

# **LAPORAN TUGAS BESAR**

## **IF2124 TEORI BAHASA FORMAL DAN OTOMATA IMPLEMENTASI PARSER BAHASA JAVASCRIPT (NODE.JS)**

Disusun untuk memenuhi tugas mata kuliah Teori Bahasa Formal dan Otomata  
pada Semester 1 (satu) Tahun Akademik 2022/2023.



**Oleh**

|                              |                 |
|------------------------------|-----------------|
| <b>Akmal Mahardika N. P.</b> | <b>13521070</b> |
| <b>Razzan Daksana Yoni</b>   | <b>13521087</b> |
| <b>Muhamad Aji Wibisono</b>  | <b>13521095</b> |

**Kelompok TubesWangiWangiÃÃÃHHH<333**

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

## DAFTAR ISI

|  |           |
|--|-----------|
| <b>BAB I LATAR BELAKANG .....</b>  | <b>1</b>  |
| <b>BAB II TEORI SINGKAT .....</b>  | <b>1</b>  |
| 2.1    Finite Automata .....   | 1         |
| 2.2    Context Free Grammar .....  | 1         |
| 2.3    Chomsky Normal Form .....   | 1         |
| 2.4    Algoritma Cocke–Younger–Kasami .....                                  | 2         |
| 2.5    Javascript .....  | 3         |
| 2.5.1    Node.js .....   | 4         |
| 2.5.2    Syntax Javascript.....  | 4         |
| <b>BAB III IMPLEMENTASI PUSTAKA DAN PROGRAM DALAM BAHASA<br/>PYTHON.....</b> | <b>5</b>  |
| 3.1    Pembuatan Grammar .....   | 5         |
| 3.2    Parser CFG ke CNF.....  | 5         |
| 3.3    Tokenisasi File Input .....   | 6         |
| 3.4    Pengaplikasian Algoritma CYK .....                                    | 9         |
| 3.5    Program Utama.....  | 10        |
| <b>BAB IV .....</b>  | <b>11</b> |
| <b>EKSPERIMEN.....</b>   | <b>11</b> |
| 4.1. Accepted.....   | 11        |
| 4.2. Rejected.....   | 14        |
| <b>DAFTAR REFERENSI .....</b>  | <b>18</b> |
| <b>LAMPIRAN.....</b>   | <b>19</b> |

## **BAB I**

### **LATAR BELAKANG**

Parsing adalah proses dalam menganalisis teks dalam program dan menentukan apakah teks tersebut diterima oleh bahasa program. Parsing ini bertujuan untuk memastikan instruksi yang dibuat oleh programmer mengikuti aturan yang sudah ditentukan oleh bahasa tersebut. Pemeriksa sintaks ada, baik pada bahasa interpreter maupun compiler. Perbedaannya terletak pada apa yang dilakukan setelah proses pemeriksaan (kompilasi/compile) tersebut selesai dilakukan.

Parsing membutuhkan grammar bahasa dan algoritma parser. Banyak grammar dan algoritma yang dikembangkan untuk menghasilkan compiler dengan performa yang tinggi. Terdapat CFG, CNF-e, CNF+e, 2NF, 2LF, untuk grammar yang dapat digunakan, dan terdapat LL(0), LL(1), CYK, Earley's Algorithm, LALR, GLR, Shift-reduce, SLR, LR(1), untuk algoritma yang dapat digunakan untuk melakukan parse.

Tugas besar ini bertujuan membuat program parsing untuk beberapa statement dan sintaks bawaan JavaScript (Node.js). Program parsing menggunakan konsep CFG untuk mengevaluasi sintaks, menggunakan FA untuk mengevaluasi komponen komponen dalam sintaks. Pada tugas besar ini, algoritma Cocke Younger Kasami (CYK) dipilih.

Program menangani kata kunci bawaan JavaScript pada tabel dibawah ini,

|        |      |       |         |          |          |
|--------|------|-------|---------|----------|----------|
| break  | case | catch | cons    | continue | default  |
| delete | else | false | finally | for      | function |
| if     | let  | null  | retrun  | switch   | throw    |
| try    | true | var   | while   |          |          |

Akan tetapi, ada hal-hal yang dapat diabaikan dalam parsing program: semantik dari objek (seperti objek Foo belum terdefinisi tapi Foo.method() tetap diterima), arti semantik dari method, regex dalam bentuk apapun, seperti r-string, r'123', syntactic sugar, karakter-karakter di luar cakupan ASCII, dan indentasi.

