Simulator Pengenal String Yang Diterima Sebuah Deterministic Finite Automata (DFA)

Suparyanto, Selo
Program Studi S2 Teknik Elektro
Konsentrasi Teknologi Informasi
Departemen Teknik Elektro dan Teknologi Informasi
Jl. Grafika No. 2 Yogyakarta- 55281
anto_suparyanto2@yahoo.com, selo@ugm.ac.id

Abstract—**This The DFA simulator is a device that can be used to check string inputs in a Deterministic Finite Automata (DFA) before a string is inputted into a DFA. In making this DFA simulator, it is determined first a DFA machine transition diagram with a number of states, then made the transition table, from the diagram and transition table is then created algorithm using multiple linked list data structure. Finally, an algorithm for string checking is entered into the simulator, taking into account the initial state and the final state. The string reading starts from the initial state, traces the character characters of the read string, matched to the received input on each state, when the read character does not end in the final state, the string is not accepted by the DFA, otherwise if the search ends in the final state, Then the read string is accepted by DFA.**
Keywords: Simulator, DFA, transition diagram, transition – table; algorithm; state; initial state; final state; string

Intisari – *Simulator DFA* adalah sebuah perangkat yang bisa dipergunakan untuk melakukan pengecekan terhadap inputan-inputan string pada sebuah *Deterministic Finite Automata (DFA)* sebelum sebuah string di inputkan kedalam sebuah *DFA*. Dalam pembuatan *simulator DFA* ini, ditentukan terlebih dahulu sebuah diagram transisi mesin *DFA* dengan sejumlah *state*, kemudian dibuat tabel transisinya, dari diagram dan tabel transisi tersebut selanjutnya dibuat algoritma dengan menggunakan struktur data multiple *linked list*. Terakhir dibuat algoritma untuk pengecekan string yang dimasukkan kedalam simulator tersebut, dengan memperhatikan state awal dan *state* akhirnya. Pembacaan string dimulai dari *state* awal, ditelusuri karakter perkarakter dari string yang dibaca, dicocokkan dengan input yang di terima pada setiap *state*, apabila karakter dibaca tidak berakhir di *state* akhir, berarti string tersebut tidak diterima oleh *DFA*, sebaliknya jika penelusuran berakhir di state akhir, maka string yang dibaca tersebut diterima oleh *DFA*.

Kata kunci – *Simulator*; *DFA*; diagram transisi; tabel transisi; algoritma; *state*; *state* awal; *state* akhir; string

I. PENDAHULUAN

Beberapa penelitian saat ini yang popular mengenai *DFA* adalah tentang cara untuk menstabilkan *Deterministic Finite Automata* menggunakan pendekatan dengan *Matrix*, dimana dari suatu *DFA* bisa dibangun suatu matriks untuk membuat sebuah *DFA* yang lebih efisien dan stabil untuk menjaga keseimbangan state-nya, sedangkan untuk meminimalkan panjang lintasan bisa dipergunakan *feedback* sebagai pengontrol state [2].

Selain itu diperkenalkan sebuah konsep baru tentang *DFA* yaitu *Deterministic Finite Automata* berbasis fungsional *PRE proxy re-enkripsi* (*DFA-based FPRE*). Dalam konsep ini, pesan dienkripsi dalam *ciphertext* terkait dengan indeks panjang string, dan hanya bisa melakukan *decryptor* jika dan hanya jika DFA yang berhubungan dengan kunci rahasia yang menerima string. Selanjutnya, enkripsi di atas diperbolehkan untuk diubah ke *ciphertext* lain terkait dengan string baru dengan *proxy semitrusted* kepada siapa kunci re-enkripsi diberikan [4].

Sedangkan untuk mengatasi masalah ledakan state yang disebabkan oleh kompilasi gabungan aturan ekspresi reguler yang mengandung ". *" Dalam kondisi tertentu, bisa menggunakan struktur *DFA* baru berdasarkan beberapa dimensi kubus, yaitu algoritma *MDC-DFA* (*A Multi-dimensional Cube Deterministic Finite Automata-Based Feature Matching Algorithm*). Algoritma ini membagi keadaan redundan dengan dimensi dan mengompresnya ke sumbu[5].

