TUGAS AKHIR

Teori Bahasa dan Automata

Disusun Oleh :

Nama : Muhammad Razief

NIM : 21346015

Prodi : Informatika (NK)

Dosen Pengampu : Widya Darwin, S.Pd., M.Pd.T

PROGRAM STUDI INFORMATIKA

JURUSAN TEKNIK ELEKTRONIKA

FAKUTAS TEKNIK

UNIVERSITAS NEGERI PADANG

2023

KATA PENGANTAR

Rasa syukur kita hanya milik Allah SWT atas segala semua rahmatnya, sehinggasaya dapat menyelesaikan tugas akhir yang saya susun ini. Meskipun banyak rintangan dan hambatan yang saya alami dalam proses pekerjaan tetapi saya berhasil mengerjakan dengan baik dan tetap pada waktunya.

Dan harapan saya di sini semoga atas tugas akhir yang saya buat ini bisa menambah pengetahuan dan pengalaman bagi para pembaca, dan untuk kedepan nya kiat juga sama-sama memperbaiki bentuk atau menambah isi dari makalah agar semua akan lebih baik dengan sebelumnya.

Saya mengucapkan terima kasih kepada Ibu Widya Darwin, S.Pd., M.Pd.T sebagai Dosen Pengampu pada mata kuliah Teori Bahasa dan Automata yang telah membimbing saya dalam menyelesaikan tugas akhir yang saya buat ini.

Karena dari semua keterbatasan dari pengetahuan atau pun pengalaman saya, saya yakin masih banyak sekali dari kekurangan yang terdapat pada makalah ini. Oleh karena itu saya sangat berharap untuk saran dan kritik yang bisa membangun dari pembaca demisemua tugas akhir ini akan terselesaikan dengan benar.

Painan, Juni 2023

Muhammad Razief

PENGANTAR TEORI BAHASA & OTOMATA

A. Tingkat bahasa pemograman.

1. Bahasa pemrograman tingkat rendah

Bahasa mesin atau kode mesin merupakan satu-satunya bahasa yang bisa di olah komputer secara langsung tanpa transformasi sebelumnya (kompilasi).

Contoh bahasa pemrograman tingkat rendah :

Bahasa mesin (machine language)

2. Bahasa pemrograman tingkat menengah

Bahasa tingkat menengah memberikan satu tingkat abstraksi di atas kode mesin. Bahasa assembly memiliki sedikit semantik atau spesifikasi formal, karena hanya pemetaan simbol yang dapat di baca manusia. Biasanya, satu instruksi mesin di wakili sebagai satu baris kode *assembly*. Assembler menghasilkan file objek yang bisa dihubungkan dengan file objek lain atau dimuat sendiri.

Contoh bahasa pemrograman tingkat menengah :

Assembler

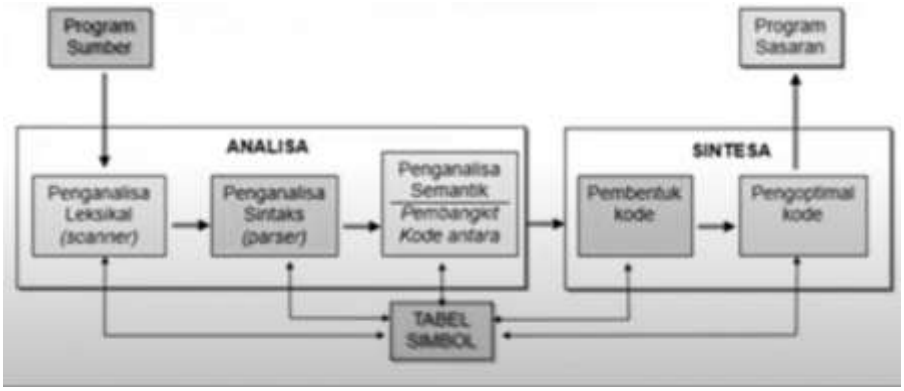
3. Bahasa pemrograman tingkat tinggi

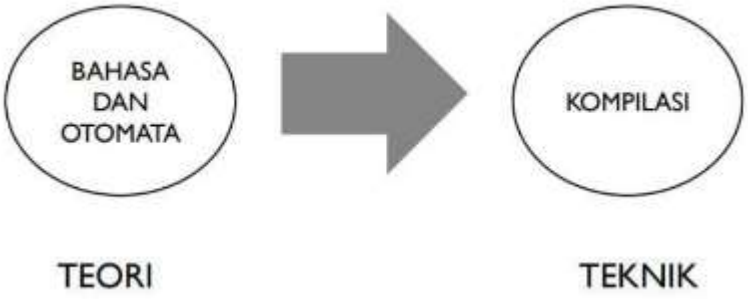
Contoh bahasa pemrograman tingkat tinggi :