## BAB II

### TEORI SINGKAT

#### 2.1 Finite Automata

Finite Automata (FA) atau Finite State Automata (FSA) adalah suatu mesin abstrak yang digunakan untuk merepresentasikan penyelesaian suatu persoalan dari suatu sistem diskrit. Sebagai sebuah mesin maka FSA akan bekerja jika diberi suatu masukan. Hasil proses dari suatu nilai kebenaran diterima atau tidaknya masukan yang diberikan. FSA memiliki state yang banyaknya berhingga, jika diberikan suatu symbol input maka dapat terjadi suatu perpindahan dari sebuah state ke state lainnya. Perubahan state tersebut dinyatakan oleh suatu symbol transisi. Mekanisme FSA tidak memiliki memori sehingga selalu mendasarkan prosesnya pada posisi state “saat ini”. Misalnya pada mekanisme kontrol sebuah lift, selalu didasari pada posisi lift saat itu pada suatu lantai, pergerakan ke atas atau ke bawah dan sekumpulan permintaan yang belum terpenuhi. Finite State Automata merupakan suatu alat yang berguna untuk merancang sistem nyata.

#### 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  $\epsilon$ , Contoh aturan produksi yang termasuk CFG adalah seperti berikut ini:

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

*Figure 1.1 Contoh CFG*

CFG adalah tata bahasa yang mempunyai tujuan sama seperti halnya tata bahasa regular yaitu merupakan suatu cara untuk menunjukkan bagaimana menghasilkan suatu untai-untai dalam sebuah bahasa.

#### 2.3 Chomsky Normal Form

Bentuk normal Chomsky atau Chomsky Normal Form (CNF) merupakan salah satu bentuk normal yang sangat berguna untuk Context Free Grammar.

Bentuk normal Chomsky dapat dibuat dari sebuah tata bahasa bebas konteks yang telah mengalami penyederhanaan yaitu penghilangan produksi useless, unit, dan  $\epsilon$ . 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  $\epsilon$

Bentuk normal Chomsky (Chomsky Normal Form, CNF) adalah Context Free Grammar (CFG) dengan setiap produksinya berbentuk :

$$A \rightarrow BC \text{ atau } B \rightarrow b \text{ atau } C \rightarrow c \mid CC$$

*Figure 2.1 Contoh CNF*

Langkah-langkah pembentukan bentuk normal Chomsky secara umum sebagai berikut:

- 1) Biarkan aturan produksi yang sudah dalam bentuk normal Chomsky
- 2) Lakukan penggantian aturan produksi yang ruas kanannya memuat simbol terminal dan panjang ruas kanan lebih dari 1
- 3) Lakukan penggantian aturan produksi yang ruas kanannya memuat lebih dari 2 simbol variabel
- 4) Penggantian-penggantian tersebut bisa dilakukan berkali-kali sampai akhirnya semua aturan produksi dalam bentuk normal Chomsky
- 5) Selama dilakukan penggantian, kemungkinan kita akan memperoleh aturan-aturan produksi baru, dan juga memunculkan simbol-simbol variabel baru

#### **2.4 Algoritma Cocke–Younger–Kasami**

Algoritma Cocke-Younger-Kasami (CYK) merupakan algoritma parsing dan keanggotaan (membership) untuk tata Bahasa bebas konteks. Algoritma ini diciptakan oleh J. Cocke, DH. Younger, dan T. Kasami. Syarat untuk penggunaan algoritma ini adalah tata bahasa harus berada dalam bentuk normal Chomsky. Obyektif dari algoritma ini adalah untuk menunjukkan apakah suatu string dapat diperoleh dari suatu tata bahasa.

