LAPORAN TUGAS BESAR
TEORI BAHASA FORMAL DAN OTOMATA

PARSER BAHASA JAVASCRIPT (NODE.JS)

Kelas Mahasiswa K03

Dosen Pengampu: Dr. Judhi Santoso, M.Sc

Diajukan sebagai tugas besar Mata Kuliah IF2124 Teori Bahasa Formal dan Otomata pada Semester I Tahun Akademik 2022/2023

Disusun Oleh:

Kelompok 3 (Ageusjayyukii)

Jason Rivalino	**13521008**
Muhhamad Syauqi Jannatan	**13521014**
Agsha Athalla Nurkareem	**13521027**

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG

2022

DAFTAR ISI

BAB I

DESKRIPSI MASALAH

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.

Untuk melakukan proses parsing ini, dibutuhkan grammar bahasa dan algoritma parser. 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. Berdasarkan hal ini, maka pada tugas besar ini akan mengimplementasikan *parser* untuk JavaScript (Node.js) untuk beberapa *statement* dan sintaks bawaan JavaScript.

BAB II

TEORI DASAR

I. Teori Dasar *Finite Automata* (FA)

Finite Automata atau FA merupakan mesin automata dari sebuah *Regular Language*. Ciri-ciri dari FA sendiri adalah memiliki *state* yang jumlahnya berhingga dan juga dapat berpindah-pindah dari satu state ke state yang lainnya. FA sendiri merupakan penggambaran model abstrak dari komputer digital. Dalam pembentukan model state dari FA, terdapat beberapa elemen yang perlu diperhatikan, antara lain:

Q	Set dari semua states
∑	Set dari semua symbol input
q	Initial State (State awal)
F	Set dari semua final
δ	Fungsi transisi

Untuk jenisnya sendiri, terdapat dua jenis dari FA ini yaitu Deterministic Finite Automata atau DFA dan Non-Deterministic Finite Automata atau NFA. Perbedaan antara kedua model state ini antara lain adalah:

- Untuk model DFA, untuk sebuah state, jika state tersebut diberi sebuah inputan, state akan mengarah tepat pada satu state, baik mengarah pada statenya sendiri ataupun mengarah pada state lainnya.
- Sedangkan, pada model NFA, untuk sebuah state, state bisa saja tidak menerima input ataupun untuk satu inputan, state dapat menuju ke beberapa state lainnya, baik menuju ke state sendiri maupun menuju kearah state-state lainnya.

II. Teori Dasar *Context Free Grammar* (CFG)

Context Free Grammar atau yang biasa disebut CFG merupakan tata bahasa formal di mana setiap aturan produksi adalah dalam bentuk A → B dimana A adalah pemroduksi dan B adalah hasil produksi . Batasannya hanyalah ruas kiri adalah sebuah simbol variabel dan ruas kanan dapat berupa terminal, simbol, variabel ataupun ε. Sehingga, dapat disimpulkan bahwa Context Free Grammar merupakan tata bahasa dimana tidak terdapat pembatasan pada hasil produksinya. Contoh aturan produksi yang termasuk CFG seperti berikut :

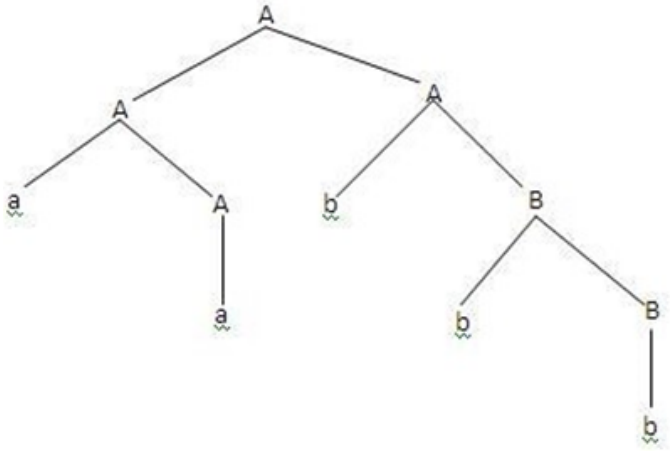
$$X \rightarrow bY \mid Za$$
$$Y \rightarrow aY \mid b$$
$$Z \rightarrow bZ \mid \varepsilon$$

CFG juga merupakan tata bahasa yang mempunyai tujuan sama seperti tata bahasa reguler yaitu untuk menunjukkan bagaimana menghasilkan suatu untai-untai dalam sebuah bahasa. Penggunaan CFG pada kompiler bermaksud untuk mempermudah serta mengurangi ambiguitas dalam parsing. Parsing sendiri merupakan proses pemrosesan string berdasarkan aturan CFG.