Setiap inputan yang diinputkan ke dalam sebuah *DFA* harus menjalani pengecekan ke setiap *state* pada sebuah *DFA* untuk mengetahui apakah mencapai *state* akhir atau tidak, yang mengindikasikan sebuah input dinyatakan diterima atau tidak oleh *DFA* tersebut. Karena hal itu maka akan mengakibatkan mesin otomata tersebut dalam hal ini *DFA* akan banyak mengalami waktu yang terbuang untuk menerima semua input yang belum tentu diterima oleh *DFA* tersebut [3].

Dari permasalahan tersebut maka bisa dibuat sebuah simulasi yang bisa mengecek apakah input tersebut diterima oleh sebuah *DFA* atau tidak, tanpa dinputkan ke dalam *DFA* yang sesungguhnya.

Penelitian ini memberikan kontribusi :

1. Menghasilkan simulator yang membantu kerja sebuah *DFA*, sehingga bisa menghemat waktu dan biaya dalam memanfaatkan sebuah *DFA*.
2. Memberikan manfaat dapat menyeleksi inputan yang diterima oleh sebuah *DFA*, sehingga *DFA* tidak perlu menerima inputan-inputan yang jelas tidak diterima *DFA*.

II. LANDASAN TEORI

A. *Automata* (Otomata)

Otomata adalah mesin abstrak yang dapat mengenali (*recognize*), menerima (*accept*), atau membangkitkan (*generate*) sebuah kalimat dalam bahasa tertentu.

Beberapa istilah yang berkaitan dengan otomata

- Simbol adalah sebuah *entitas* abstrak (seperti halnya pengertian titik dalam geometri). Sebuah huruf atau sebuah angka adalah contoh simbol.
- String adalah deretan terbatas (*finite*) simbol-simbol. Sebagai contoh, jika a, b, dan c adalah tiga buah simbol maka abcb adalah sebuah string yang dibangun dari ketiga simbol tersebut.
- Jika w adalah sebuah string maka panjang string dinyatakan sebagai [w] dan didefinisikan sebagai cacahan (banyaknya) simbol yang menyusun string tersebut. Sebagai contoh, jika w = abcb maka [w]= 4.
- String hampa adalah sebuah string dengan nol buah simbol. String hampa dinyatakan dengan simbol e (atau ^) sehingga [e]= 0. String hampa dapat dipandang sebagai simbol hampa karena keduanya tersusun dari nol buah simbol.
- Alfabet adalah himpunan hingga (*finite set*) simbol-simbol

B. *Finite State Automata* (*FSA*)

Adalah suatu mesin otomata dalam bentuk model matematika dari suatu sistem yang memiliki sejumlah *state* (kedudukan) tak berhingga banyaknya dan dapat berpindah dari suatu *state* ke *state* lain dengan suatu fungsi transisi [1].

Merupakan jenis otomata yang tidak memiliki penyimpanan, sehingga kemampuan mengingatnya terbatas.

Mekanisme kerja suatu *Finite State Automata* dapat diaplikasikan pada *analisis leksikal*, *text editor*, protokol komunikasi jaringan.

Secara formal *finite state automata* dinyatakan oleh 5 tupel atau M=(Q, Σ, δ, S, F), di mana :

Q = himpunan *state* / kedudukan
Σ = himpunan simbol *input* / masukan / abjad
δ = fungsi transisi
S = *state* awal / kedudukan awal (*initial state*)
F = himpunan *state* akhir

Berdasarkan pendefinisian kemampuan berubah state-statenya, *Finite State Automata* (*FSA*) bisa dikelompokkan :

- Otomata berhingga deterministic (*Deterministik Finite Automata/DFA*)
- Otomata berhingga non deterministik (*Non Deterministik Finite Automata/NFA/NDFA*)

C. *Deterministik Finite Automata* (*DFA*)

DFA adalah jenis *Finite State Automata* yang memiliki tepat satu *state* berikutnya untuk setiap symbol masukan yang diterima suatu state.