Algoritma CYK menggunakan tabel dua dimensi untuk menyimpan hasil keputusan permasalahan yang lebih kecil terlebih dahulu. Sisi Program Dinamis dari algoritma ini terletak pada pembangunan array dua dimensi atau tabel saat memarsing sebuah untai, kemudian ketika parsing untai dilakukan dalam iterasi selanjutnya, algoritma ini akan memanfaatkan array atau tabel yang telah dibangun sebelumnya. Dari tabel yang telah terbentuk, untai yang diparsing dapat diketahui apakah valid, dalam artian CFG tersebut dapat memproduksi untai tersebut melalui aturan-aturan yang ada. Berikut ini adalah persyaratan yang dibentuk dengan mengaplikasikan CYK:

- Input: untai dengan  $n$  simbol
- Output: valid/ tak valid
- Struktur data: tabel  $n \times n$
- Baris dengan indeks 0 sampai  $n-1$  (atau  $1-n$  dengan modifikasi)
- Kolom dengan indeks 1 sampai  $n$
- Sel  $[i,j]$  simbol yang termasuk dalam untai input

Setiap kali variabel produksi dapat dikembalikan, maka simbol dalam untai berkurang satu. Apabila tidak ada hasil produksi, maka sel diberikan penanda khusus, atau dikosongkan. Apabila untai masih berisi simbol setelah iterasi selesai sampai pada sel batas, maka untai tersebut tidak valid.

CYK merupakan algoritma yang cukup efisien dalam hal mengenali CFG apapun dan merupakan algoritma dasar yang diperkenalkan dalam pemrograman compiler suatu bahasa pemrograman.

## **2.5 Javascript**

JavaScript adalah bahasa pemrograman yang digunakan developer untuk membuat halaman web yang interaktif. Dari menyegarkan umpan media sosial hingga menampilkan animasi dan peta interaktif, fungsi JavaScript dapat meningkatkan pengalaman pengguna situs web. Sebagai bahasa skrip sisi klien, JavaScript adalah salah satu teknologi inti dari World Wide Web. Secara historis, halaman web statis, mirip dengan halaman dalam buku. Halaman statis terutama menampilkan informasi dalam tata letak tetap dan tidak melakukan semua yang kita

harapkan saat ini dari situs web modern. JavaScript muncul sebagai teknologi sisi peramban untuk menjadikan aplikasi web lebih dinamis. Dengan JavaScript, browser dapat merespons interaksi pengguna dan mengubah tata letak konten di halaman web.’

### 2.5.1 Node.js

Node.js adalah runtime environment untuk JavaScript yang bersifat open-source dan cross-platform. Dengan Node.js kita dapat menjalankan kode JavaScript di mana pun, tidak hanya terbatas pada lingkungan browser. Node.js menjalankan V8 JavaScript engine (yang juga merupakan inti dari Google Chrome) di luar browser. Ini memungkinkan Node.js memiliki performa yang tinggi. Node.js juga menyediakan banyak library/module JavaScript yang membantu menyederhanakan pengembangan aplikasi web. Node.js dirancang untuk aplikasi dengan proses I/O yang intensif seperti network server atau backend API. Pemrograman dengan multithreading relatif lebih berat dan sulit untuk dilakukan.

### 2.5.2 Syntax Javascript

Sintaks JavaScript adalah seperangkat aturan yang menentukan program JavaScript yang terstruktur dengan benar. JavaScript terdiri dari pernyataan JavaScript yang ditempatkan di dalam tag HTML `<script>` `</script>` di halaman web, atau di dalam file JavaScript eksternal yang memiliki ekstensi `.js`. Contoh berikut menunjukkan bagaimana pernyataan JavaScript terlihat:

```
var x = 5;
var y = 10;
var sum = x + y;
document.write(sum); // Prints variable value
```

*Figure 3.2 Contoh sintaks JavaScript*

JavaScript sensitif dengan penulisan huruf besar-kecil. Ini berarti bahwa variabel, kata kunci bahasa, nama fungsi, dan pengidentifikasi lainnya harus selalu diketik dengan huruf besar. Sebagai contoh, variabel `myVar` harus diketik `myVar` bukan `MyVar` atau `myvar`. Demikian pula, nama metode `getElementById()` harus diketik dengan case yang tepat bukan sebagai `getElementByID()`.

