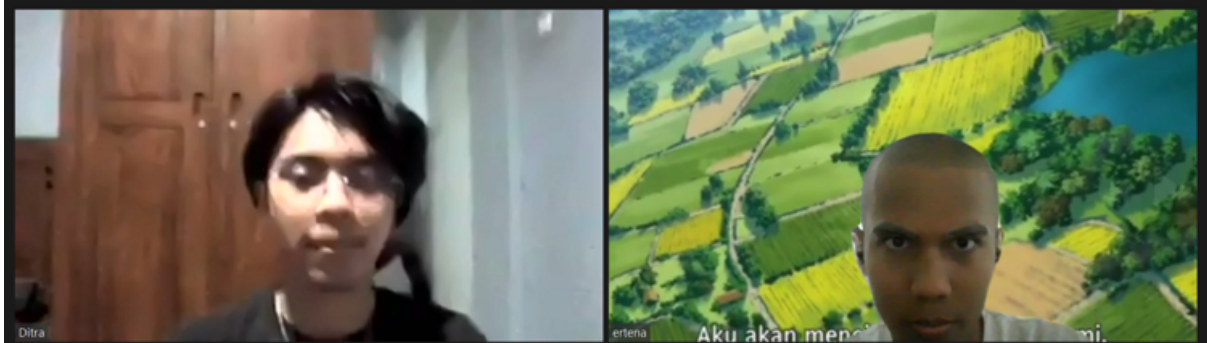


**LAPORAN TUGAS BESAR IF2124 2022**  
**APLIKASI FINITE AUTOMATA DAN CONTEXT-FREE GRAMMAR**  
**DALAM SINTAKS BAHASA JAVASCRIPT (NODE.JS)**

**Disusun untuk memenuhi tugas besar mata kuliah IF2124 Teori Bahasa Formal dan**  
**Otomata pada Semester 1 Tahun Akademik 2022/2023**

**Disusun oleh:**

|                                |                 |
|--------------------------------|-----------------|
| <b>Bagus Lathif Firmansyah</b> | <b>13521017</b> |
| <b>Ditra Rizqa Amadia</b>      | <b>13521019</b> |
| <b>Varraz Hazzandra Abrar</b>  | <b>13521020</b> |

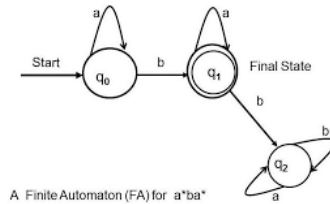


**PROGRAM STUDI TEKNIK INFORMATIKA**  
**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**  
**INSTITUT TEKNOLOGI BANDUNG**  
**2022**

# BAB 1

## TEORI DASAR

### 1.1 Finite Automata (FA)



Gambar 1.1.1 Finite Automata

Finite automata (FA) adalah kumpulan state dengan aturan transisi dari satu state menuju state lain atau state itu sendiri (*loop*). FA memiliki satu *start state* dan satu atau lebih *final state*. FA berguna untuk memodelkan kerja suatu mesin/otomata.

### 1.2 Context-Free Grammar (CFG)

$$\begin{aligned} X &\rightarrow bY \mid Za \\ Y &\rightarrow aY \mid b \\ Z &\rightarrow bZ \mid \epsilon \end{aligned}$$

Gambar 1.2.1 Contoh Context Free Grammar

Context free grammar (CFG) adalah tata bahasa formal yang setiap aturan produksinya dalam bentuk  $A \rightarrow B$ . A adalah produser dan B adalah produknya. Ruas kiri berupa variabel saja. Ruas kanan bisa berupa terminal, simbol, variabel ataupun epsilon ( $\epsilon$ ). CFG mendeskripsikan sintaks dari setiap bahasa pemrograman modern secara mendasar.

### 1.4 Chomsky's Normal Form (CNF)

$$\begin{aligned} A &\rightarrow BC \\ A &\rightarrow b \\ B &\rightarrow a \\ C &\rightarrow BA \mid d \end{aligned}$$

Gambar 1.4.1 Chomsky's Normal Form

Suatu CFG dapat dikatakan sebagai Chomsky's Normal Form (CNF) jika CFG tersebut tidak mengandung produksi *useless*, produksi unit, dan produksi *empty* ( $\epsilon$ ). Pada CNF, ruas kanan hanya boleh berupa sebuah terminal atau 2 buah simbol variabel.

### 1.5 Algoritma Cocke–Younger–Kasami (CYK)

Algoritma Cocke–Younger–Kasami (CYK) adalah algoritma parsing dan keanggotaan yang menentukan apakah suatu string bisa diperoleh dari suatu tata bahasa. Syarat penggunaan algoritma ini adalah tata bahasa harus dalam bentuk CNF.