Untuk menggambarkan simbol-simbol variabel menjadi simbol terminal ialah dengan menggunakan pohon penurunan atau yang biasa disebut dengan derivation tree/parse tree dimana cara ini berarti setiap simbol variabel akan diturunkan menjadi terminal sampai tidak ada yang bisa tergantikan.Sebagai contoh, terdapat CFG dengan aturan produksi sebagai berikut dengan simbol awal S :

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

Maka, jika ingin dicari gambar pohon penurunan dengan string ‘aabbb’ hasilnya adalah seperti di bawah:



Proses penurunan / parsing bisa dilakukan dengan cara sebagai berikut:

- Penurunan terkiri (leftmost derivation): simbol variabel terkiri yang diperluas terlebih dahulu.
- Penurunan terkanan (rightmost derivation): simbol variabel terkanan yang diperluas terlebih dahulu.

Misalkan terdapat grammar sebagai berikut:

$$S \rightarrow aAS \mid a$$

$$A \rightarrow SbA \mid ba$$

Untuk memperoleh string ‘aabbaa’ dari grammar di atas dilakukan dengan cara:

- Penurunan terkiri: S => aAS => aSbAS => aabAS => aabbaS => aabbaa
- Penurunan terkanan : S => aAS => aAa => aSbAa => aAbbaa => aabbaa

Ambiguitas terjadi bila terdapat lebih dari satu pohon penurunan yang berbeda untuk memperoleh suatu string. Misalkan terdapat tata bahasa sebagai berikut:

$$S \rightarrow A \mid B$$

$$A \rightarrow a$$

$$B \rightarrow a$$

Untuk memperoleh untai ‘a’ bisa terdapat dua cara penurunan sebagai berikut:

$$S => A => a$$

$$S => B => a$$

Sebuah string yang mempunyai lebih dari satu pohon sintaks disebut string ambigu (ambiguous). Grammar yang menghasilkan paling sedikit sebuah string ambigu disebut grammar ambigu.

III. Penjelasan Syntax JavaScript dalam Pembuatan FA dan CFG

Dalam pembuatan CFG dan FA, ada beberapa peraturan per-syntax-an yang perlu diperhatikan terkait penamaan maupun struktur program yang dirinci sebagai berikut.

A. Variabel

Variabel digunakan untuk menyimpan data, seperti string teks, angka, dll. Data atau nilai yang disimpan dalam variabel dapat diatur, diperbarui, dan diambil kapanpun diperlukan. JavaScript dapat membuat variabel dengan kata kunci var, let, dan cons, sedangkan operator penugasan (=) digunakan untuk menetapkan nilai ke variabel, seperti ini: var varName = value;

Berikut adalah aturan untuk memberi nama variabel JavaScript:

- Nama variabel dimulai dengan huruf, garis bawah (_), atau tanda dolar ($).
- Nama variabel tidak boleh dimulai dengan angka.
- Nama variabel hanya boleh berisi karakter alfanumerik (A-z, 0-9) dan garis bawah.
- Nama variabel tidak boleh berisi spasi.
- Nama variabel tidak boleh berupa keyword JavaScript atau kata JavaScript reserved word.

B. Operator

Dalam JavaScript, operator adalah simbol khusus yang digunakan untuk melakukan operasi pada operan (nilai dan variabel). Sebagai contoh,

```
2+3;  //5
```

JavaScript mencakup kategori operator berikut.

- Operator Aritmatika

Operator	Deskripsi
+	Penjumlahan
-	Pengurangan
*	Perkalian
/	Pembagian
%	Modulus (sisa bagi)
**	Eksponensial
++	Increment
--	Decrement

- Operator Perbandingan

Operator	Deskripsi
==	Nilai Sama dengan
===	Nilai dan tipe data sama dengan
!=	Tidak sama dengan
!==	Nilai dan tipe data tidak sama dengan
>	Lebih besar dari
<	Lebih kecil dari
>=	Lebih besar dari atau sama dengan
<=	Lebih kecil dari atau sama dengan

- Operator Logika

Operator	Deskripsi
&&	And (dan)
||	Or (atau)
!	Not (tidak)

- Operator Penugasan

Operator	Contoh	Sama Dengan
=	x = y	x = y
+=	x += y	x = x + y
-=	x -= y	x = x - y
*=	x *= y	x = x * y
/=	x /= y	x = x / y
%=	x %= y	x = x % y
**=	x **= y	x = x ** y

- Operator Terner

  kondisi ? jikaBenar : jikaSalah;

C. Komentar

JavaScript mendukung komentar single-line serta multi-line. Komentar single-line dimulai dengan garis miring ganda (//), diikuti oleh teks komentar. Berikut contohnya:

```
// This is my first JavaScript program
```

Sedangkan, komentar multi-line dimulai dengan garis miring dan tanda bintang (/ *) dan berakhir dengan tanda bintang dan garis miring (* /). Berikut adalah contoh komentar multi-line:

```
/* This is my first program
   in JavaScript */
```

D. Conditional

Anda dapat menggunakan pernyataan bersyarat dalam kode Anda untuk melakukan ini.