# BAB III

## IMPLEMENTASI PUSTAKA DAN PROGRAM DALAM BAHASA PYTHON

### 3.1 Pembuatan Grammar

Grammar dibuat berdasarkan CFG yang telah dijelaskan pada Bab II. Grammar sengaja dibuat tanpa ada *production* antara terminal dan nonterminal. Cara pembuatan tersebut membantu ketika menganalisis grammar dan membantu dalam mempercepat proses parsing.

Akan tetapi, ketika grammar, CFG, diubah ke CNF terdapat pola dalam terminal-terminal baru yang dibentuk. Oleh karena itu, grammar diubah kembali dan disesuaikan dengan hasil tokenisasi

### 3.2 Parser CFG ke CNF

Grammar yang telah dibuat lalu akan ditransformasikan ke dalam bentuk Chomsky Normal Form berdasarkan Algoritma yang telah dijelaskan di Bab II. Disebabkan adanya perjanjian pada tahap pembuatan grammar, beberapa tahapan parser dapat dilewati tanpa melakukan apapun. Berikut adalah ringkasan header dari implementasi yang penulis lakukan:

Tabel 1.3 Spesifikasi file CFG2CNF.py

|   |
|---|
| Nama File: CFG2CNF.py   |
| <b>Variabel Global:</b><br><br>NewVars (Array of string):<br>Array dari string random untuk menambahkan nama variabel baru  |
| <b>def loadCFG(path) :</b><br># I.S. File berbentuk txt dengan format CFG dan mark '~'<br># F.S. tiga array yaitu array yang berisi terminal, variabel, dan production rule yang sudah terpisah untuk tiap or |



|   |
|---|
| <pre> def simple(terminals, production):     # I.S. Menerima array terminal, variabel, dan     SATU rule production     # F.S. Mengembalikan true jika production sudah     berbentuk satu terminal saja </pre>   |
| <pre> def unitable(variables, production):     # I.S. Menerima array terminal, variabel, dan     SATU rule production     # F.S. Mengembalikan true jika production sudah     berbentuk satu variable saja </pre> |
| <pre> def unite(variables, productions):     # I.S. Menerima array variabel dan productions     # F.S. Mengembalikan array production baru dengan     production yang unitable digabungkan </pre>                 |
| <pre> def prodToDict(productions):     # I.S. Menerima array productions     # F.S. Mengembalikan dictionary dari array     production </pre>   |
| <pre> def CFGtoCNF(path):     # I.S. Menerima alamat file grammar yang telah     dibuat     # F.S. Mengembalikan dictionary dari grammar yang     telah dibuat dalam bentuk CNF </pre>                            |

### 3.3 Tokenisasi File Input

Agar dapat diterima oleh algoritma CYK, file yang diberikan perlu disederhanakan terlebih dahulu menjadi terminal yang ada pada grammar saja. Oleh karena itu perlu dilakukan penyederhanaan terhadap file input menjadi sebuah array dari string token yang merupakan terminal yang terdapat pada grammar.

Pada proses tokenisasi dilakukan penyederhanaan sampai terbentuk terminal. Selain itu digunakan juga Finite Automata untuk mendeteksi adanya

kesalahan pada file input sebelum akhirnya diaplikasikan pada CYK. Berikut adalah ringkasan header dari implementasi finite automata yang penulis lakukan:

*Tabel 2.3 Spesifikasi file Simplifier.py*

|   |
|---|
| Nama File: SimplifierFA.py  |
| <b>Variabel Global:</b><br><br>StateMachine (integer):<br>Variabel untuk menunjukkan state dari simplifier<br>0 : Reject<br>1 : Acc<br>2 : deteksi tutup petik atau komen<br>3 : detektsi petik atau komen<br>4 : deteksi huruf pertama identifier<br>5 : deteksi identifier selain pertama<br>6 : detektsi angka |
| <b>def removeComments(testcase) :</b><br># I.S. String yang diambil dari file<br># F.S. Mengembalikan string yang diambil dari file dengan komen dihilangkan  |
| <b>def removeStrings(testcase) :</b><br># I.S. String yang sudah dihilangkan komennya<br># F.S. Mengembalikan string yang diambil dari file dengan isi string dihilangkan   |
| <b>def identifierFirst(char) :</b><br># I.S. Menerima karakter pertama dari sebuah identifier<br># F.S. Mengganti stateMachine sesuai kondisi yang didapatkan   |
| <b>def identifierBody(char) :</b><br># I.S. Menerima karakter selain pertama dari sebuah identifier   |

