Jurnal Sistem Komputer dan Kecerdasan Buatan
Vol III No.1 Tahun 2019

SIMULATOR STRING YANG DITERIMA *FINITE STATE AUTOMATA* (FSA) BERBASIS ANDROID

Azizah[1], Fauziah[2]

Program Studi Informatika, Universitas Nasional, Jalan Sawo Manila No. 61 Jakarta Selatan[1,2]

Azizahif99@gmail.com[1], fauziah@civitas.unas.ac.id[2]

Diterima 20 Agustus 2019
Disetujui 23 September 2019

***Abstract*- Automata is an abstract machine that can recognize (recognize), accept (accept), or generate a sentence in a particular language. There are two types of Deterministic Finite Automata (DFA) and Non-deterministic Finite Automata (NFA). Simulator Finite State Automata is a device that can be used to check the string input in Finite State Automata. The way the simulator works still does not know much about it, through the simulator it is hoped that users are more aware of it.**

Index Terms- ***dfa, nfa, string accepted, automata, android***

I. PENDAHULUAN

Teori automata sangat erat kaitannya dengan mesin-mesin abstrak. Bahasa yang dipakai untuk menginstruksikan computer disebut bahasa pemrograman. Bahasa yang dimengerti oleh mesin computer adalah intruksi dalam bahasa mesin (*Machine Language*) yang merupakan bahasa tingkat rendah, jadi bahasa tingkat tinggi yang kita sebutkan diatas agar dapat dimengerti oleh komputer haruslah diterjemahkan lebih dahulu oleh kompilator. [1]. Bahasa pemrograman dengan automata sangat berkaitan erat karena dengan Bahasa dapat menterjemahkan instruksi yang akan diterima oleh mesin abstrak (FSA). Dari jenisnya yaitu *finite state automata (FSA)* terdapat mesin bahasa yang berarti dapat mengenali, menerima dan menolak terdiri atas DFA (*Deterministic Finite Automata*) dan NFA (*Non-deterministic Finite Automata*) [2]. *Deterministic Finite Automata*/otomata berhingga deterministik, selanjutnya disebut sebagai DFA. Pada DFA, dari suatu *state* ada tepat satu *state* berikutnya untuk setiap simbol masukan yang diterima. Suatu string x dinyatakan diterima bila δ(S, x) berada pada *state* akhir/*final state*[3]. *Non-deterministic Finite Automata*, selanjutnya disebut sebagai NFA. Pada NFA, dari suatu *state* bisa terdapat 0 atau 1 atau lebih busur keluar (transisi) berlabel simbol *input* yang sama. Suatu string diterima oleh NFA bila terdapat suatu urutan transisi sehubungan dengan *input* string tersebut dari *state* awal menuju *state* akhir. Untuk NFA, maka harus mencoba semua kemungkinan yang ada sampai terdapat satu yang mencapai *state* akhir. Dan ada FSA output yang mana berbeda dengan jenis FSA sebelumnya (DFA & NFA), pada jenis ini tidak terdapat state menerima dan menolak, memiliki fungsi & himpunan output[4]. Simulasi adalah tiruan dari proses dunia nyata atau sistem. Simulasi menyangkut pembangkitan proses serta pengamatan dari proses untuk menarik kesimpulan dari sistem yang diwakili. Simulasi mempelajari atau memprediksi sesuatu yang belum terjadi dengan cara meniru atau membuat model sistem yang dipelajari[5].

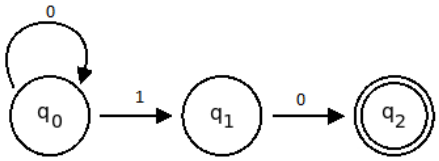
Mesin FSA menggunakan Simulator String yang diterima dan diimplementasikan pada aplikasi berbasis android. Simulator ini akan membantu *user* dalam mempelajari cara kerja dari mesin FSA.