Dalam JavaScript kami memiliki pernyataan bersyarat berikut:

- Gunakan if untuk menentukan blok kode yang akan dieksekusi, jika kondisi yang ditentukan benar
- Gunakan else untuk menentukan blok kode yang akan dieksekusi, jika kondisi yang sama salah
- Gunakan else if untuk menentukan kondisi baru yang akan diuji, jika kondisi pertama salah

E. Array

Array adalah jenis objek yang digunakan untuk menyimpan banyak nilai dalam variabel tunggal. Setiap nilai (juga disebut elemen) dalam array memiliki posisi numerik, yang dikenal sebagai indeksnya, dan mungkin berisi data dari semua tipe-angka data, string, boolean, fungsi, objek, dan bahkan array lainnya. Indeks array dimulai dari 0, sehingga elemen array pertama adalah arr[0] bukan arr[1].

Cara paling sederhana untuk membuat array adalah dengan menentukan elemen array sebagai daftar yang dipisahkan koma (comma-separated) yang diapit oleh tanda kurung siku, seperti yang ditunjukkan dalam contoh di bawah ini:

```
var colors = ["Red", "Yellow", "Green",
"Orange"];
var cities = ["London", "Paris", "New York"];
```

F. While

Pernyataan while membuat perulangan yang mengeksekusi pernyataan tertentu selama kondisi pengujian bernilai benar. Kondisi dievaluasi sebelum mengeksekusi pernyataan. Pengulangan while dapat ditulis sebagai berikut:

```
while (condition){
  statement
}
```

G. Break

Break adalah kondisi dimana ia akan memberhentikan switch maupun loop dalam JavaScript.

H. Class

Class merupakan template untuk membentuk objek. Class dapat diartikan ekspersi fungsi dan deklarasi fungsi, dengan class memiliki dua komponen yaitui, ekspresi class dan deklarasi class. Untuk mendeklarasikan class, menggunakan kata kunci class dan diikuti dengan nama class yang ingin dibuat. Sedangkan class ekpresi dapat diberi nama ataupun tidak. Membuat class dalam JavaScript selalu ditambahkan constructor{} ke dalam setiap indentasi seperti contoh berikut:

```
class ClassName {
  constructor() { ... }
}
```

I. Function

Fungsi JavaScript didefinisikan dengan kata kunci fungsi, diikuti dengan nama, diikuti dengan tanda kurung ().Nama fungsi dapat berisi huruf, angka, garis bawah, dan tanda dolar (aturan yang sama dengan variabel). Tanda kurung dapat menyertakan nama parameter yang dipisahkan dengan koma:

(parameter1, parameter2, ...)

Kode yang akan dieksekusi, oleh fungsi, ditempatkan di dalam kurung kurawal: {}

J. For Loop

Dalam JavaScript, terdapat tiga jenis pengulangan, yaitu for, for in dan for of. Masing-masing memiliki fungsi yang berbeda.

A for loop berulang sampai kondisi tertentu bernilai false. JavaScript for loop mirip dengan Java dan C for loop.

Pernyataan for terlihat seperti berikut:

```
for ([initialExpression]; [conditionExpression];
[incrementExpression])
  statement
```

Pernyataan for...in mengiterasi variabel tertentu pada semua properti enumerable dari suatu objek. Untuk setiap properti yang berbeda, JavaScript mengeksekusi pernyataan yang ditentukan. Pernyataan for...in terlihat seperti berikut:

```
for (variable in object)
  statement
```

Pernyataan for...of membuat loop Iterasi pada objek yang dapat diubah (termasuk Array, Peta, Set, objek argumen, dan seterusnya), memanggil pengait iterasi khusus dengan pernyataan yang akan dieksekusi untuk nilai setiap properti yang berbeda.

```
for (variable of object)
  statement
```

K. Ekspresi

Ekspresi dalam JavaScript adalah kombinasi variabel, nilai, dan operator, yang dihitung menjadi nilai. Kami menyebut perhitungan ini sebagai evaluasi. Oleh karena itu, itu menjadi ungkapan. Misalnya, ekspresi 2 * 2 akan bernilai 4, seperti pada contoh di bawah ini:

```
var hasil = 7*2
```

