LAPORAN TUGAS BESAR IF2124

Parser Bahasa JavaScript (Node.js)

Ditujukan untuk memenuhi salah satu tugas besar mata kuliah IF2124 Teori Bahasa Formal dan Otomata pada Semester I Tahun Akademik 2022/2023

Disusun oleh:

Fajar Maulana Herawan (K1)	13521080
Vieri Fajar Firdaus (K1)	13521099
Aulia Mey Diva Annandya (K1)	13521103

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2022

DAFTAR ISI

BAB I
DESKRIPSI MASALAH

JavaScript adalah bahasa pemrograman yang digunakan dalam pengembangan website agar menjadi lebih dinamis. JavaScript diciptakan oleh Brendan Eich di Netscape Communications pada tahun 1955. Mulanya, bahasa pemrograman ini dinamai Mocha, lalu sempat diubah menjadi LiveScript, yang kemudian diubah menjadi JavaScript agar bisa menjadi "teman" bagi bahasa pemrograman Java. Di tahun 2005, pamor JavaScript naik. Library seperti JQuery dan MooTools dikembangkan untuk meminimalisir ketidak-konsistenan browser dan memudahkan untuk diterapkan pada design patterns. Hingga saat ini, JavaScript menjadi bahasa skrip yang banyak digunakan dalam pembuatan website untuk menarik pengunjung seperti situs-situs ternama seperti Google dan Facebook.

Seiring berkembangnya dunia per-website-an maka diperlukan grammar bahasa, pemeriksaan syntax bahasa, dan algoritma parsing terhadap bahasa JavaScript. Pada Tugas Besar Teori Bahasa Formal dan Otomata ini kami membuat sebuah program parser (pemeriksaan syntax) untuk bahasa JavaScript (Node.js).

Dalam proses pembuatan program dari sebuah bahasa menjadi instruksi yang dapat dieksekusi oleh mesin, terdapat pemeriksaan sintaks bahasa atau *parsing* yang dibuat oleh programmer untuk memastikan program dapat dieksekusi tanpa menghasilkan *error*. *Parsing* ini bertujuan untuk memastikan instruksi yang dibuat oleh programmer mengikuti aturan yang sudah ditentukan oleh bahasa tersebut. Baik bahasa berjenis *interpreter* maupun *compiler*, keduanya pasti melakukan pemeriksaan sintaks. Perbedaannya terletak pada apa yang dilakukan setelah proses pemeriksaan (kompilasi/*compile*) tersebut selesai dilakukan.

Dibutuhkan *grammar* bahasa dan algoritma *parser* untuk melakukan *parsing*. Sudah sangat banyak *grammar* dan algoritma yang dikembangkan untuk menghasilkan *compiler* dengan performa yang tinggi. Terdapat CFG, CNF^{-e}, CNF^{+e}, 2NF, 2LF, dll untuk *grammar* yang

dapat digunakan, dan terdapat LL(0), LL(1), CYK, Earley's Algorithm, LALR, GLR, Shift-reduce, SLR, LR(1), dll untuk algoritma yang dapat digunakan untuk melakukan *parsing*.

Pada tugas besar ini, kami diminta untuk mengimplementasikan *parser* untuk JavaScript (Node.js) untuk beberapa *statement* dan sintaks bawaan JavaScript. Gunakanlah konsep CFG untuk pengerjaan *parser* yang mengevaluasi syntax program. Untuk nama variabel dan operasi (+, -, >, dll) dalam program menggunakan FA.