- *Deterministik Finite Automata* (*NFA*) adalah otomata berhingga yang pasti arahnya
- untuk setiap pasangan *state input*, bisa memiliki 0 (nol) atau satu pilihan untuk state berikutnya
- Untuk setiap state selalu tepat ada satu *state* berikutnya untuk setiap simbol *input* yang ada
- Dari suatu *state* bisa terdapat 0,1 atau lebih busur keluar (transisi) berlabel simbol *input* yang sama.
- *DFA* didefinisikan dengan 5 tupel M= (Q,Σ, δ,S,F)
- Suatu string x dinyatakan diterima oleh bahasa DFA, M= (Q, Σ, δ, S, F) bila {x | δ (S,x) memuat sebuah *state* di dalam F}
- DFA dari suatu state bisa terdapat 0 atau 1 busur keluar (transisi).
- Pada fungsi transisi setiap pasangan *state input* bisa mempunyai 0 atau satu pilihan untuk *state* berikutnya.

III. METODOLOGI

Tahap Persiapan

1. Kajian pustaka, yaitu mencari referensi dan mempelajari buku, artikel, dan literatur internet yang berhubungan dengan topik penelitian dan penelitian penelitian terdahulu yang berkaitan dengan pemanfaatan algorithma dalam *Finite State Automata* terutama untuk jenis *Deterministik Finite Automata* (*DFA*)
2. Identifikasi permasalahan
3. Menyiapkan Diagram transisi mesin *Deterministik Finite Automata* (*DFA*)
4. Membuat tabel transisi dari Diagram transisi mesin *Deterministik Finite Automata* (*DFA*)

Tahap Pelaksanaan

1. Menganalisis diagram transisi dan tabel transisi dari mesin *DFA* yang sudah dilakukan pada tahap persiapan
2. Membuat representasi struktur data dengan menggunakan struktur data multiple linked list
3. Membuat algoritma untuk *simulator DFA*
4. Membuat algoritma untuk pembacaan sebuah string
5. Menganalisa hasil pembacaan string yang dibaca oleh *Deterministik Finite Automata* (*NFA*)
6. Memberikan *Output* apakah string yang dibaca,diterima oleh *Deterministik Finite Automata* (DFA) tersebut atau tidak.

Tahap Pengujian

a. Menguji sebuah inputan string kedalam simulasi *Deterministic Finite Automata* (*DFA*).
b. Menguji *validitas* hasil pembacaan string yang dibaca oleh *Deterministik Finite Automata* (*DFA*) secara *manual*
c. Menyimpulkan tingkat keberhasilan penggunaan simulator pembacaan atau penelusuran string yang dibaca oleh *simulator Deterministik Finite Automata* (*DFA*).

Tahap Pemakaian

- Sebelum di inputkan ke dalam sebuah *Deterministic Finite Automata* (*DFA*) sebuah string di inputkan dulu kedalam simulasi *Deterministic Finite Automata* (*DFA*).
- Menggunakan inputan inputan string ke *DFA* yang sudah pasti diterima mesin *DFA*.

IV. HASIL DAN PEMBAHASAN

A. HASIL

Contoh untuk hasil dengan input string "baabaab" adalah seperti pada gambar 1 dibawah.

```
TABEL TRANSISI DARI DFA
--------------------------
| Transisi  |  a  |  b  |
--------------------------
|    Q0     |  Q1 |  Q2 |
|    Q1     |  Q3 |  Q2 |
|    Q2     |  Q1 |  Q4 |
|    Q3     |  Q4 |  Q1 |
|    Q4     |  -  |  Q4 |
--------------------------

Input String  = baabaab
String baabaab diterima oleh DFA diatas
```

Gambar 1. Input string yang diterima *DFA*

Untuk membuktikan apakah hasil tersebut ditas benar atau salah, maka dilakukan pengujian secara manual dengan *input* string "baabaab"