Beberapa ekspresi dapat menyertakan variabel juga. Namun, ini tidak wajib – Anda tidak harus menyertakannya jika tidak membutuhkannya.

```
a * 8
n * 8
```

Nilai juga dapat berupa gabungan variabel dan string. Misalnya, ekspresi “Joe” + “” + “Johnson”, akan dievaluasi menjadi “Joe Johnson”:

```
"Jason" + " " + "Doe"
```

BAB III

IMPLEMENTASI DAN PENGUJIAN

I. IMPLEMENTASI PROGRAM

Program yang dibuat oleh kelompok kami terdiri dari file .txt yang digunakan untuk membuat Context Free Grammar (CFG). CFG yang dibuat dari kelompok kami pada bagian awalnya akan menginisiasi state awal yang ditandai dengan huruf S. Dari state awal ini kemudian akan terhubung menuju state-state lain dimulai dari state controlflow. State ini memiliki fungsi untuk membagi-bagi state yang diperlukan untuk membuat parser dari program. Parser dari bahasa Javascript yang dibuat antara lain untuk kasus kondisional (if dan else), perulangan (while dan for), boolean (true dan false), kondisi kosong (null), pembuatan tipe variabel (const, let, dan var), kondisi percabangan (switch, case, dan default), procedure dan fungsi (function), error-handling (try, catch, finally, dan throw), kondisi untuk melanjutkan dan menghentikan jalan program (continue dan break), dan untuk melakukan penghapusan pada item dari object (delete).

Setelah grammar dari program sudah dibuat, kelompok kami juga membuat token yang berfungsi untuk menerjemahkan grammar yang telah dibuat. Token yang dibuat oleh kelompok kami antara lain sebagai berikut:

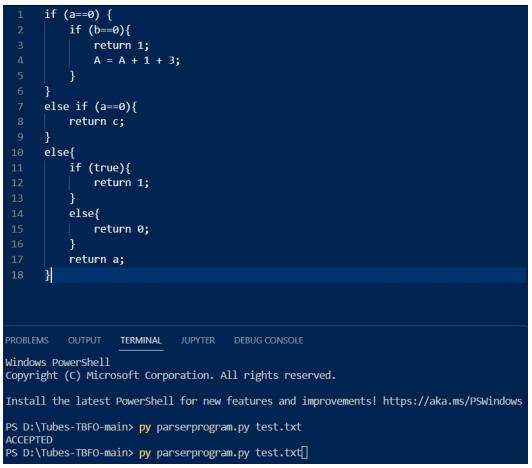
- \n -> NEWLINE (untuk mengganti baris)
- (-> KURUNG_DEPAN
-) -> KURUNG_BELAKANG
- [-> KURUNG_SIKU_DEPAN
-] -> KURUNG_SIKU_BELAKANG
- { -> KURUNG_KURAWAL_DEPAN
- } -> KURUNG_KURAWAL_BELAKANG
- ; -> SEMICOLON
- : -> COLON
- + -> ADD
- - -> SUB

- * -> MUL
- / -> DIV
- % -> MOD
- <= -> LEQ (Less Equal)
- < -> L (Less)
- >= -> GEQ (Greater Equal)
- > -> G (Greater)
- != -> NEQ (Not Equal)
- \== -> ISEQ (Is Equal)
- = -> EQUALS
- and -> AND
- or -> OR
- not -> NOT
- if -> IF
- else -> ELSE
- try -> TRY
- catch -> CATCH
- finally -> FINALLY
- while -> WHILE
- for -> FOR
- continue -> CONTINUE
- break -> BREAK
- true -> TRUE
- false -> FALSE
- none -> NONE
- function -> FUNCTION
- return -> RETURN
- . -> DOT
- , -> COMMA
- var -> VAR
- const -> CONST
- let -> LET
- throw -> THROW