II. METODOLOGI PENELITIAN

Metode yang digunakan yaitu dengan menggunakan studi literatur dan perancangan aplikasi *finite state*, yang mana jika diambil contoh kasus sebagai berikut :

Tuple M pada FSA jenis DFA maupun NFA diantaranya (Q,∑,δ,S,F) yang mengartikan untuk Q=himpunan *state*,

∑ =himpunan *input*, δ =fungsi transisi, S=*state* awal, F=*state* penerima. Disini sebagai contoh sederhana, kita gunakan FSA jenis NFA[6].



Gambar 1. Diagram Transisi NFA.

Dari diagram diatas, kita bisa melihat bahwa *tuple*nya sebagai berikut :

Q = (q0, q1, q2), ∑ = (0, 1), S=(q0), F=(q2)

Fungsi transisi = δ

δ(q0,0) = q0, δ(q0,1) = q1, δ(q1,0) = q2, δ(q1,1) = ϕ, δ(q2,0) = ϕ, δ(q2,1) = ϕ

Dari fungsi transisi tersebut didapat table transisi sebagai berikut :

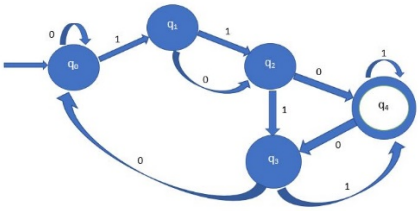
Table 1. Tabel Transisi

δ	0	1
q0	q0	q1
q1	q2	ϕ
q2	ϕ	ϕ

Sebagai contoh, kita *input* ‘0111’, maka untai akan bergerak dari *state* awal (q0) kemudian *input* ‘0’ pertama posisi tetap di (q0), lalu *input* kembali ‘1’ maka posisi akan berubah menjadi (q1), kemudian *input* kembali ‘1’ maka posisi akan berada di hampa (ϕ) dan hasilnya akan ditolak karena tidak berakhir di *state* penerima.

A. Perancangan Diagram *State* DFA

Untuk merancang simulator ini dibutuhkan diagram transisi untuk mengetahui cara kerja mesin FSA[7] .



Gambar 2. Diagram transisi DFA

Dari diagram diatas, kita bisa melihat bahwa *tuple*nya sebagai berikut :

Q = (q0, q1, q2, q3, q4), ∑ = (0, 1), S=(q0), F=(q4)

Fungsi transisi = δ

δ(q0,0) = q0, δ(q0,1) = q1, δ(q1,0) = q2, δ(q1,1) = q2, δ(q2,0) = q4, δ(q2,1) = q3, δ(q3,0) = q0, δ(q3,1) = q4, δ(q4,0) = q3, δ(q4,1) = q4

Dari fungsi transisi tersebut didapat table transisi sebagai berikut :

Table 2. Tabel Transisi

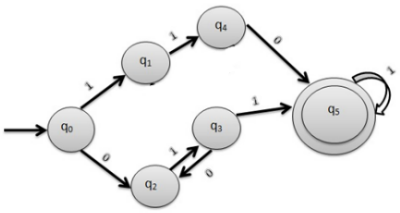
δ	0	1
q0	q0	q1
q1	q2	q2
q2	q4	q3
q3	q0	q4
q4	q3	q4

Sebagai contoh, kita *input* ‘0101’, maka untai akan bergerak dari *state* awal (q0) kemudian *input* ‘0’ pertama posisi tetap di (q0), lalu *input* kembali ‘1’ maka posisi akan berubah menjadi (q1), kemudian *input*

kembali ‘0’ maka posisi akan berubah menjadi (q2), kemudian *input* kembali ‘1’ maka posisi akan berubah menjadi (q3) dan hasilnya akan ditolak karena tidak berakhir di *state* penerima.

B. Perancangan Diagram *state* NFA

Untuk merancang simulator ini dibutuhkan diagram transisi untuk mengetahui cara kerja mesin NFA. Pada NFA bias memiliki himpunan hampa[8].



Gambar 3. Diagram transisi NFA

Dari diagram diatas, kita bisa melihat bahwa *tuple*nya sebagai berikut :

Q = (q0, q1, q2, q3, q4,q5), ∑ = (0, 1), S=(q0), F=(q5)

Fungsi transisi = δ

δ(q0,0) = q0, δ(q0,1) = q1, δ(q1,0) = ϕ, δ(q1,1) = q4, δ(q2,0) =ϕ, δ(q2,1) = q3, δ(q3,0) = q2, δ(q3,1) = q5, δ(q4,0) = q5, δ(q4,1) = ϕ, δ(q5,0) = ϕ, δ(q5,1) = q5.

