_table[['q24', 'o']] = 'q25'
106 transition_table[['q25', 's']] = 'q26'
107 transition_table[['q26', 'i']] = 'q27'
108 transition_table[['q27', 'k']] = 'q29'
109 transition_table[['q29', 'a']] = 'q30'
110 transition_table[['q30', 'l']] = 'q31'
111 transition_table[['q31', ' ']] = 'q32'

```

```

JS lexical_analyzer.js > ...
112
113 //update the transition table for the following token: epal
114 transition_table[['q0', 'e']] = 'q28'
115 transition_table[['q28', 'p']] = 'q29'
116 transition_table[['q29', 'a']] = 'q30'
117 transition_table[['q30', 'l']] = 'q31'
118 transition_table[['q31', ' ']] = 'q32'
119
120 //update the transition table for the new token
121 transition_table[['q32', 'a']] = 'q1'
122 transition_table[['q32', 's']] = 'q4'
123 transition_table[['q32', 'd']] = 'q6'
124 transition_table[['q32', 'k']] = 'q7'
125 transition_table[['q32', 'o']] = 'q12'
126 transition_table[['q32', 'm']] = 'q14'
127 transition_table[['q32', 'e']] = 'q28'
128
129 clear.onclick = (event) => {
130     masukan.value = "";
131     hasil.value = "";
132 }
133
134 submit.onclick = (event) => {
135
136     loading.style = 'display: inline-block'
137
138     // lexical analysis
139     var indexChar = 0;
140     var state = 'q0';
141     var currentToken = '';
142     var validation = '';
143     var inputChar = masukan.value + '#';
144     console.log(inputChar);
145     while (state != 'accept') {
146         var currentChar = inputChar.charAt(indexChar)
147         currentToken += currentChar
148         state = transition_table[[state, currentChar]]

```

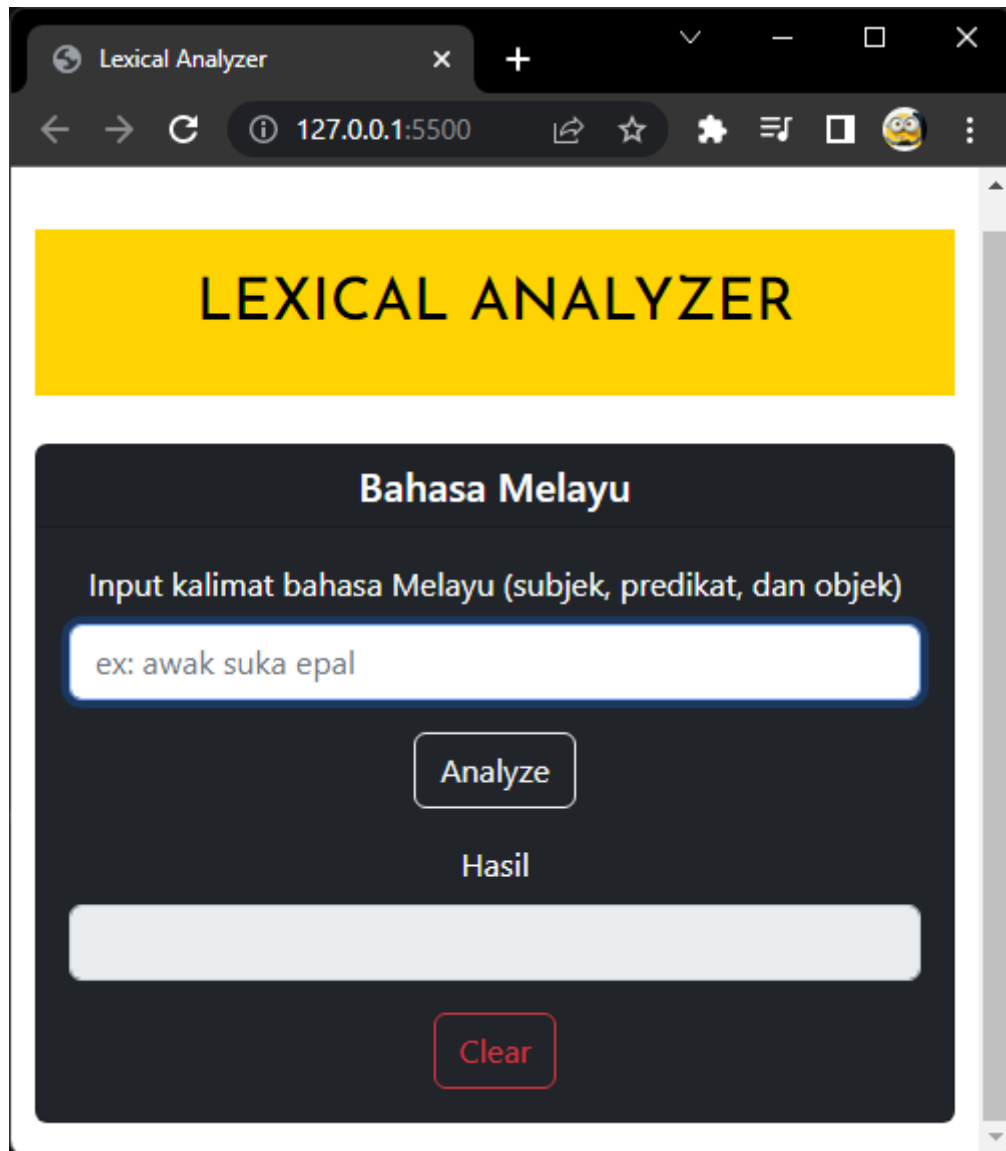
```