C++ (Turbo C++)

Pascal (Turbo Pascal)

Proses Kompilasi





Teori Bahasa dan otomata adalah dasar dari Teknik kompilasi

Sebuah mesin yang hanya mengenali Bahasa mesin dapat memahami Bahasa pemograman tingkat tinggi karena ada compilator/penerjemah

Teori bahasa membahas mengenai bahasa formal. Salah satunya adalah untuk kepentingan kompilator

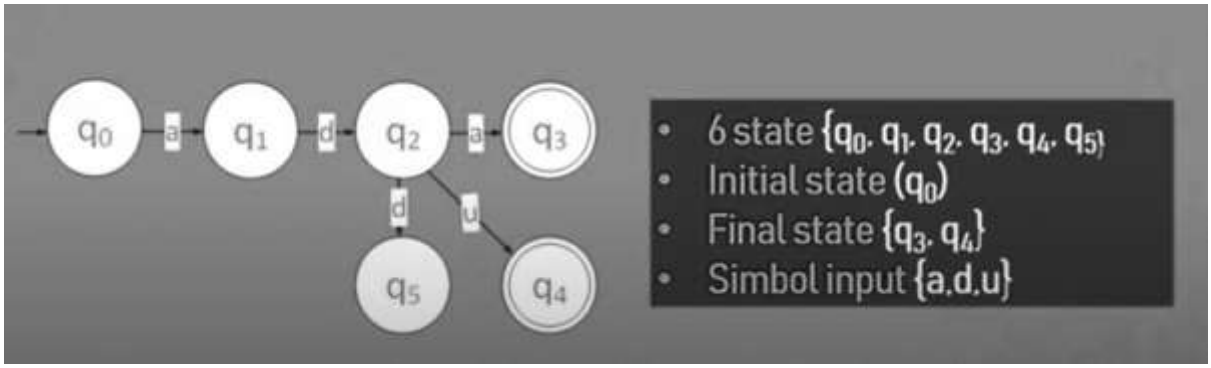
Bahasa di dalam kamus adalah suatu sistem yang meliputi pengekspresian gagasan, fakta, konsep, termasuk sekumpulan simbol-simbol dan aturan untuk melakukan manipulasinya. Bahasa bisa juga disebut sebagai rangkaian simbol-simbol yang mempunyai makna.

Otomata merupakan suatu sistem yang terdiri atas sejumlah berhingga state, di mana state menyatakan informasi mengenai input. Otomata juga dianggap sebagai mesin otomatis (bukan mesin fisik) yang merupakan suatu model matematika dari suatu sistem yang menerima input dan menghasilkan output, serta terdiri dari sejumlah berhingga state.

Input pada mesin otomata dianggap sebagai Bahasa yang harus dikenali oleh mesin, mesin akan mengindikasikan apakah suat bahasa dapat diterima atau tidak.

Hubungan di antara bahasa dan otomata adalah bahasa dijadikan sebagai input oleh suatu mesin otomata, selanjutnya mesin otomata akan membuat keputusan yang mengindikasikan apakah input itu diterima atau tidak.

Contoh mesin otomata sederhana

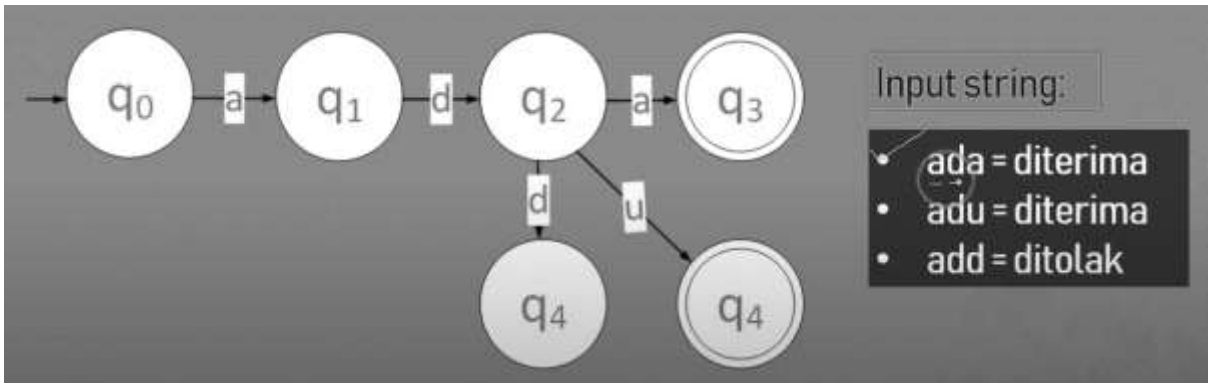


String input diterima jika mencapai final state, selain itu ditolak

Pembacaan simbol pertama dimulai dari initial state

Perpindahan state berdasarkan simbol yang dibaca

Maka hasilnya akan menjadi



SIMBOL SRING DAN BAHASA

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 ε (atau ^) sehingga $|\varepsilon| = 0$. String hampa dapat dipandang sebagai simbol hampa karena keduanya tersusun dari nol buah simbol.