Dari fungsi transisi tersebut didapat *table* transisi sebagai berikut :

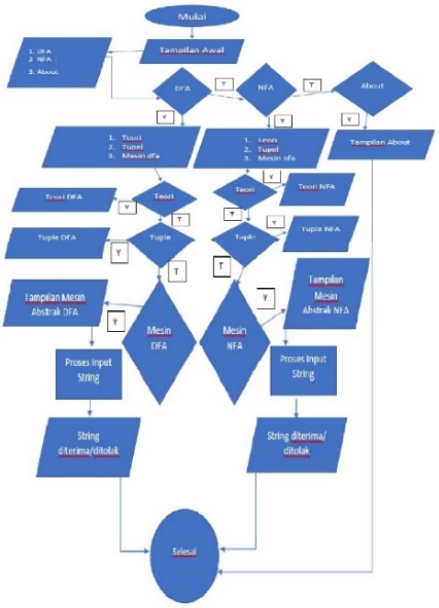
Table 3. Tabel Transisi

δ	0	1
q0	q0	q1
q1	ϕ	q4
q2	ϕ	q3
q3	q2	q5
q4	q5	ϕ
q5	ϕ	q5

Sebagai contoh, kita ambil jalur q0 dengan *input* ‘0101’, maka untai akan bergerak dari *state* awal (q0) kemudian *input* ‘0’ pertama posisi tetap di (q0), lalu *input* kembali ‘1’ maka posisi akan berubah menjadi (q1), kemudian *input* kembali ‘0’ maka posisi akan berubah menjadi (ϕ) dan hasilnya akan ditolak karena tidak berakhir di *state* penerima.

C. Perancangan Aplikasi

Aplikasi dirancang menggunakan aplikasi berbasis Android yang menggunakan Bahasa Pemrograman Java dan XML. XML(Extensible Markup Language) adalah bahasa markup untuk keperluan umum yang disarankan oleh W3C untuk membuat dokumen markup keperluan pertukaran data antar sistem yang beraneka ragam. Secara sederhana XML adalah suatu bahasa yang digunakan untuk mendeskripsikan dan memanipulasi dokumen secara terstruktur, serta menyediakan format tertentu untuk dokumen – dokumen yang mempunyai data terstruktur[5]. Implementasi XML pada aplikasi ini membangun tampilan Aplikasi simulator DFA dan NFA dengan baik salah satu hasil penggunaan XML bias dilihat di gambar 5.



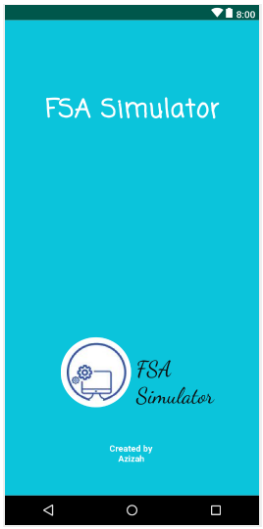
Gambar 4. Alur Aplikasi

Pada gambar 4 menjelaskan alur aplikasi ini berjalan dari sisi *user* atau pengguna, dimana aplikasi dimulai hal pertama yang akan dilihat adalah tampilan awal berupa

splash screen , lalu tampilan *Home* yang terdiri dari menu DFA, NFA, dan *About.* Jika *user* memilih menu DFA maka akan tampil menu teori, *tuple,* dan Mesin DFA, dimana di menu ini *user* dapat mengetahui string yang di *input* diterima atau ditolak. Setelah itu aplikasi selesai digunakan.