BAB II
TEORI SINGKAT

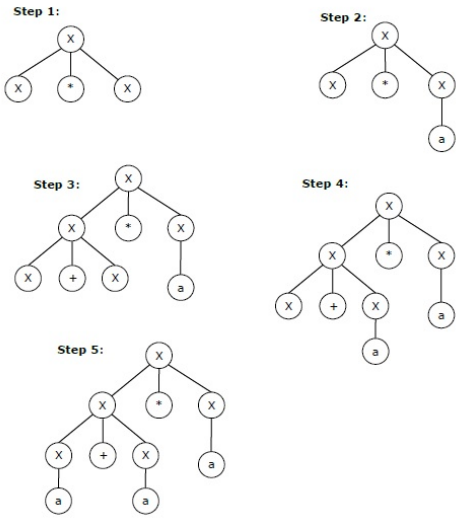
2.1 Context Free Grammar

Context Free Grammar atau CFG adalah tata bahasa formal di mana setiap aturan produksi adalah dalam bentuk A -> B, yang mana A adalah pemproduksi, dan B adalah hasil produksi. Batasannya, ruas kiri adalah simbol variabel sedangkan ruas kanan dapat mencakup terminal, variabel, simbol maupun **ε**. Berikut merupakan contoh dari CFG :

$$S \longrightarrow aAB$$
$$A \longrightarrow bBb$$
$$B \longrightarrow A \mid \lambda$$

CFG juga merupakan tata bahasa yang mempunyai tujuan seperti tata bahasa reguler untuk menunjukkan cara menghasilkan suatu untai-untai dalam sebuah bahasa. Menggambarkan simbol variabel menjadi terminal dapat menggunakan pohon penurunan atau derivation tree yang mana setiap simbol variabel akan diturunkan menjadi terminal.

Suatu context-free grammar dapat bersifat ambigu jika jumlah string berasal dari dua pohon atau lebih, baik turunan kiri (*leftmost derivations)* maupun turunan kanan (*rightmost derivations*). Leftmost derivation adalah simbol variabel yang paling kiri diganti terlebih dahulu, sedangkan rightmost derivation adalah simbol variabel yang paling kanan diganti terlebih dahulu. Contoh dari leftmost derivation dan rightmost derivation :



2.2 CHOMSKY NORMAL-FORM

Chomsky Normal-Form atau biasa disebut CNF merupakan salah satu bentuk normal yang sangat berguna untuk Context Free Grammar (CFG). Bentuk normal Chomsky dapat dibuat dari sebuah tata bahasa bebas konteks yang telah mengalami penyederhanaan yaitu penghilangan produksi useless, unit, dan **ε**. Suatu tata bahasa bebas konteks dapat dibuat menjadi bentuk normal chomsky dengan syarat tata bahasa bebas konteks tidak memiliki produksi useless, unit, dan **ε**. Bentuk dari CNF menjadi CFG sebagai berikut :

$$S \rightarrow aB \quad => \quad S \rightarrow P_1B$$

$$A \rightarrow bc \quad => \quad A \rightarrow P_2P_3$$

$$B \rightarrow Ab \quad => \quad B \rightarrow AP_2$$

$$C \rightarrow aB \quad => \quad C \rightarrow P_1B$$

Terdapat aturan produksi cnf, yaitu: ruas kanannya tepat berupa sebuah simbol terminal atau dua variabel, Jika terdapat lebih dari satu simbol terminal maka harus dilakukan perubahan.

2.3 COCKE-YOUNGER KASAMI

Cocke-Younger Kasami atau biasa disebut algoritma CYK adalah algoritma penguraian untuk menentukan apakah suatu untai (string) dapat diterima oleh suatu Context Free Grammar. Untuk dapat menggunakan algoritma CYK ini dibutuhkan CFG dalam bentuk normal form. Algoritma parsing menggunakan dynamic programming karena proses parsing membutuhkan hasil parsing sebelumnya untuk memutuskan apakah proses yang sedang berlangsung diterima atau tidak. Sebagai contoh pengecekan string "baabbb" dengan menggunaan algoritma CYK sebagai berikut :

6	C,S					
5	B	B,S				
4	S	S	C,S			
3	C	C	B	S		
2	A,S	B	∅	C	C	
1	B	A,C	A,C	B	B	B
	b	**a**	**a**	**b**	**b**	**b**
	1	2	3	4	5	6