|   |
|---|
| # F.S. Mengganti stateMachine sesuai kondisi yang didapatkan  |
| <b>def identifier(string):</b><br># I.S. Menerima potongan string dan mendeteksi apakah string tersebut merupakan identifier<br># F.S. Mengembalikan true jika stateMachine tidak masuk ke dead state (stateMachine != 0) |
| <b>def numberBody(char):</b><br># I.S. Menerima karakter dari sebuah angka<br># F.S. Mengganti stateMachine sesuai kondisi yang didapatkan  |
| <b>def number(string):</b><br># I.S. Menerima potongan string dan mendeteksi apakah string tersebut merupakan angka<br># F.S. Mengembalikan true jika stateMachine tidak masuk ke dead state (stateMachine != 0)          |

Berikut adalah ringkasan header dari implementasi tokenizer yang penulis lakukan:

*Tabel 3.3 Spesifikasi file Tokenizer.py*

|   |
|---|
| Nama File: Tokenizer.py   |
| <b>Variabel Global:</b><br><br>StateMachine (integer):<br>Variabel global yang terdapat pada simplifierFA                           |
| <b>def readFile(filename):</b><br># I.S. Menerima alamat dari test file<br># F.S. Mengembalikan string dari test file yang diterima |
| <b>def transformEnters(testcase):</b><br># I.S. Menerima string dari file   |

|  |
|--|
| <pre># F.S. Mengembalikan string dengan karakter 'pindah baris' diubah menjadi terminal yang terdapat pada grammar</pre>   |
| <pre><b>def splitOperators(testcase) :</b>     # I.S. Menerima string dari file     # F.S. Mengembalikan array token dengan membagi     komponen command serta operator yang terdapat pada     string menjadi komponen tersendiri</pre>                            |
| <pre><b>def simplifyIdNNum(testcase) :</b>     # I.S. Menerima array token yang sudah dibagi     untuk tiap operator dan command     # F.S. Mengembalikan array token dengan angka dan     identifier diubah menjadi terminal yang terdapat pada     grammar</pre> |
| <pre><b>def tokenize(path) :</b>     # I.S. Menerima alamat dari test file     # F.S. Mengembalikan array token dengan     komponennya diubah menjadi terminal yang terdapat     pada grammar</pre>  |

### 3.4 Pengaplikasian Algoritma CYK

Setelah mendapatkan kamus berbentuk CNF dan file input yang distrukturkan untuk tiap terminal, jika tidak terjadi kesalahan pada tokenisasi akan dilakukan deteksi kevalidan bahasa dengan menggunakan algoritma CYK. Berikut adalah ringkasan header dari implementasi algoritma CYK yang penulis lakukan:

*Tabel 4.3 Spesifikasi file CYK.py*

|  |
|--|
| Nama File: CYK.py  |
| <pre><b>def checkCYK(testCase, CNFdict):</b>     # I.S. Menerima test case berupa array of token yang sudah disesuaikan     dengan tokenizer</pre> |

|   |
|---|
| # F.S. Mengembalikan true jika bahasa valid, false jika tidak |
|---|

### 3.5 Program Utama

*Tabel 5.3 Spesifikasi file main.py*

|  |
|--|
| Nama File: main.py   |
| # I.S. Menerima pembacaan file sebarang  |
| # F.S. Mengembalikan hasil pembacaan file apakah file javascript tersebut bisa diterima atau tidak |

