Laporan Tugas Besar

IF 2124 Teori Bahasa Formal dan Otomata

Parser Bahasa JavaScript (Node.js)

Kelompok 13 - random

Ahmad Ghulam Ilham	13521118
Muhammad Dhiwaul Akbar	13521158
Muhammad Habibi Husni	13521169

Teknik Informatika

Institut Teknologi Bandung

Semester I Tahun 2022/2023

BAB I

Teori Dasar

Finite Automata

Finite Automata (FA) adalah mesin abstrak sederhana yang digunakan untuk membaca pola / string input. Suatu FA memiliki state dan aturan tertentu untuk berpindah dari satu state ke state lainnya, bergantung pada input symbol yang dibaca. FA terbagi menjadi dua jenis, yaitu:

1. Deterministic Finite Automata (DFA), sebuah FA yang terdiri atas lima bagian seperti berikut.

   A DFA is a quintuple

   $$A = (Q, \Sigma, \delta, q_0, F)$$

   - Q is a finite set of *states*
   - Σ is a *finite alphabet* (=input symbols)
   - δ is a *transition function* $(q, a) \mapsto p$
   - $q_0 \in Q$ is the *start state*
   - $F \subseteq Q$ is a set of *final states*

2. Nondeterminisitic Finite Automata (NFA), suatu FA yang terdiri atas lima bagian seperti berikut.

Formally, a NFA is a quintuple

$$A = (Q, \Sigma, \delta, q_0, F)$$

- Q is a finite set of states
- Σ is a finite alphabet
- δ is a transition function from $Q \times \Sigma$ to the *powerset* of Q
- $q_0 \in Q$ is the *start state*
- $F \subseteq Q$ is a set of *final states*

Perbedaan antara DFA dan NFA terletak pada aturan state automata. Pada DFA, state tidak boleh beririsan sehingga state hanya bisa berpindah dari satu state tepat ke satu state lain. Sedangkan, pada NFA state boleh beririsan sehingga perpindahan state bisa "dipilih" dan state dapat terdiri atas kumpulan state. Berikut adalah contoh FA.

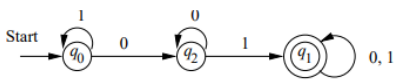
Example: An automaton A that accepts

$L = \{x01y : x, y \in \{0,1\}^*\}$

The automaton $A = (\{q_0, q_1, q_2\}, \{0,1\}, \delta, q_0, \{q_1\})$
as a *transition table*:

	0	1
$\rightarrow q_0$	q_2	q_0
$\star q_1$	q_1	q_1
q_2	q_2	q_1

The automaton as a *transition diagram*:



Context Free Grammar

Context Free Grammar (CFG) adalah grammar yang digunakan untuk menghasilkan semua string yang dapat dibentuk dalam suatu language. Suatu CFG terdiri atas empat bagian seperti berikut.

A *context-free grammar* is a quadruple

$$G = (V, T, P, S)$$

where

V is a finite set of *variables*.

T is a finite set of *terminals*.

P is a finite set of *productions* of the form $A \to \alpha$, where A is a variable and $\alpha \in (V \cup T)^*$

S is a designated variable called the *start symbol*.

Menurunkan sebuah string dimulai dari start symbol. Dengan menggunakan production, start symbol dapat diturunkan terus-menerus menjadi variabel-variabel lain. Menggunakan production, variabel-variabel akan diturunkan sampai string hanya berisi terminal. Berikut adalah contoh CFG.

The expressions are defined by the grammar

$$G = (\{E, I\}, T, P, E)$$

where $T = \{+, *, (,), a, b, 0, 1\}$ and P is the following set of productions:

1. $E \to I$
2. $E \to E + E$
3. $E \to E * E$
4. $E \to (E)$
5. $I \to a$
6. $I \to b$
7. $I \to Ia$
8. $I \to Ib$
9. $I \to I0$
10. $I \to I1$

Syntax JavaScript

JavaScript adalah bahasa bahasa pemrograman yang bersifat lightweight dan dynamic. JavaScript digunakan untuk membuat clisent-side dynamic pages. Bahasa JavaScript adalah bahasa yang open-source dan cross-platform. Syntax JavaScript merupakan aturan-aturan dalam bahasa JavaScript tentang konstruksi atau penulisan dalam suatu program JavaScript.

Syntax JavaScript memiliki dua definisi value, yaitu fixed values dan variable values. Fixed value disebut sebagai “Literals”, sedangkan variable values disebut sebagai “Variables”.