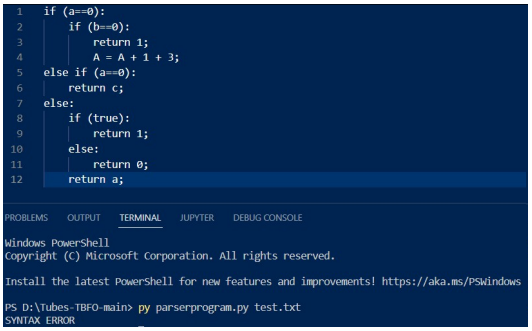
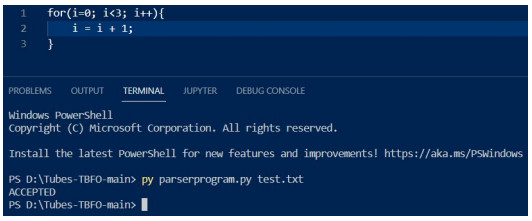
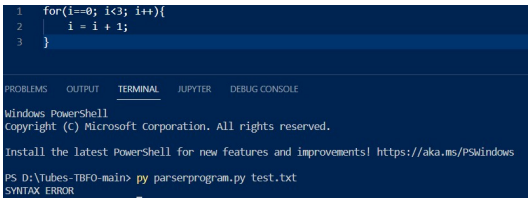
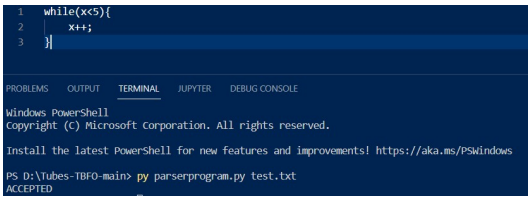
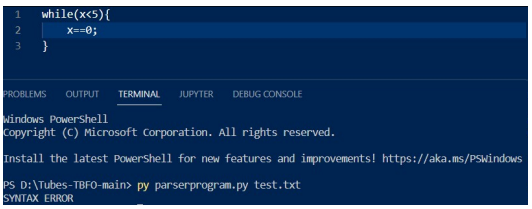
- default -> DEFAULT
- delete -> DELETE
- switch -> SWITCH
- case -> CASE
- null -> NULL

Dari berbagai macam token yang ada ini, kelompok kami akan membuat algoritma untuk melakukan penerjemahan kedalam bentuk CNF (Chomsky Normal Form). Setelah bentuk CNF terdefinisi, program akan melakukan pengecekan pada nama-nama variabel dan operasi yang telah diterjemahkan sebelumnya dan melakukan pengecekan pada token yang telah didefinisikan pada programnya. Pengecekan token dilakukan dengan memanfaatkan proses Regular Expression yang akan menghasilkan sebuah program. Pengecekan token ini kemudian akan dicocokan pada file Javascript test terhadap grammar yang telah didefinisikan. Jika sesuai, maka program akan menampilkan pesan ‘Accepted’ dan jika tidak sesuai, program akan menampilkan pesan ‘Syntax Error’.

II. HASIL PENGUJIAN



Hasil pengujian test case yang benar adalah “ACCEPTED”

	Hasil pengujian test case yang segaja kami salahkan adalah “SYNTAX ERROR”
	Hasil pengujian test case yang benar adalah “ACCEPTED”
	Hasil pengujian test case yang segaja kami salahkan adalah “SYNTAX ERROR”
	Hasil pengujian test case yang benar adalah “ACCEPTED”
	Hasil pengujian test case yang segaja kami salahkan adalah “SYNTAX ERROR”

BAB IV
KESIMPULAN DAN SARAN

I. KESIMPULAN

Kesimpulan dari tugas besar yang kami kerjakan adalah bahwa Mata kuliah Teori Bahasa Formal dan Otomata sangat bermanfaat dalam membentuk compiler suatu bahasa pemrograman. Compiler pada dasarnya dibentuk dari sekumpulan grammar yang disusun menggunakan Context Free Grammar (CFG) dan kemudian diubah menjadi Chomsky Normal Form(CNF) serta diolah menggunakan Cocke-Younger Kasami (CYK).

II. SARAN

Ada beberapa saran dari kami berdasarkan implementasi program dan hasil pengujian yang telah dilakukan, yaitu sebagai berikut:

1. Gunakan Algoritma CYK pada implementasi program compiler karena dokumentasi yang dapat dijadikan referensi untuk algoritma ini sudah sangat banyak.
2. Implementasikan algoritma yang lebih cepat dan efisien agar waktu yang dibutuhkan untuk mengevaluasi source code tidak terlalu besar.

REFERENSI

“Cocke-Younger-Kasami (CYK) Algorithm

https://www.geeksforgeeks.org/cocke-younger-kasami-cyk-algorithm/

https://www.programiz.com/javascript

Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2007). Context Free Grammars and Languages. In Introduction to automata theory, languages, and computation. essay, Pearson/Addison Wesley.

PEMBAGIAN TUGAS

NAMA	TUGAS
Jason Rivalino	Grammar, laporan
Muhammad Syauqi Jannatan	CYK, laporan
Agsha Athalla Nurkareem	Grammar, membuat token untuk parsing

LAMPIRAN

Pranala Github: https://github.com/jasonrivalino/TubesTBFO-AgeusJayyUkii