## BAB IV EKSPERIMEN

### 4.1. Accepted

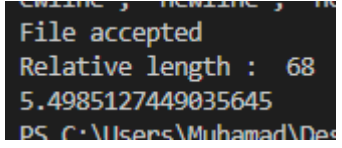
*Tabel 6.4 Test program yang diterima*

| Screenshot Code  | Keterangan dan Hasil  |
|--|---|
| <pre> JS test.js &gt; ... 1  x=555  21 123&amp;&amp;12&amp;70 2  if (x == 0) { 3      return 0 //ini komen 4  } else if (x+4==1) { 5      if (true) { 6          return 3 7      } else { 8          return 2 </pre> | <p>Hasil:</p> <pre> File accepted Relative length : 607 2960.873663663864 </pre> <p>Keterangan:<br/>File ini merupakan copy-paste dari modifikasi contoh accepted yang terdapat pada spesifikasi. Tujuannya untuk menghitung waktu yang diperlukan, didapat waktu 49 menit 20 detik</p> |
| <pre> 110  return 'Momin' 111  } else 112  asdasd5 = 8 113  car = {type:"Fiat \'", model:"500 \' \' ", color:"white 114  return car[1] 115 </pre>  | <p>Hasil:</p> <pre> File accepted Relative length : 49 1.940427541732788 </pre> <p>Keterangan:<br/>Nested Loop</p>  |
| <pre> for (var i = 0; i ; i++){     for (var j = 0; j ; j++){         if (i == j) {             console.log(i)         }     } } </pre>  | <p>Hasil:</p> <pre> File accepted Relative length : 14 0.053337812423706055 </pre> <p>Keterangan:<br/>Function in variable</p>  |
| <pre> func = function foo(fa,fb) {     return "Hello World" } </pre>   | <p>Hasil:</p> <pre> File accepted Relative length : 17 0.08271265029907227 </pre> <p>Keterangan:<br/>Function in variable with title</p>  |

|  |   |
|--|---|
|  |   |
| <pre>delete foo.gfdsg["sfdg"].gfh['gfd']</pre>   | <p>Hasil:</p> <pre>File accepted Relative length : 15 0.05879807472229004</pre> <p>Keterangan :<br/>Delete function</p>           |
| <pre>for (i ; i &lt; 10; i++) {     if (x + y &gt; 10) {         break     } else {         continue     } }</pre>             | <p>Hasil:</p> <pre>File accepted Relative length : 35 0.7597301006317139</pre> <p>Keterangan :<br/>Break and continue in loop</p> |
| <pre>switch(n) {     case 1:         text = "January"         break     case 2:         break     case 3:     default: }</pre> | <p>Hasil :</p> <pre>File accepted Relative length : 32 0.5510621070861816</pre> <p>Keterangan :<br/>Switch, case, default</p>     |
| <pre>var i let j = i const k = i</pre>   | <p>Hasil :</p> <pre>File accepted Relative length : 14 0.05377554893493652</pre> <p>Keterangan :<br/>Variable declaration</p>     |

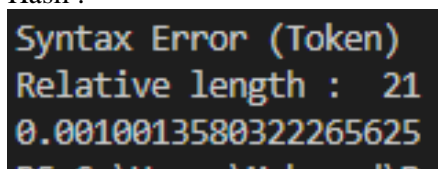
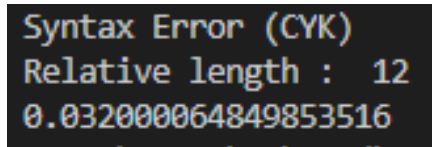
|  |  |
|--|--|
| <pre>while (i &lt; N) {     i++ }</pre>  | <p>Hasil :</p> <pre>File accepted Relative length : 13 0.04451608657836914</pre> <p>Keterangan :<br/>While loop</p>                  |
| <pre>if (kamu != "tidur"){     console.log("kamu tidak tidur") } else if (jamTidur &gt; 8){     console.log("tidur mulu") } else {     console.log("bangun hey nubes") }</pre> | <p>Hasil :</p> <pre>File accepted Relative length : 49 2.009424924850464</pre> <p>Keterangan :<br/>Conditional if, else if, else</p> |
| <pre>throw tubes</pre>   | <p>Hasil :</p> <pre>File accepted Relative length : 3 0.0013241767883300781</pre> <p>Keterangan :<br/>Throw declaration</p>          |
| <pre>found = false  if (true){  }</pre>  | <p>Hasil :</p> <pre>File accepted Relative length : 14 0.050252676010131836</pre> <p>Keterangan :<br/>boolean</p>                    |
| <pre>point = null point = NaN</pre>  | <p>Hasil :</p> <pre>File accepted Relative length : 8 0.010618448257446289</pre> <p>Keterangan :</p>                                 |