δ(q0, baabaab) = δ (Q2, aabaab)
= δ (Q1, abaab)
= δ (Q3, baab)
= δ (Q1, aab)
= δ (Q3, ab)
= δ (Q4, b)
= Q4

Karena Q4 adalah anggota himpunan state akhir, maka string 'baabaab' diterima oleh DFA

Contoh lain dengan *input* string "ababab" adalah seperti pada gambar 2.

```
TABEL TRANSISI DARI DFA
--------------------------
| Transisi  |  a  |  b  |
--------------------------
|    Q0     |  Q1 |  Q2 |
|    Q1     |  Q3 |  Q2 |
|    Q2     |  Q1 |  Q4 |
|    Q3     |  Q4 |  Q1 |
|    Q4     |  -  |  Q4 |
--------------------------

Input String  = ababab
String ababab tidak diterima oleh DFA diatas
```

Gambar 2. Input string yang tidak diterima *DFA*

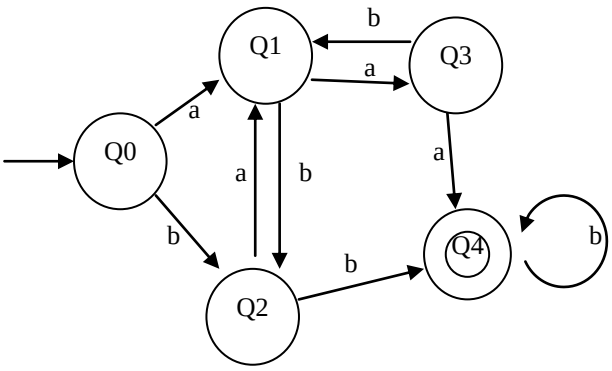
Untuk membuktikan apakah hasil tersebut ditas benar atau salah, maka dilakukan pengujian secara manual dengan *input* string "ababab"

δ(q0, ababab) = δ (Q1, babab)
= δ (Q2, abab)
= δ (Q1, bab)
= δ (Q2, ab)
= δ (Q1, b)
= Q2

Karena Q2 bukan anggota himpunan *state* akhir, maka string 'ababab' tidak diterima oleh *DFA*

B. PEMBAHASAN

Untuk penelitian dengan hasil diatas digunakan sebuah kasus dengan dDiagram transisi sebuah mesin *DFA* seperti pada gambar 1 berikut :



Gambar 3. Diagram transisi mesin *DFA*

Konfigurasi dari *Deterministic Finite Automata* (*DFA*) diatas secara formal dapat dinyatakan sebagai berikut :

Q = {Q0,Q1,Q2,Q3,Q4}
Σ = {a,b}
S = Q0
F ={Q4}

Sedangkan fungsi transisi yang dihasilkan berdasarkan digram transisi pada gambar 3 adalah :

δ (Q0,a) = Q1
δ (Q0,b) = Q2
δ (Q1,a) = Q3
δ (Q1,b) = Q2
δ (Q2,a) = Q1
δ (Q2,b) = Q4
δ (Q3,a) = Q4
δ (Q3,b) = Q2
δ (Q4,a) = Ø
δ (Q4,b) = Q4

selanjutnya bisa dijabarkan sebagai berikut :
Dari state Q0 jika mendapat input 'a' berpindah ke state Q1, dinyatakan : δ(Q0,a) = Q1

- Sehingga disebut *deterministic* (pasti arahnya) Tabel transisinya seperti pada table 1:

Tabel 1

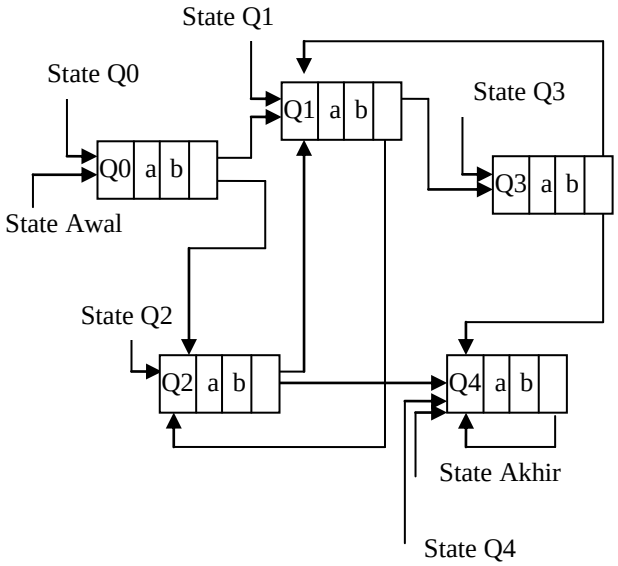
δ	a	b
Q0	Q1	Q2
Q1	Q3	Q2
Q2	Q1	Q4
Q3	Q4	Q1
Q4	-	Q4

- Suatu string diterima oleh DFA jika terdapat suatu urutan transisi sehubungan dengan input string tersebut dari *state* awal menuju ke *state* akhir.
- Misal, jika L(M) adalah bahasa yang diterima oleh *DFA* diatas. Maka bisa ditelusuri apakah string 'baabaab' termasuk dalam L(M), secara manual maka string tersebut bisa ditelusuri apakah bisa diterima oleh *DFA* atau tidak dengan cara sebagai berikut :

δ(q0, baabaab) = δ (Q2, aabaab)
= δ (Q1, abaab)
= δ (Q3, baab)
= δ (Q1, aab)
= δ (Q3, ab)
= δ (Q4, b)
= Q4

- Karena Q4 adalah anggota himpunan *state* akhir, maka string 'baabaab' diterima oleh DFA

Untuk penyelesaian diagram transisi seperti pada gambar dibuat struktur datanya menggunakan linked list seperti pada gambar 4.



Gambar 4. Struktur Data *linked list* mesin *DFA*

Algoritma untuk membuat diagram transisi DFA pada gambar diatas adalah sebagai berikut :

1. Mulai
2. Definisikan *structure* untuk menyimpan informasi state
   typedef struct linkedList
   {
   char Nama_state, input_1,input_2;
   linkedList *berikut;
   }linkedList;
   linkedList *awal, *akhir, *State_Q1,*State_Q2, *State_Q3, *State_Q4;
   *awal = State_Q 0;
   *akhir = State_Q4;
3. Buat *state* awal Q0 yang ditunjuk oleh *pointer State* Qo dan *State* Awal
4. Masukkan nama state dan input ke state tersebut
5. Jika nilai Q0->input_1 = "a", Q0->berikut = *State_Q1, lanjutkan ke langkah 6
   Jika nilai Q0->input_2 = "b", Q0->berikut = *State_Q2, lanjutkan ke langkah 7
   lanjutkan ke langkah 10
6. Jika nilai Q1>input_1 = "a", Q1->berikut = *State_Q3, lanjutkan ke langkah 8
   Jika nilai Q1->input_2 = "b", Q1->berikut = *State_Q2, lanjutkan ke langkah 7
   lanjutkan ke langkah 10
7. Jika nilai Q2->input_1 = "a", Q2->berikut = *State_Q1, kembali ke langkah 6
   Jika nilai Q2->input_2 = "b", Q2->berikut = *State_Q4, lanjutkan ke langkah 9
   lanjutkan ke langkah 10
8. Jika nilai Q3->input_1 = "a", Q3->berikut = *State_Q4, lanjutkan ke langkah 9
   Jika nilai Q3->input_2 = "b", Q3->berikut = *State_Q1, kembali ke langkah 6

lanjutkan ke langkah 10

9. Jika nilai Q4->input_1 = “a”, lanjutkan ke langkah 10
   Jika nilai Q4->input_2 = “b”, Q4->berikut = *State_Q4, kembali ke langkah 9
10. selesai

Sedangkan algoritma pembacaan string yang diinputkan kedalam mesin *DFA* seperti yang sudah dibuat diatas adalah sebagai berikut :