### 1.6 Syntax JavaScript

JavaScript adalah salah satu bahasa pemrograman yang menggunakan titik koma pada akhir setiap kalimatnya walaupun pada versi barunya, titik koma sudah tidak diwajibkan lagi.

Sintaks deklarasi diawali dengan `let`, `var`, atau `const` diikuti variabel statis diikuti operator *assign* seperti sama dengan diikuti statis yang akan disimpan nilainya. Sintaks pada definisi sama seperti deklarasi namun tidak membutuhkan `let`, `var`, atau `const`.

Sintaks pada deklarasi fungsi bermacam-macam. Terdapat 3 bentuk fungsi dalam JavaScript. Fungsi dalam bentuk reguler menggunakan sintaks `function` diikuti nama fungsi, parameter, lalu blok kode. Fungsi dalam bentuk ekspresi menggunakan sintaks deklarasi diikuti sintaks fungsi reguler tanpa `function`. Fungsi dalam bentuk arrow harus dimasukkan ke dalam deklarasi diikuti parameternya, tanda panah, dan blok kode.

Sintaks kondisional `if` mengikuti aturan `if` diikuti ekspresi, blok kode, dan boleh dilanjutkan dengan sintaks `else` atau `else if`. Sintaks kondisional `switch` menggunakan aturan sintaks `switch` diikuti variabel yang akan diuji, dan diakhiri dengan blok kode yang harus mengandung `case` kondisi.

## **BAB 2**

### **HASIL CFG DAN CNF**

#### **3.1 Hasil CFG**

Hasil CFG yang telah dibuat untuk memeriksa sintaks kode javaScript sebanyak 246 *grammar*. Variabel pada *grammar* CFG dibagi menjadi dua yaitu terminal dan non terminal.

##### **3.1.1 Terminal**

Variabel terminal adalah variabel yang tidak memiliki cabang. Variabel ini direpresentasikan dengan huruf non-kapital. Berikut variabel terminal yang kami buat:

if, else, for, while, in, switch, case, break, default, continue, function, return, delete, from, import, as, try, catch, throw, finally, arrow, equal, iseq, siseq, neq, sneq, l, g, le, ge, colon, semicolon, increment, decrement, sumeq, subtreq, muleq, diveq, add, subtr, mul, div, mod, comma, dot, dotbetween, and, or, not, true, false, lb, rb, lsb, rsb, lcb, rcb, int, string, null, undefined, type, var.

##### **3.1.2 Non-Terminal**

Variabel non-terminal adalah variabel yang memiliki cabang dan dapat membuat untaian-untaian dari kumpulan variabel non-terminal lainnya dan variabel terminal. Variabel ini pada akhirnya akan menjalar membentuk untaian variabel terminal. Berikut variabel non-terminal yang kami buat:

IF, ELSE, FOR, WHILE, IN, SWITCH, CASE, BREAK, DEFAULT, CONTINUE, FUNCTION, RETURN, DELETE, FROM, IMPORT, AS, TRY, CATCH, THROW, FINALLY, ARROW, EQUAL, ISEQ, SISEQ, NEQ, SNEQ, L, G, LE, GE, COLON, SEMICOLON, INCREMENT, DECREMENT, SUMEQ, SUBTREQ, MULEQ, DIVEQ, ADD, SUBTR, MUL, DIV, MOD, COMMA, DOT, DOTBETWEEN, AND, OR, NOT, TRUE, FALSE, LB, RB, LSB, RSB, LCB, RCB, INT, STRING, NULL, UNDEFINED, TYPE, VAR, S, SS, PART, SENTENCE, SINGLE SENTENCE, RETURN SENTENCE, THROW SENTENCE, IMPORT STATEMENT, ASSIGN OPERATOR, ASSIGNMENT, ASSIGN\_STATEMENT, DELETE SENTENCE, ARIT\_OPERATOR, ARIT\_OPERAND, ARIT\_OPERATION, LOGIC\_OPERATOR, LOGIC\_OPERAND, LOGIC\_OPERATION, IF\_STATEMENT, SWITCH\_STATEMENT, SWITCH\_BLOCK, SWITCH\_SENTENCE, CASE\_STATEMENT, FUNCTION\_STATEMENT, FUNCTION\_STATEMENT\_EXPR, FUNCTION\_STATEMENT\_ARROW, PARAM, FUNCTION\_SENTENCE, METHOD\_SENTENCE, FOR\_STATEMENT, WHILE\_STATEMENT, TRY\_CATCH\_STATEMENT, JSON\_STATEMENT, STATIC, BOOLEAN, LIST, OBJECT, STATIC\_GROUP, VAR\_GROUP, BLOCK

#### **3.2 Hasil CNF**

Hasil CNF yang kami buat sebanyak 712 *grammar*. Hasil CNF ini merupakan konversi dari CGF dengan bantuan fungsi **cfg2cnf**.