• Alfabet adalah hinpunan hingga (finite set) simbol-simbol.

Suatu sistem yang meliputi pengeskspresian gagasan. fakta, konsep, termasuk sekumpulan simbol-simbol dan aturan untuk melakukan manipulasinya

Bahasa adalah Suatu sistem yang meiliputi pengekspresian gagasan, fakta konsep,termasuk sekumpulan simbol-simbol dan aturan untuk melakukan manipulasinya.Alphabet adalah himpunan berhingga dari simbol-simbol.

Bahasa juga disebut sebagai rangkaian simbol yang memiliki makna.

oSebuah bahasa adalah himpunan string-string dari simbolsimbol untuk suatualphabet.

oKarena sebuah bahasa adalah kumpulan dari string-string, maka kita bisamempunyai bahasa yang tidak mempunyai string-string, yaitu Bahasa kosongyang dinotasikan seperti menuliskan notasi himpunan kosong.

oInput pada mesin otomata dianggap sebagai Bahasa yang harus dikenali olehmesin.

oMesin akan mengindikasikan apakah suatu Bahasa dapat diterima atau tidak

ATURAN PRODUKSI DAN GRAMER

Simbol terminal dan non terminal

Apa itu simbol terminal?

- simbol terminal adalah simbol yang tidak dapat diturunkan lagi
- arti bisa **diturunkan** yaitu misalkan simbol A yang dapat diturunkan menjadi bc

yang termasuk simbol terminal:

- huruf kecil alfabet: a,b,c
- simbol operator: + (tambah), – (kurang)
- simbol tanda baca: , (koma) ! (tanda seru)
- string yang tercetak tebal, misalnya: if, then dan else

Apa itu simbol NON terminal (Variabel)?

- adalah simbol yang masih bisa diturunkan menjadi simbol terminal atau non terminal lainnya

yang termasuk simbol NON terminal (variabel):

- huruf besar alfabet: A, B, C
- huruf S sebagai simbol awal
- String yang tercetak miring misal expr dan stmt

Aturan Produksi

Dalam teori bahasa dan otomata, aturan produksi dinyatakan dalam:

$\alpha \rightarrow \beta$

(dibaca: alpha menghasilkan atau menurunkan beta)

- Dengan menerapkan aturan produksi, suatu tata bahasa bisa menghasilkan sejumlah string.
- Himpunan semua string tersebut adalah bahasa yang didefinisikan oleh tata bahasa tersebut.

contoh aturan produksi:

$T \rightarrow a$

(dibaca: T menghasilkan a)

Contoh lain:

E → T | T + E

(dibaca: E menghasilkan T atau E menghasilkan T + E)

Hirarki Chomsky

Grammar

- Grammar atau tata bahasa didefinisikan secara formal sebagai kumpulan dari himpunan himpunan variabel, simbol simbol terminal, simbol awal, yang dibatasi oleh aturan aturan produksi.

4 tingkatan tata bahasa menurut Chomsky:

1. Tipe 0 – Unrestricted Grammar
2. Tipe 1 – Context Sensitive Grammar
3. Tipe 2 – Context Free Grammar
4. Tipe 3 – Regular Grammar

Tipe 0 – Unrestricted Grammar

- Aturan produksinya tidak memiliki batasan
- Dalam aturan α → β pada tipe 0:
    - alpha adalah string terminal dan non-terminal dengan setidaknya 1 non-terminal
    - alpha tidak boleh kosong
    - beta adalah rangkaian simbol terminal dan non – terminal
    - α adalah (V + T)*V(V+T)*
    - β adalah (V+T)*
    - note:
    - (tanda plus '+' dibaca atau)
    - (tanda * (asterisk) yang berarti bisa tidak muncul atau bisa juga muncul hingga berkali kali (0 – n))
- Tipe ini menghasilkan bahasa yang dikenali oleh mesin Turing

- Bahasa ini juga dikenal dengan nama “Recursively Enumerable Languages”.

jadi aturan yang perlu diingat adalah:

- α adalah (V + T)*V(V+T)*
- β adalah (V+T)*

Contoh penerapan aturan tipe 0:

- S → ACaB (Terpenuhi, karena di ruas kiri ada non terminal, di ruas kanan ada terminal dan non-terminal)
- Bc → acB (Terpenuhi, karena di ruas kiri ada non terminal, di ruas kanan ada terminal dan non-terminal)
- CB → DB (Terpenuhi)
- aD → Db (Terpenuhi)
- Sab → ba (Terpenuhi)
- A → S (Terpenuhi)