|  |  |
|--|--|
| <pre> test &gt; JS debugjs &gt; ... 1 2 let x = document.getElementById("641724")[5].innerHTML = err.message 3 4 try { 5     addlert("Welcome guest!") 6 } 7 catch(err) { 8     document.getElementById("641724")[5].innerHTML = err.message 9 } 10 finally{ 11 12 13 14 15 </pre> | <p>Null, NaN</p> <p>Hasil:</p>  <p>Keterangan:<br/>Try, Catch, Finally</p> |
| <pre> 1 function fib(n){ 2     // mengecek apakah faktorial c 3     if (n == 0){ 4         return 1 5     } 6     else { 7         return n * fib(n-1) 8     } 9 } </pre>  | <p>Hasil:</p> <p>Keterangan:<br/>Function, if, else</p>  |

## 4.2. Rejected

Tabel 7.4 Test program ditolak

|   |  |
|---|--|
| <pre> const longString = 'This is a very long string which needs to wrap across multiple lines because otherwise my code is unreadable.' </pre> | <p>Hasil :</p>  <p>Keterangan :<br/>String gak inline</p>      |
| <pre> if (x == 2){     break } </pre>   | <p>Hasil :</p>  <p>Keterangan :<br/>Break datang tiba-tiba</p> |

|   |  |
|---|--|
| <pre>default :   x = 0   break</pre>                      | <p>Hasil :</p> <pre>Syntax Error (CYK) Relative length : 9 0.013000011444091797</pre> <p>Keterangan :<br/>Default tanpa ada switch</p> |
| <pre>for(i = 0; i &lt; 5, ++i) {   console.log(i) }</pre> | <p>Hasil :</p> <pre>Syntax Error (CYK) Relative length : 25 0.26000022888183594</pre> <p>Keterangan :<br/>Koma di header for</p>       |
| <pre>return return 1</pre>                                | <p>Hasil :</p> <pre>Syntax Error (CYK) Relative length : 4 0.0019996166229248047</pre> <p>Keterangan :<br/>Return dua kali</p>         |
| <pre>var = 1</pre>  | <p>Hasil :</p> <pre>Syntax Error (CYK) Relative length : 4 0.0019998550415039062</pre> <p>Keterangan :<br/>Var tidak boleh</p>         |
| <pre>const i</pre>  | <p>Hasil :</p> <pre>Syntax Error (CYK) Relative length : 4 0.0020008087158203125</pre> <p>Keterangan :<br/>Const tanpa deklarasi</p>   |
| <pre>delete</pre>   | <p>Hasil :</p> <pre>Syntax Error (CYK) Relative length : 2 0.0009987354278564453</pre> <p>Keterangan :<br/>Delete tanpa parameter</p>  |

|   |   |
|---|---|
| <pre>function (x,y) {     You, 1 second }</pre> | <p>Hasil :</p> <pre>Syntax Error (CYK) Relative length : 11 0.02300095558166504</pre> <p>Keterangan :<br/>Function tanpa title</p>    |
| <pre>switch () {     case 1: }</pre>            | <p>Hasil :</p> <pre>Syntax Error (CYK) Relative length : 11 0.023000001907348633</pre> <p>Keterangan :<br/>Switch tanpa parameter</p> |
| <pre>while () {     You,</pre>                  | <p>Hasil :</p> <pre>Syntax Error (CYK) Relative length : 11 0.025908470153808594</pre> <p>Keterangan :<br/>While tanpa parameter</p>  |
| <pre>if (x) {     case 1:     You,</pre>        | <p>Hasil :</p> <pre>Syntax Error (CYK) Relative length : 12 0.030843257904052734</pre> <p>Keterangan :<br/>Case tanpa switch</p>      |
| <pre>else {     return 2 }</pre>                | <p>Hasil :</p> <pre>Syntax Error (CYK) Relative length : 8 0.009966850280761719</pre> <p>Keterangan :<br/>Else tanpa if</p>           |