## BAB 3

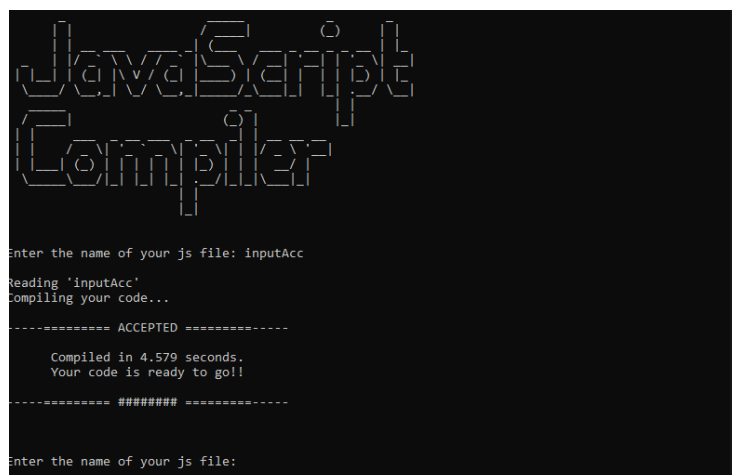
### IMPLEMENTASI DAN PENGUJIAN

#### 3.1 Implementasi Program

Program ini memanfaatkan teori Finite Automata (FA), Context-Free Grammar (CFG), Chomsky's Normal Form (CNF), dan algoritma Cocke-Younger-Kasami (CYK). Program dimulai dengan meminta *input* dari user berupa *path file* kode yang akan di-*compile*. Program membaca kode sebagai teks lalu merubah masing-masing kata menjadi *token* yang sudah didefinisikan pada program. Program memproses token-token tersebut dan memeriksa setiap kata atau simbol. Token yang telah dibuat diperiksa susunannya berdasarkan *grammar* CNF untuk mengetahui kebenaran sintaks kode dan memberitahu hasilnya kepada pengguna. Apabila program mendeteksi karakter yang tidak *valid* pada token atau susunan token yang tidak terdaftar pada CNF, program akan mengembalikan "*Syntax error*". Apabila kode sesuai dengan sintaks yang terdaftar, program akan mengembalikan "*Accept*".



Gambar 3.1.1 Tampilan idle



Gambar 3.1.2 Tampilan 'Accept'

```
E:\Kuliah\Semester 1 2022-2023\IF2124\Tubes\javascript-compiler>python "Javascript Compiler.py"

JavaScript
Compiler

Enter the name of your js file: InputReject.js
Reading 'InputReject.js'
Compiling your code...

----- SYNTAX ERROR -----
          Maybe you forgot a comma?
----- ##### -----

E:\Kuliah\Semester 1 2022-2023\IF2124\Tubes\javascript-compiler>
```

Gambar 3.1.3 Tampilan 'Syntax Error'

## 3.2 Fungsi dan Prosedur Program

Berikut yaitu fungsi dan prosedur yang kami buat terkait *compiler* ini.

### 3.2.1 Fungsi tokenizer

Fungsi ini membaca masukan kode JavaScript dari pengguna dan mengubahnya menjadi token. Fungsi ini juga memeriksa setiap kata pada kode. Setelah membaca dan memeriksa kode, fungsi mengembalikan *array* berisi token-token yang telah dibuat.

### 3.2.2 Fungsi createToken

Fungsi ini membaca masukan kode JavaScript dari user, mengubahnya menjadi token, dan mengembalikan *array* berisi token dengan memanfaatkan fungsi **tokenizer**. Fungsi ini juga menulis hasil token ke dalam file teks bernama **tokenResult.txt**.

### 3.2.3 Fungsi readCFG

Fungsi ini membaca grammar CFG pada file **cfg.txt**.

### 3.2.4 Fungsi addGrammar

Fungsi ini menambahkan setiap aturan pada *grammar* CFG dan menambahkan hasilnya ke dalam *dictionary grammars*.

### 3.2.5 Fungsi mapCNF

Fungsi ini membaca *dictionary grammars* dan memetakan masing-masing *grammar*.

### 3.2.6 Fungsi cfg2cnf

Fungsi ini membaca *grammar* CFG dan mengubahnya menjadi bentuk CNF dengan memanfaatkan fungsi **readCFG**, **addGrammar**, dan **mapCNF**.