Tipe 1 – Context Sensitive Grammar

- Aturan produksinya sama dengan tipe 0 namun dibatasi dengan aturan |a|≤|b| (jumlah simbol di ruas kiri harus lebih kecil atau sama dengan jumlah simbol di ruas kanan)
- Aturan S → Ɛ dibolehkan jika S tidak muncul pada ruas kanan setiap aturan
- Menghasilkan bahasa yang dikenali oleh Linear Bound Automata

jadi aturan yang perlu diingat adalah:

- |a|≤|b|
- a adalah (V+T)*V(V+T)*
- b adalah (V+T)* (V+T) (V+T)*

Contoh penerapan:

- S → AB (Terpenuhi)
- AB → abcd (Terpenuhi, ukuran ruas kiri lebih kecil dari ruas kanan)
- B → b (Terpenuhi)
- aD → Db (Terpenuhi, ukuran kedua ruas sama)
- aB → aa | aaAA (Terpenuhi)

- Ac → Bbcc (Terpenuhi)

Tipe 2 – Context Free Grammar

- Aturan produksi ini sama dengan tipe 0 namun a sisi kiri hanya boleh memiliki 1 variabel non terminal (|a| = 1) dan b tidak memiliki batasan
- a adalah sebuah simbol non terminal
- b dapat berupa rangkaian atau simbol terminal, non terminal atau epsilon
- tipe ini menghasilkan bahasa yang dikenali oleh Non Deterministic Push Down Automata

jadi aturan yang perlu diingat adalah:

- |a| = 1
- a adalah V
- b adalah (V+T)*

contoh:

- S → X a (Terpenuhi, di ruas kiri hanya ada satu non terminal yaitu S dan di ruas kanan boleh terminal atau non terminal)
- X → a (Terpenuhi)
- X → aX (Terpenuhi)
- X → abc (Terpenuhi)
- X → Є (Terpenuhi)
- S → AB (Terpenuhi)
- A → a (Terpenuhi, di ruas kiri hanya ada satu non terminal dan di ruas kanan ada satu terminal)
- B → b (Terpenuhi)

Tipe 3 – Regular Grammar

- Aturan S → Ɛ dibolehkan jika S tidak muncul pada ruas kanan setiap aturan
- α adalah sebuah simbol non terminal
- β adalah simbol terminal atau simbol terminal dengan sebuah simbol variabel yang jika ada terletak pada posisi paling kanan
- Menghasilkan bahasa yang dikenali oleh Finite State Automata

jadi perlu diingat:

- α adalah V
- β adalah T* atau T*V

Contoh:

- X → Ɛ
- X a | aY
- Y → b
- S → abB
- B → cd

FINAL STATE AUTOMATA

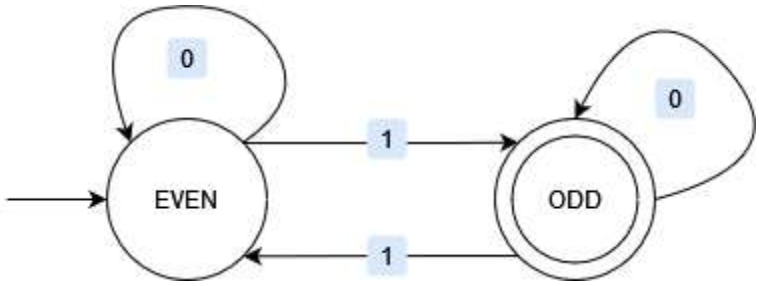
Apa itu FSA?

- FSA adalah mesin abstrak berupa sistem model matematika dengan masukan dan keluaran diskrit yang dapat mengenali bahasa paling sederhana
- mekanisme kerja FSA dapat diterapkan pada analisis leksikal, text editor, protocol dan komunikasi jaringan

contoh analisis leksikal:

- analisis leksikan adalah salah satu bagian dari proses penerjemahan
- code yang dituliskan akan dianalisis seperti yang mana variabel, operator, keyword

Arti bentuk symbol pada graf transisi FSA



Keterangan gambar:

- Initial state ditandai dengan busur tanpa asal
- Lingkaran menyatakan state
- Label pada lingkaran adalah nama state, yaitu EVEN dan ODD
- Busur adalah transisi / arah perpindahan state
- Label pada busur adalah simbol input, yaitu 0 dan 1
- Lingkaran ganda menyatakan final state, yaitu ODD

Pernyataan Final State Automata secara Formal

FSA dapat dinyatakan dalam 5 tupel atau M=(Q, Σ, δ, q0, F) dimana:

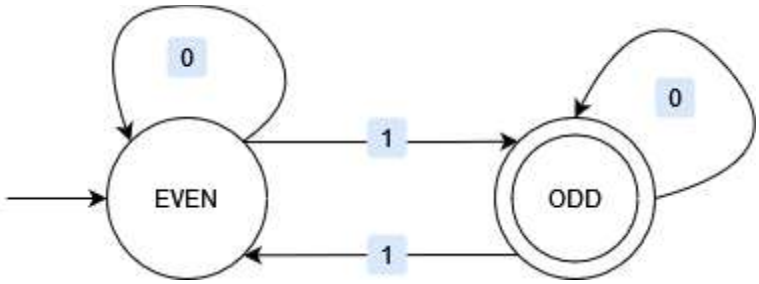
- Q adalah Himpunan state atau kedudukan
- Σ (dibaca: sigma) adalah Himpunan symbol input / masukan / abjad

    - Jika pada contoh sebelumnya, terdapat input / simbol yaitu 0 dan 1
- δ (dibaca: delta) adalah Fungsi transisi
- q0 (atau bisa juga S) adalah State awal q0. dimana q0 Q
- F adalah Himpunan State akhir (final) F ⊆ Q

Catatan:

- Jumlah state akhir bisa lebih dari satu

Contoh:



- Q = {ODD, EVEN}
- Σ = {0, 1}
- q0 = EVEN
- F = {ODD}
- δ (Transisi):
    - δ(EVEN, 0) = EVEN State EVEN menerima inputan 0 ke EVEN
    - δ(EVEN, 1) = ODD
    - δ(ODD, 0) = ODD
    - δ(ODD, 1) = EVEN

catatan: Final state F = {ODD} berupa himpunan karena final state dapat berjumlah lebih dari satu

Jadi aturan ini menyesuaikan model mesinnya

Contoh ketika mesin digunakan:

- Ketika mendapat input 1101, maka urutan state yang terjadi adalah EVEN1ODD1EVEN0EVEN1ODD
- berakhir dengan state ODD, maka 1011 diterima mesin

- Karena diterima maka dapat diketahui bahwa jumlah bit 1 nya adalah ganjil

Contoh Lain:

- Ketika mendapat input 101, maka urutan state yang terjadi adalah EVEN1ODD0ODD1EVEN
- berakhir dengan state EVEN, maka 101 ditolak mesin
- Karena ditolak maka dapat diketahui nilai bit nya genap

Berdasarkan pendefinisian kemampuan merubah statenya, FSA dikelompokkan kedalam 2 jenis, yaitu:

- Deterministic FSA
- Non Deterministic FSA

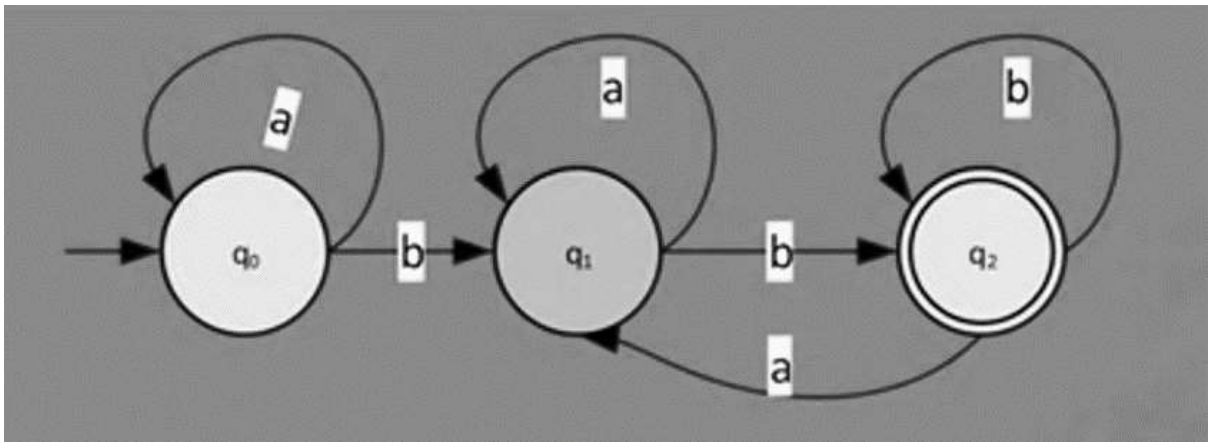
DETERMINISTIC FINAL STATE AUTOMATA

Berdasarkan pendefinisian kemampuan merubah statenya, FSA dikelompokkan ke dalam dua jenis

- Deterministic FSA
- Non- Deterministic FSA

Contoh 1:

Pada DFA, dari suatu state hanya ada tepat satu state berikutnya untuk setiap simbol masukan yang diterima