|   |   |
|---|---|
| <pre>if x + 4 === 5 {</pre>   | <p>Hasil :</p> <pre>File accepted Relative length : 13 0.039870500564575195</pre> <p>Keterangan :<br/>Parameter if tanpa kurung</p>                               |
| <pre>catch (x) {</pre>  | <p>Hasil :</p> <pre>Syntax Error (CYK) Relative length : 9 0.013695240020751953</pre> <p>Keterangan :<br/>Catch tanpa try</p>                                     |
| <pre>let let = 1</pre>  | <p>Hasil :</p> <pre>Syntax Error (CYK) Relative length : 5 0.0031359195709228516</pre> <p>Keterangan :<br/>Let sebagai nama variable dengan let adalah fungsi</p> |
| <pre>try {   var x = 1;   var y = 2;   var z = x + y;   console.log(z); }</pre> | <p>Hasil :</p> <pre>Syntax Error (CYK) Relative length : 33 0.6312499046325684</pre> <p>Keterangan :<br/>try tanpa catch</p>                                      |
| <pre>if (x + 2 &gt; 5) {   continue }</pre>                                     | <p>Hasil :</p> <pre>Syntax Error (CYK) Relative length : 14 0.05242919921875</pre> <p>Keterangan :<br/>Continue bukan di loop</p>                                 |
| <pre>finally {   console.log('finally') }</pre>                                 | <p>Hasil :</p> <pre>Syntax Error (CYK) Relative length : 14 0.0528261661529541</pre> <p>Keterangan :<br/>Finally tanpa try and catch</p>                          |

## DAFTAR REFERENSI

[https://repository.dinus.ac.id/docs/ajar/file\\_2013-03\\_18\\_071001\\_dr.r.\\_heru\\_tjahjana\\_s.si\\_m.si\\_5883745141.pdf](https://repository.dinus.ac.id/docs/ajar/file_2013-03_18_071001_dr.r._heru_tjahjana_s.si_m.si_5883745141.pdf)  
<http://web.if.unila.ac.id/ilmukomputer/cnf-chomsky-normal-form/>  
<https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2007-2008/Makalah2008/MakalahIF2251-2008-079.pdf>  
<https://aws.amazon.com/id/what-is/javascript/>  
<https://www.dicoding.com/blog/apa-itu-node-js/>  
<https://www.webhozz.com/code/sintaks-javascript/>

## **LAMPIRAN**

Repository Github

<https://github.com/akmaldika/Parsing-NodeJs.git>

**Pembagian Tugas**

| <b>Tanggung Jawab</b>                                 | <b>Nama (NIM)</b>  |
|---|--|
| <b>Pembuatan grammar (CFG)</b>                        | <b>Akmal Mahardika N P (13521070)<br/>Razzan Daksana Yoni (13521087)<br/>Muhamad Aji Wibisono (13521095)</b> |
| <b>Pembuatan konversi CFG ke CNF</b>                  | <b>Muhamad Aji Wibisono (13521095)</b>   |
| <b>Pembuuatan konversi input *.js<br/>(Tokenizer)</b> | <b>Muhamad Aji Wibisono (13521095)</b>   |
| <b>Pembuatan simplifier.py</b>                        | <b>Muhamad Aji Wibisono (13521095)</b>   |
| <b>Pembuatan program CYK</b>                          | <b>Akmal Mahardika N P (13521070)<br/>Razzan Daksana Yoni (13521087)</b>                                     |
| <b>Pembuatan main.py</b>                              | <b>Razzan Daksana Yoni (13521087)</b>  |
| <b>Pembuatan Laporan</b>                              | <b>Akmal Mahardika N P (13521070)<br/>Razzan Daksana Yoni (13521087)<br/>Muhamad Aji Wibisono (13521095)</b> |
| <b>Analisis coding (debug)</b>                        | <b>Akmal Mahardika N P (13521070)<br/>Razzan Daksana Yoni (13521087)<br/>Muhamad Aji Wibisono (13521095)</b> |