BAB III
ANALISIS PERSOALAN

3.1 CFG Production

Non-Terminal Symbol/Variable (V)

NUM1	OP	VALUE	FUNC_SYNTAX	CODE3	ELSE_SYNTAX1
NUM2	NL	VV	FUNC_HEADER	IFF	IF_SYNTAX1
NUMM	KATA	COMPARE	STRING	ELIFF	IFF_SYNTAX2
NUMBER	VAR_2	BOOLEAN	CODE1	IFF_SYNTAX1	ELIF_SYNTAX2
OTHER	VAR	FUNC_CALL	CODE2	ELIF_SYNTAX1	ELSE_SYNTAX2
IF_SYNTAX2	WHILE_SYNTAX	CONST_SYNTAX	VAR_SYNTAX	DICT_VAR	DELETE_SYNTAX
WHILES	ARRAY	LET_SYNTAX	BEBAS_SYNTAX	DICT_CHILD	THROW_SYNTAX
SWITCH_HEADER	FOR_HEADER3	TRYCATCH1_1	TRYCATCH1	SWITCH_FUNC_SYNTAX	TRYCATCH31_FUNC
SWITCH_SYNTAX	FOR_HEADER4	TRYCATCH2_1	TRYCATCH2	DEFAULT_FUNC_SYNTAX	TRYCATCH32_FUNC
DEFAULT_SYNTAX	FOR_HEADER5	TRYCATCH3_1	IF_FUNC_SYNTAX1	CASE_FUNC_SYNTAX	TRYCATCH1_FUNC
CASE_SYNTAX	FOR_HEADER6	TRYCATCH1_2	IF_FUNC_SYNTAX2	FORS_FUNC	TRYCATCH2_FUNC
CASE_DEFAULT_SYNTAX	FOR_HEADER7	TRYCATCH2_2	ELIF_FUNC_SYNTAX1	TRYCATCH11_FUNC	KURUNG
FOR_CONDI	FOR_HEAD	TRYCATCH	ELIF_FUNC	TRYCATCH	ARROWFU

TION	ER8	3_2	_SYNTAX2	12_FUNC	NCT1
FOR_HEADER1	FORS_HEADER	COMS	ELSE_FUNC_SYNTAX	TRYCATCH21_FUNC	ARROWFUNCT2
FOR_HEADER2	FORS	COMMENT	WHILE_FUNC_SYNTAX	TRYCATCH22_FUNC	ARROWFUNCT

Terminal Symbol

if	return	Let	finally	default	for
else	false	const	catch	delete	do
brace	true	var	case	or	and
comma	not	while	switch		

BAB IV
IMPLEMENTASI FUNGSI

4.1 Module convertCFG

Nama Fungsi	Tujuan
loadGrammar	Mengubah dari txt menjadi array
setOneTerminal	Membuat menjadi satu terminal (list apa aja yang punya 1 terminal)
subtitutedRule	Memilah grammar yang memiliki 1 terminal
ConvertToCNF	Algoritma mengubah cfg ke cnf
saveCNF	Menyimpan hasil ke file
converttoDict	Menjadikan bentuk dictionary
CYK	Algoritma CYK
cekVar	Mengecek apakah grammar yang diterima berupa variabel
cekNum	Mengecek apakah grammar yang diterima berupa angka
split	Memisah input file

4.2 File main.py

File main.py merupakan source code yang berisi main program dari *compiler* bahasa Javascript. File ini akan melakukan import import ke semua file yang lain. Program ini akan melakukan pembacaan file py lalu mengubahnya menjadi token splitter, lalu CFG yang telah dibuat akan diubah menjadi CNF, dan akhirnya akan dilakukan *parsing* terhadap token dihasilkan dari

pembacaan tersebut. File ini akan menampilkan output berupa hasil kompilasi dari program JavaScript yang ingin dilakukan pengecekan.