Q = {q0, q1, q2}

∑ = { a,b }

S = q0

F = {q2}

Fungsi Transisi

$$\delta(q_0, a) = q_0 ; \quad \delta(q_0, b) = q_1$$
$$\delta(q_1, a) = q_1 ; \quad \delta(q_1, b) = q_2$$
$$\delta(q_2, a) = q_1 ; \quad \delta(q_2, b) = q_2$$

Tabel transisi

δ	a	b
q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_1	q_2

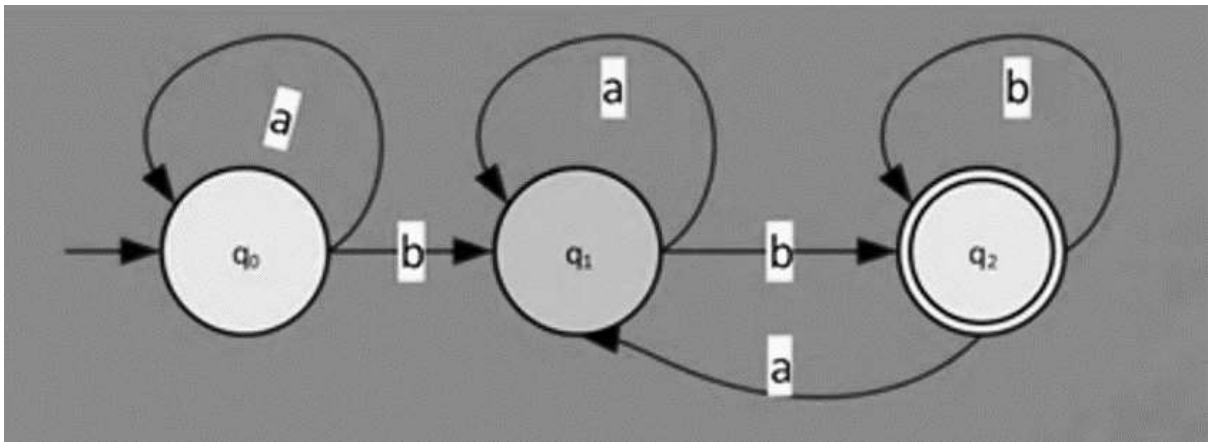
Kapan suatu string input dinyatakan diterima?

Suatu string x dinyatakan diterima,Bila δ (S,x) berada pada state akhir.

Bila M adalah sebuah bahasa FSA

M = (Q, ∑, δ, S, F) menerima bahasa yang disebut L(M) yang merupakan himpunan { x | δ(S,x) di dalam F }

Contoh 2:

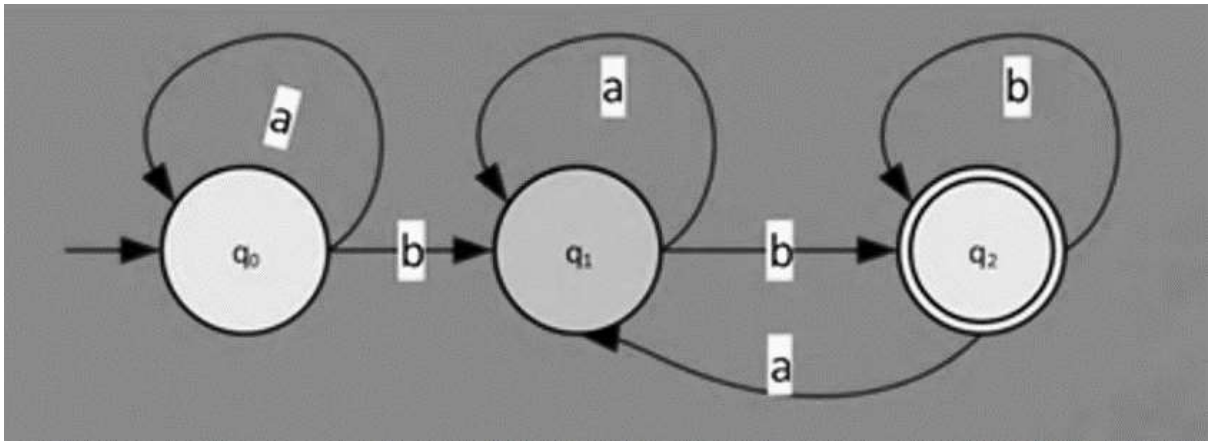


Jika pada gambar contoh 1 kita inputkan string ‘abb’. Maka:

$$\delta(q_0 , abb) = \delta(q_0 , bb) = \delta(q_1 , b) = q_2$$

Karena q2 adalah state akhir maka ‘abb’ berada berada dalam L(M}

Contoh 3:

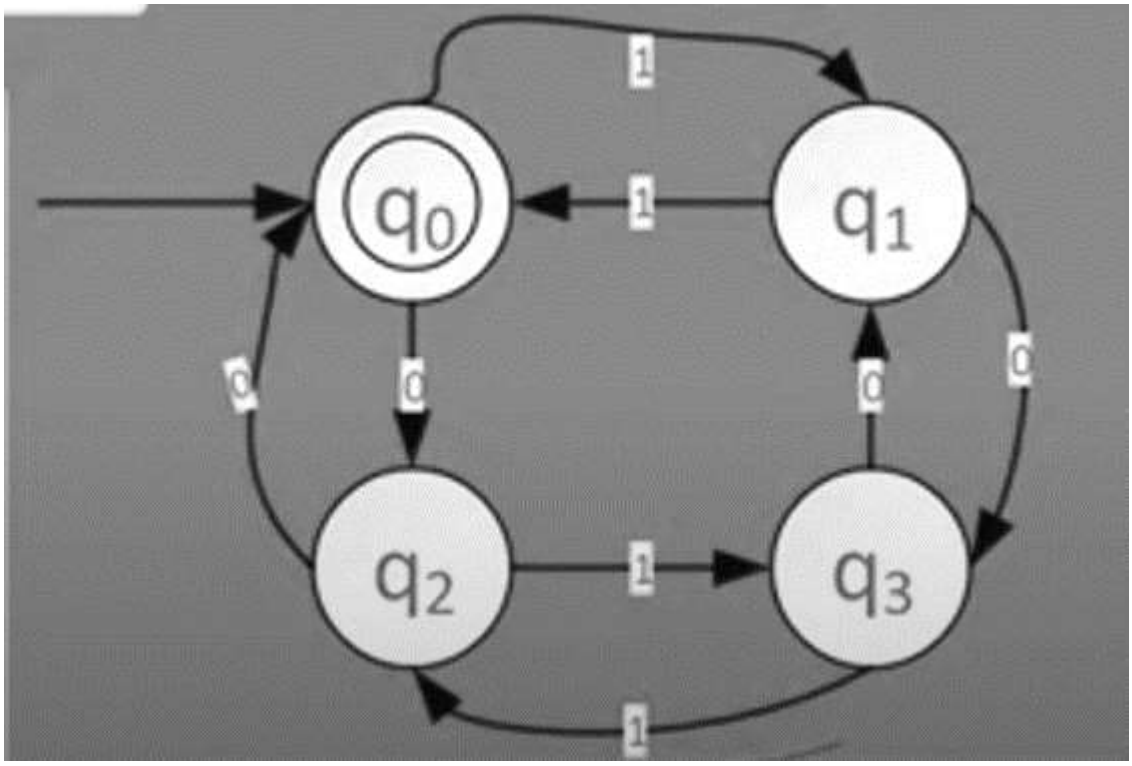


Jika pada gambar contoh 1 inputkan string ‘baba’. Maka;

$$\delta(q_0 , baba) = \delta(q_1 , aba) = \delta(q_1 , ba) = \delta(q_2 , a) = q_1$$

Karena q1 bukan state akhir maka ‘baba’ tidak berada dalam L(M)

Contoh 4 – Pengecekan bit 0 dan 1 ganap



Tabel transisi

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

Konfigurasi NFA berikut :

Q = {q0, q1, q2,q3}

∑ = { 0,1 }

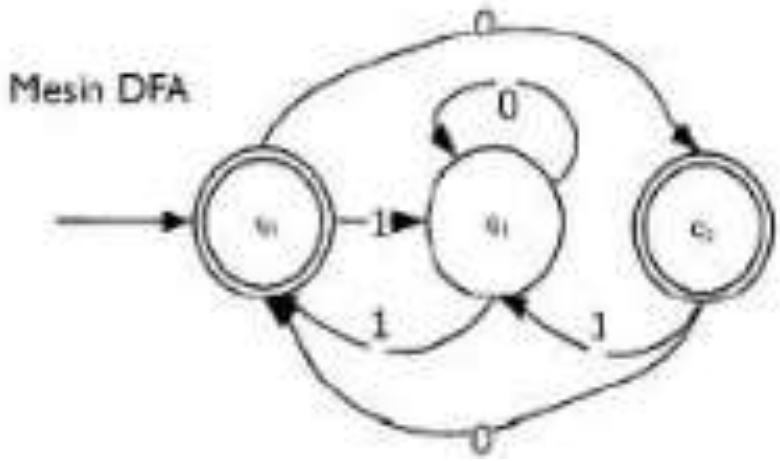
S = q0

F = {q0}

Secara formal dinyatakan sebagai berikut:

String	Status
0011	Diterima
00111	Ditolak
01010	Ditolak
010111	Diterima

Contoh 5



Tabel transisi:

δ	0	1
q_0	q_2	q_1
q_1	q_1	q_0
q_2	q_0	q_1

Secara formal dinyatakana sebagai berikut:

String	Status
01	Ditolak
10	Ditolak
1110	Ditolak
11	Diterima
101	Diterima
1100101	Diterima
0100011101	Ditolak

NON DETERMINISTIC FINITE STATE AUTOMATA

Kemungkinan transisinya ke lebih dari satu state. Dari suatu statebisa terdapat 0, 1 atau lebih transisi dengan label input yang sama.Perubahan state dapat terjadi secara spontan tanpa input (transisikosong).