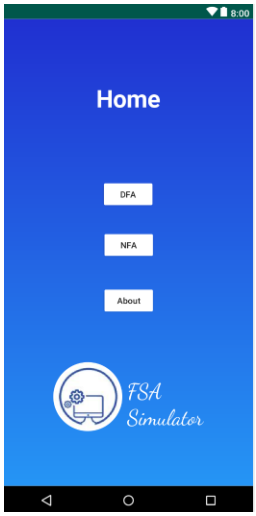
III. HASIL DAN PEMBAHASAN

Untuk membuat aplikasi simulator ini hal awal yang dilakukan mendesign tampilan awal di Android Studio, ini hanya optional saja.



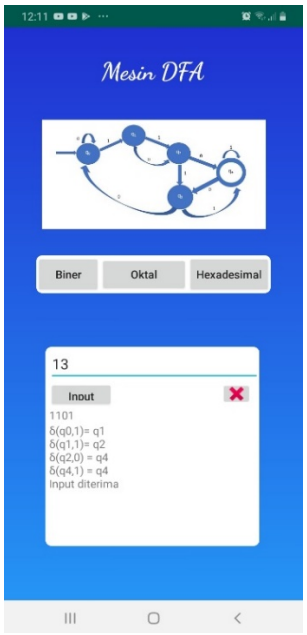
Gambar 5. Tampilan awal aplikasi simulator

Pada gambar 6 dapat dilihat terdapat pilihan yang dapat dilakukan, untuk *button* DFA terdapat teori, *tuple,* dan simulator. Pada *button* NFA juga sama terdapat teori, *tuple,* dan simulator. Pada *button* About berisi informasi tentang pembuat aplikasi.



Gambar 6. Tampilan *Home*

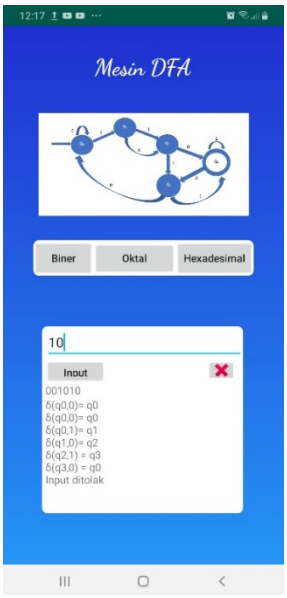
Pada gambar 7 merupakan simulator DFA, pada simulator ini bisa *input* string dengan bilangan *octal* dan *hexadecimal* yang akan di *convert* ke bilangan biner. Salah satu *input* diterima pada simulator dengan *input decimal* 13 lalu di *convert* ke biner menjadi 1101



Gambar 7. Simulator DFA dengan *input* diterima

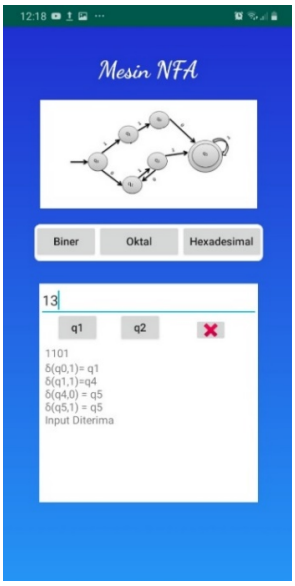
Pada gambar 8 merupakan simulator DFA, pada simulator ini bisa *input* string dengan bilangan *octal* dan *hexadecimal* yang akan di *convert* ke bilangan biner. Salah satu *input* ditolak pada simulator dengan *input*

decimal 10 lalu di *convert octal* ke biner menjadi 001010



Gambar 8. Simulator DFA dengan *input* ditolak

Pada gambar 9 merupakan simulator NFA, pada simulator ini bisa *input* string dengan bilangan *octal* dan *hexadecimal* yang akan di *convert* ke bilangan biner. Salah satu *input* diterima pada simulator dengan *input decimal* 13 di jalur q1 lalu di *convert* ke biner menjadi 1101