### 3.2.7 Fungsi parse

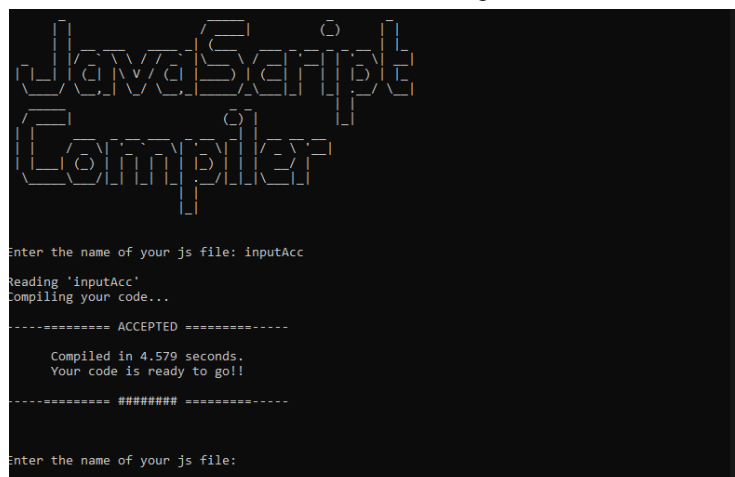
Fungsi ini membaca hasil token **createToken** dan *grammar* CNF lalu membandingkannya untuk memeriksa kebenaran *grammar* pada kode.

### 3.3 Pengujian

#### 3.3.1 Kode diterima 'ACCEPT'

```
function do_something(x) {  
  // sample-comment  
  if (x == 0) {  
    return 0;  
  } else if (x + 4 == 1) {  
    if (true) {  
      return 3;  
    } else {  
      return 2;  
    }  
  } else if (x == 32) {  
    return 4;  
  } else {  
    return "Momen";  
  }  
}
```

Gambar 3.3.1 Kode input



```
JavaScript  
Compiler  
  
Enter the name of your js file: inputAcc  
Reading 'inputAcc'  
Compiling your code...  
  
----- ACCEPTED -----  
  
Compiled in 4.579 seconds.  
Your code is ready to go!!  
  
----- ##### -----  
  
Enter the name of your js file:
```

Gambar 3.3.2 Tampilan program

### 3.3.2 Kode tidak sesuai 'SYNTAX ERROR'

```
function do_something(x) {  
    // This is a sample multiline comment  
    if (x == 0) {  
        return 0;  
    } else if x + 4 == 1 {  
        if (true) {  
            return 3;  
        } else {  
            return 2;  
        }  
    } else if (x == 32) {  
        return 4;  
    } else {  
        return "Momen";  
    }  
}
```

Gambar 3.3.3 Kode input

```
E:\Kuliah\Semester 1 2022-2023\IF2124\Tubes\javascript-compiler>python "Javascript Compiler.py"  
  
Javascript  
Compiler  
  
Enter the name of your js file: inputReject.js  
Reading 'inputReject.js'  
Compiling your code...  
  
----- SYNTAX ERROR -----  
Maybe you forgot a comma?  
-----  
  
E:\Kuliah\Semester 1 2022-2023\IF2124\Tubes\javascript-compiler>
```

Gambar 3.3.4 Tampilan program

### 3.3.2 Kode tidak sesuai 'ILLEGAL CHARACTER'

```
if(true) return \;
```

Gambar 3.3.5 Kode input



```
E:\Kuliah\Semester 1 2022-2023\IF2124\Tubes\javascript-compiler>python "Javascript Compiler.py"

JavaScript
Compiler

Enter the name of your js file: 15-illegalChar.js
Reading '15-illegalChar.js'
Compiling your code...

----- SYNTAX ERROR -----

    Illegal character '\' found at 1:8.

----- ##### -----

E:\Kuliah\Semester 1 2022-2023\IF2124\Tubes\javascript-compiler>
```

Gambar 3.3.6 Tampilan program

## LAMPIRAN

### Repositori Github

<https://github.com/ditramadia/javascript-compiler>

### Pembagian Tugas

| Nama                    | NIM      | TUGAS   |
|-------------------------|----------|---|
| Bagus Lathif Firmansyah | 13521017 | -   |
| Ditra Rizqa Amadia      | 13521019 | <ul style="list-style-type: none"><li>- Program utama</li><li>- Token</li><li>- CFG</li><li>- CNF</li><li>- CYK parser</li><li>- Test kasus</li><li>- Laporan</li></ul> |
| Varraz Hazzandra Abrar  | 13521020 | <ul style="list-style-type: none"><li>- Laporan</li><li>- README Program</li></ul>  |