BAB V
EKSPERIMEN

5.1 Hasil FA

```
a = cekVar("hello")
print(a)
b = cekVar("h3ll0")
print(b)
c = cekVar("4ello")
print(c)
```

Hasil :

```
auliameydivaannandya@Aulias-MacBook-Pro TUBES_TBFO % python -u "/Users/auliameydivaannandya/Documents/GitHub/TUBES_TBFO/fa.py"
True
True
False
```

```
print(cekNum("150"))
```

```
print(cekNum("15-0"))
```

Hasil :

```
True
fajarherawan@Fajars-MBP TUBES_TBFO % /u
False
fajarherawan@Fajars-MBP TUBES_TBFO %
```

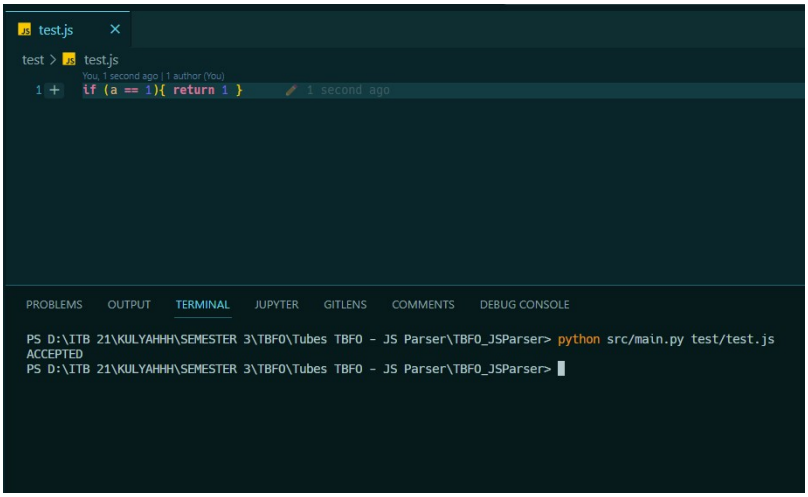
5.2 Hasil Let

```
JS coba.js > [e] a
        You, 1 minute ago | 1 author (You)
1       let a=12        You, 1 minute ago • Uncommitted changes
```

Hasil:

```
PS D:\Kuliah\Semester 3\IF2121 Logika Komputasional\asu\TUBES_TBFO> python
main.py coba.js
Valid
```

5.3 Hasil if



BAB VI
KESIMPULAN

5.1 Kesimpulan

Berdasarkan tugas besar “Parser Bahasa Javascipt”, program yang kami buat berjalan tidak cukup baik mengingat masih banyaknya kekurangan dalam program kami. Pengujian yang dilakukan juga belum memuat semua sintaks bahasa Python sehingga kami tidak bisa mengetahui batasan dari program yang kami buat. CFG yang kami buat juga tidak sepenuhnya meliputi keseluruhan sintaks dari bahasa Python sehingga untuk kasus yang lebih banyak ataupun asing, bisa terjadi beberapa kesalahan.

5.2 Saran

Setelah mengerjakan tugas besar ini, kami memiliki beberapa saran pada program Compiler Bahasa JavaScript kami karena belum tercapai parsing syntax yang sempurna. Hal tersebut disebabkan karena grammar yang kami buat masih belum sempurna. Oleh karena itu, jika diberikan grammar yang lebih luas dan menyeluruh maka program akan berjalan dengan lebih baik.

5.3 Refleksi

Tugas besar kedua ini , kami merasa *time management* kami masih kurang disebabkan beberapa tugas besar juga diberikan dalam kurun waktu yang singkat. Dalam tugas besar kedua ini, kami belajar bukan hanya dalam pemrograman ,melainkan juga mengkoordinasi waktu dan koordinasi antara sesama anggota.

REFERENSI

https://emirat8.github.io/article/aplikasi-pemanfaatan-context-free-grammar/

https://ziakode.com/contoh-algoritma-cyk-inputan-cfg/

LAMPIRAN

Link Repository GitHub : https://github.com/vierifirdaus/TUBES_TBFO.git