Salah satu tracing-nya berakhir di state akhir, atau himpunan state setelah membaca string tersebut mengandung state akhir

FSA dinyatakan oleh 5 tuple :M = (Q , Σ , δ , S , F) dimana:

Q = Himpunan state / kedudukan

Σ = Himpunan symbol input

δ = Fungsi Transisi

q0 = State awal q0,

F = himpunan state akhir

* Catatann : Jumlah state akhir bisa lebih dari satu

Contoh 1 :

FSA dinyatakan oleh 5 tuple :M = (Q , Σ , δ , S , F) dimana:

Q = {q0 , q1 , q2 , q3 , q4 }

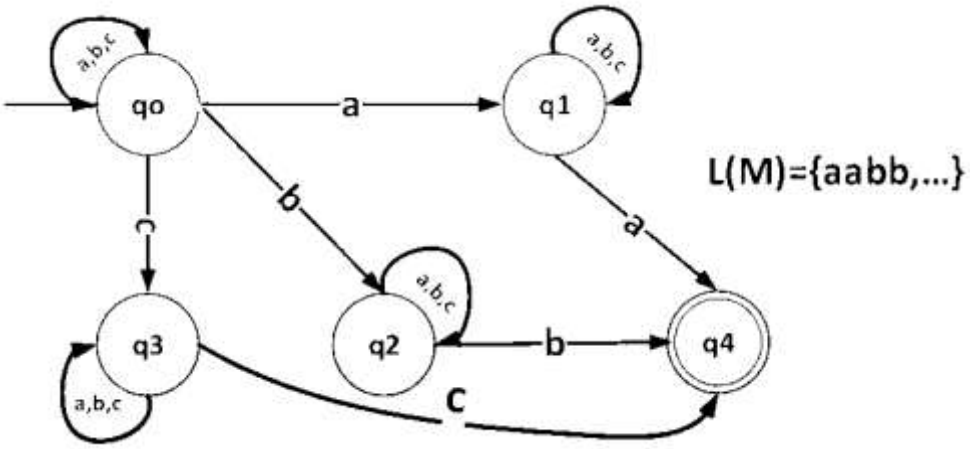
Σ = {a,b,c}

S = q0

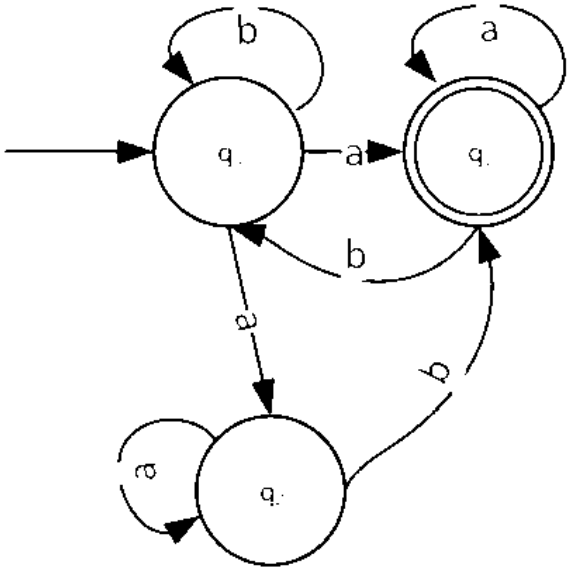
F = {q4}

Tabel Transisi :

δ	a	b	c
q0	{q0,q1}	{q0,q2}	{q0,q3}
q1	{q1,q4}	{q1}	{q1}
q2	{q2}	{q2,q4}	{q2}
q3	{q3}	{q3}	{q3,q4}
q4	ø	ø	ø



Contoh 2 :



Gambar tersebut termasuk NFA karena jika q0 manerima inputan ‘a’ maka akan berpindah ke q1 atau q2. Tidak tepat satu berikutnya

δ	a	b
q_0	{q1, q2}	q_o
q_1	q_1	q_0
q_2	q_2	q_1

Pada NFA, String diterima jika setidaknya ada 1 dari semuakemungkinan transisi berakhir pada sebuah final sate. Harus mencobasemua kemungkinan.Pada NFA boleh terdapat transisi kosong. Simbol ε berarti tanpamasukan atau disebut transisi kosong. Terjadi perubahan state secaraspontan