JS lexical_analyzer.js > ...
149         if(state == 'q31') {
150             console.log("valid")
151             validation += "valid "
152             currentToken = ''
153         }
154         if(state == 'error') {
155             console.log("error")
156             validation += "error "
157             break;
158         }
159         indexChar += 1
160     }
161
162     console.log(validation);
163     hasil.value = validation.trim();
164
165     loading.style = 'display: none'
166 }

```



Berikut adalah hasil pengujian untuk *lexical analyzer* yang digunakan untuk menguji kata yang dimasukkan oleh user.



The screenshot shows a web browser window with the title "Lexical Analyzer". The address bar displays "127.0.0.1:5500". The main content area has a yellow header with the text "LEXICAL ANALYZER". Below this, there is a dark gray box titled "Bahasa Melayu". Inside this box, the text "Input kalimat bahasa Melayu (subjek, predikat, dan objek)" is followed by a text input field containing "ex: awak suka epal". Below the input field is a button labeled "Analyze". Underneath the button is the word "Hasil" above a large, empty light gray rectangular box. At the bottom of the dark gray box is a button labeled "Clear".

Berikut merupakan hasil pengujian untuk kalimat 'awak suka epal' yang ketiga kata tersebut merupakan kata yang valid.

Lexical Analyzer

127.0.0.1:5500

# LEXICAL ANALYZER

## Bahasa Melayu

Input kalimat bahasa Melayu (subjek, predikat, dan objek)

awak suka epal

Analyze

Hasil

valid valid valid

Clear

Berikut merupakan hasil pengujian untuk kalimat 'awak minum susu' dengan kata 'awak' tersebut merupakan kata yang valid, sedangkan kata 'minum' dan 'susu' merupakan kata yang tidak valid.

Lexical Analyzer

127.0.0.1:5500/index.h...

# LEXICAL ANALYZER

## Bahasa Melayu

Input kalimat bahasa Melayu (subjek, predikat, dan objek)

awak minum susu

Analyze

Hasil

valid error

Clear

Berikut merupakan hasil pengujian untuk kalimat 'risma awak oren' dengan kata 'risma' tersebut merupakan kata yang tidak valid, maka program akan langsung menyatakan bahwa kalimat tersebut error.

Lexical Analyzer

127.0.0.1:5500/index.h...

# LEXICAL ANALYZER

## Bahasa Melayu

Input kalimat bahasa Melayu (subjek, predikat, dan objek)

risma awak oren

Analyze

Hasil

error

Clear

## References

- (n.d.). TEORI BAHASA DAN AUTOMATA. Retrieved June 8, 2022, from [https://repository.dinus.ac.id/docs/ajar/bab%2010\\_file\\_2013-05-31\\_080824\\_dr.r\\_heru\\_tjahjana\\_s.si\\_m.si\\_.pdf](https://repository.dinus.ac.id/docs/ajar/bab%2010_file_2013-05-31_080824_dr.r_heru_tjahjana_s.si_m.si_.pdf)
- BAB 2 LANDASAN TEORI 2.1 Finite Automata Finite automata adalah mesin abstrak berupa sistem model matematika dengan masukan dan.* (n.d.). Library Binus. Retrieved June 7, 2022, from <http://library.binus.ac.id/eColls/eThesisdok/Bab2/2011-2-00004-MTIF%20Bab2001.pdf>
- PENYEDERHANAAN CONTEXT FREE GRAMMAR.* (2018, December 20). School of Computer Science | BINUS University. Retrieved June 8, 2022, from <https://socs.binus.ac.id/2018/12/20/penyederhanaan-context-free-grammar/>
- Teknik Kompilasi : TAHAPAN KOMPILASI.* (2019, December 23). School of Computer Science | BINUS University. Retrieved June 8, 2022, from <https://socs.binus.ac.id/2019/12/23/teknik-kompilasi-first-set-pada-top-down-parsing/>
- Yulianto, Alfiah, F., Wijaya, A. N., Ramadhan, M. R., Sakti, L. K., Muhtasir, & Mukti, A. (2015, February 8). *IMPLEMENTASI PENGGUNAAN SISTEM APLIKASI WEB PDF PARSER UNTUK MENAMPILKAN INFORMASI ISI DOKUMEN.* AMIKOM OJS Journal. Retrieved June 8, 2022, from <https://ojs.amikom.ac.id/index.php/semnasteknomedia/article/viewFile/806/772>