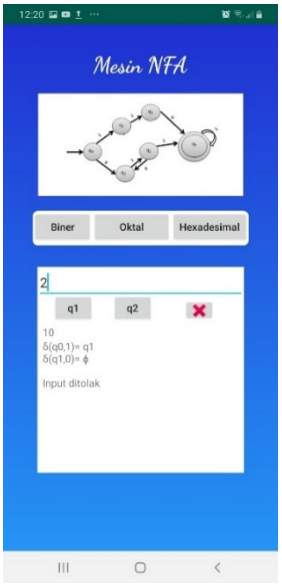


Gambar 9. Simulator NFA dengan *input* diterima

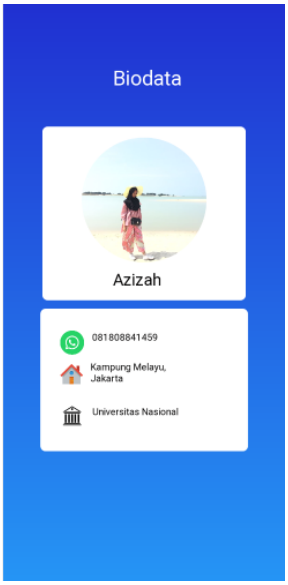
Pada gambar 10 merupakan simulator NFA, pada simulator ini bisa *input* string dengan bilangan *octal* dan *hexadecimal* yang akan di *convert* ke bilangan biner. Salah satu *input* diterima pada simulator dengan *input decimal* 7 di jalur q2 lalu di *convert* ke biner menjadi 0111

Gambar 10. Simulator NFA dengan *input* diterima

Kemudian pada gambar 11 merupakan simulator NFA, pada simulator ini bisa *input* string dengan bilangan *octal* dan *hexadecimal* yang akan di *convert* ke bilangan biner. Salah satu *input* ditolak pada simulator dengan *input decimal* 2 di jalur q1 lalu di *convert* ke biner menjadi 10



Gambar 11. Simulator NFA dengan *input* ditolak



Gambar 12. Tampilan Menu *About*

Pada gambar 12 merupakan tampilan menu *about* yang berisi tentang biodata pembuat aplikasi Simulator *Finite State Automata.*

IV. SIMPULAN

Simulator Finite State Automata adalah sebuah perangkat yang bisa digunakan untuk melakukan pengecekan terhadap inputan-inputan string pada Finite State Automata. Cara kerja simulator masih belum banyak yang mengetahuinya, melalui simulator ini diharapkan pengguna juga lebih paham akan hal tersebut.

DAFTAR PUSTAKA

[1]. Wahyudi, Jusuf dkk. INSTRUKSI BAHASA PEMROGRAMAN ADT (ABSTRACT DATA TYPE) PADA VIRUS DAN LOOP BATCH. Jurnal Media Infotama, Vol.9, No.2, September 2013

[2]. Sinurat, S. SIMULASI TRANSFORMASI REGULAR EXPRESSION TERHADAP FINITE STATE AUTOMATA. Pelita Informatika: Informasi dan Informatika, 4(1).2013

[3]. Suparyanto, Selo, Simulator Pengenal String Yang Diterima Sebuah Deterministic Finite Automata (DFA), CITEE, pp. 377–381.2017

[4]. Saputra, Tri Ichsan, dkk, "Simulasi *Vending Machine* Dengan Mengimplementasikan *Finite State Automata,* Vol. 3, No. 3, September 2018

[5]. Irawan, J. C., Pakereng, I. M., & Somya, R. (2012). Perancangan dan Implementasi Finite Automata pada Simulasi Vending Machine. De CARTESIAN, 1(1), 42-52.

[6]. Suparyanto, Selo, Simulator Pengenal String Yang Diterima Sebuah Deterministic Finite Automata (DFA), CITEE, pp. 377–381.2017

[7]. Saputra, Tri Ichsan, dkk, "Simulasi Vending Machine Dengan Mengimplementasikan Finite State Automata, Vol. 3, No. 3, September 2018

[8]. Widyasari, "Telaah Teoritis Finite State Automata Dengan Pengujian Hasil Pada Mesin Otomata." Jurnal Ilmiah SISFOTENIKA, Vol. 1, No. 1, Januari 2011.

[9]. Habibie, Aulia Akbar. ANALISIS DAN PERANCANGAN WEB SERVICE UNTUK SISTEM INFORMASI PEMESANAN TIKET BIOSKOP. Jawa Timur.