1. Masukkan string yang akan di baca oleh *DFA*
2. Dibaca mulai dari karakter pertama string oleh simpul yang ditunjuk oleh *state* awal, Q0,
   jika nilai Q0->a = a, Q0->*next* = *state* Q1, lanjutkan ke langkah 4
   jika nilai Q0->b = b, Q0->*next* = *state* Q2, lanjutkan ke langkah 3
   lanjutkan ke langkah 8
3. Baca karakter berikutnya
   jika nilai Q2->a = a, Q2->*next* = *state* Q1, lanjutkan ke langkah 4
   jika nilai Q2->b = b, Q2->*next* = *state* Q2, kembali ke langkah 3
   lanjutkan ke langkah 8
4. Baca karakter berikutnya
   jika nilai Q1->a = a, Q1->*next* = *state* Q3, lanjutkan ke langkah 5
   jika nilai Q1->b = b, Q1->*next* = *state* Q2, kembali ke langkah 3
   lanjutkan ke langkah 8
5. Baca karakter berikutnya
   jika nilai Q3->a = a, Q3->*next* = *state* Q4, lanjutkan ke langkah 6
   jika nilai Q3->b = b, Q3->*next* = *state* Q1, kembali ke langkah 4
   lanjutkan ke langkah 8
6. Baca karakter berikutnya
   jika nilai Q4->a = a, Q4->*next* = *state* Q3, kembali ke langkah 5
   jika nilai Q4->b = b, Q4->*next* = s*tate* Q2, kembali ke langkah 3
7. Cetak “String diterima oleh *DFA*”
8. Cetak “String tidak diterima oleh *DFA*”
9. Selesai

V. KESIMPULAN DAN SARAN

A. KESIMPULAN

Untuk sejumlah input string kedalam mesin DFA dapat disimpulkan bahwa algoritma yang dirancang dapat bekerja dengan baik, atau dengan kata lain algoritmanya valid.

B. SARAN

Kelemahan dari hasil penelitian ini, mesin automata masih ditentukan diawal, maka bisa dikatakan masih bersifat *statis*, karena masih untuk satu jenis mesin *automata* jenis *DFA*. Ditentukan terlebih dahulu mesin DFA nya, merepresentasikan struktur datanya dengan *multiple linked list*, membuat algoritma simulator dan membuat algoritma untuk membaca string bagi simulator.

Sehingga tidak bisa digunakan untuk sembarang mesin *DFA*.

Untuk penelitian selanjutnya disarankan untuk bisa dibuat yang bersifat dinamis, bisa bebas menggunakan sembarang mesin DFA, dengan memasukkan kedalam simulator.

REFERENCE

[1] M. Ihsan, Ilden Abi Neri, Hafda Bayu Nanda, “Penerapan Algoritma Depth First Search (DFS) Dinamis Untuk Menentukan Apakah Sebuah String Diterima Oleh Bahasa Reguler yang Didefinisikan Nondeterministic Finite Automata (NFA),” Makalah STMIK, tahun 2006.

[2] X. Xu, Yanqiong Zhang, Yiguang Hong, Matrix Approach to Stabilizability of Deterministic Finite Automata, American Control Conference (ACC) Washington, DC, USA, June 17-19, 2013

[3] A. Khalid_, Rajat Sen†, Anupam Chattopadhyay, SI-DFA: Sub-expression Integrated deterministic Finite Automata for Deep Packet Inspection, IEEE 14th International Conference on High Performance Switching and Routing

[4] Y. Li, Xingguo Luo, Xiangyu Shao, Dong Wei, “MDC-DFA:A Multi-dimensional Cube Deterministic Finite Automata-Based Feature Matching Algorithm “IEEE, 978-1-4673-7116-2/15, 2015

[5] K. Liang, Man Ho Au, *Member, IEEE*, Joseph K. Liu, Willy Susilo, *Senior Member, IEEE*, , Duncan S. Wong, *Member, IEEE*, Guomin Yang, *Member, IEEE*, Tran Viet Xuan Phuong, and Qi Xie, “A DFA-Based Functional Proxy Re-Encryption Scheme for Secure Public Cloud Data Sharing”,IEEE transactions on information forensics and security, VOL. 9, NO. 10, October 2014