Aturan penulisan fixed values di antaranya adalah numbers dapat ditulis dengan atau tanpa menggunakan desimal, sedangkan strings adalah teks yang ditulis dalam tanda petik atau kutip.

JavaScript menggunakan keywords “var”, “let”, dan “const” untuk mendeklarasi suatu variabel. Tanda “=” digunakan untuk meng-assign nilai suatu variabel. Untuk melakukan perhitungan, JavaScript menggunakan operator +, -, *, /. Dalam JavaScript, expression adalah gabungan values, variables, dan operators untuk menghasilkan value.

Komentar dalam bahasa JavaScript dilakukan dengan manmbahkan // di depan komentar atau meletakkan komentar di antara tanda /* dan */. Dalam bahasa JavaScript terdapat identifiers dalam penamaan variabel, keyword, dan fungsi. Aturan penamaan yang legal / valid dalam JavaScript sebagai berikut.

1. Dimulai dengan huruf alfabet
2. Dimulai dengan dollar sign ($)
3. Dimulai dengan underscore (_)

Numbers tidak dapat digunakan sebagai karakter pertama dalam penamaan agar JavaScript dapat membedakan antara identifier dan number. Identifier JavaScript bersifat case sensitive. Untuk menggabungkan dua kata atau lebih sebagai nama suatu variabel tidak boleh menggunakan tanda - (hyphen) karena tanda tersebut akan dibaca sebagai operator pengurangan.

BAB II

Hasil FA dan CFG

2.1. Finite Automata

Simbol	Deskripsi / Keterangan

2.2. Context Free Grammar

Simbol	Deskripsi / Keterangan
CASE	
OPERAND	
BODY	
STATEMENTS	
DECLARE	
VAR	
VARC	
EKSPRESI	
OBJ	
ASSIGNMENT	
INCDEC	
RETURN	
STRING	
SENTENCE	
WORD	

NUMBER	
INTEGER	
INTEGERBACK	
FLOAT	
FLOATBACK	
WHILE	
KONDISI	
CONST	
CONSTVAL	
ARRAY	
ARRAYCOMP	
ARRAYAWAL	
ARRAYVAL	
ARRAYACCESS	
FUNCTION	
PARAMETERS	
PARAMETER	
PARAMAWAL	
PARAMVAR	
TRY	
CATCH	
FOR	
DELETE	

IF	
ELSE	
DO	
SWITCH	
ALPHABET	
UPPERCASE	
LOWERCASE	
OPERATOR	
COMPARE	
SYMBOL	
DIGIT	
DIGIT2	
BREAKW	
CONSTW	
CASEW	
CATCHW	
CONTINUEW	

BAB III

Implementasi dan Pengujian

Hasil Implementasi

3.1. CFG.py

Struktur Data / Fungsi / Prosedur	Deskripsi / Keterangan
produksi	Variabel bertipe dictionary. Digunakan untuk menyimpan produksi CFG.
variabel	Variabel bertipe set. Dingunakan untuk menyimpan variabel dari produksi CFG.
terminal	Variabel bertipe set. Digunakan untuk menyimpan terminal dari produksi CFG.
readCFG(self,filepath)	Prosedur untuk membaca file .txt berisi grammar CFG dan menghasilkan produksi, variabel, dan terminal CFG sesuai isi file .txt.
print(self)	Prosedur untuk meng-output hasil yang didapat dari fungsi readCFG(self,filepath).

3.2. CFGtoCNF.py

Struktur Data / Fungsi / Prosedur	Deskripsi / Keterangan

3.3. Cyk_pharser.py

Struktur Data / Fungsi / Prosedur	Deskripsi / Keterangan

3.4. main.py

Struktur Data / Fungsi / Prosedur	Deskripsi / Keterangan

3.5. test.py

Struktur Data / Fungsi / Prosedur	Deskripsi / Keterangan

Hasil Pengujian

3.1. Kasus Uji 1

```
function do_something(x) {
    // This is a 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";
    }
}
```

3.2. Kasus Uji 2

3.3. Kasus Uji 3

BAB IV

Repository GitHub dan Pembagian Tugas

Repository GitHub

https://github.com/Agilham/Tubes-TBFO-Parser.git

Pembagian Tugas

NIM	Nama	Tugas
13521118	Ahmad Ghulam Ilham	CFG reader
13521158	Muhammad Dhiwaul Akbar	CYK parser
13521169	Muhammad Habibi Husni	CFG to CNF, FA

Tabel 4.1. Pembagian Tugas