ISBN:

BUKU AJAR

MATA KULIAH

TEORI BAHASA DAN AUTOMATA

PENYUSUN:

DEWI SOYUSIAWATY, S.T., M.T.

MUREIN MIKSA MARDHIA, S.T., M.T.

NAMA PENERBIT

KATA PENGANTAR

Puji dan syukur dipanjatkan kepada Allah SWT yang telah memberi rahmat dan hidayahNya sehingga penyusunan buku ajar mata kuliah Teori Bahasa dan Automata ini akhirnya bisa diselesaikan. Buku ajar ini disusun sebagai referensi dosen dan mahasiswa untuk perkuliahan Teori Bahasa dan Automata di Program Studi Teknik Informatika Universitas Ahmad Dahlan.

Materi yang disajikan sudah diurutkan disesuaikan dengan perencanaan pembelajaran mingguan mata kuliah, sehingga harapannya mahasiswa dapat memiliki referensi belajar untuk tiap materi yang disampaikan.

Penulis menyadari masih banyak ketidaksempurnaan pada penulisan, baik isi maupun redaksi, oleh karenanya kritik dan saran yang membangun diharapkan dapat memperbaiki untuk tahun berikutnya.

Terima kasih kepada semua pihak yang telah membantu baik secara langsung ataupun tidak terhadap terselesaikannya buku ajar ini. Semoga dapat bermanfaat bagi yang membutuhkan.

Yogyakarta, Agustus 2019

Penyusun

HAK CIPTA

BUKU AJAR TEORI BAHASA OTOMATA

Dewi Soyusiawaty, S.T., M.T
Murein Miksa Mardhia, S.T., M.T.

Diterbitkan oleh:
Program Studi Teknik Informatika
Fakultas Teknologi Industri
Universitas Ahmad Dahlan
Jalan Ring Road Selatan, Tamanan, Banguntapan, Bantul Yogyakarta 55166

Penulis	: Dewi Soyusiawaty, S.T., M.T Murein Miksa Mardhia, S.T., M.T.
Editor	: Laboratorium Teknik Informatika, Universitas Ahmad Dahlan
Desain sampul	: Laboratorium Teknik Informatika, Universitas Ahmad Dahlan
Tata letak	: Laboratorium Teknik Informatika, Universitas Ahmad Dahlan
Ukuran/Halaman	: 21 x 29,7 cm / 94 halaman

Didistribusikan oleh:

Laboratorium Teknik Informatika
Universitas Ahmad Dahlan
Jalan Ring Road Selatan, Tamanan, Banguntapan, Bantul Yogyakarta 55166
Indonesia

BAB I
PENGANTAR TEORI BAHASA DAN AUTOMATA

1.1. Tujuan Instruksional

A. Tujuan Instruksional Umum

Mahasiswa memahami konsep utama dari Teori Bahasa dan Automata

B. Tujuan Instruksional Khusus

Mahasiswa memahami bahasa dan automata serta penerapannya.

1.2. Definisi Bahasa dan Automata

A. Bahasa

Ada beberapa definisi bahasa yaitu :

1. Rangkaian simbol yang mempunyai makna
2. Pengekspresian gagasan, fakta, konsep, termasuk sekumpulan simbol-simbol dan aturan untuk melakukan manipulasinya.
3. Bahasa Formal → Kumpulan kalimat
4. Semua kalimat dalam sebuah bahasa dibangkitkan oleh sebuah tata bahasa (grammar) yang sama.
5. Bahasa formal karena grammar diciptakan mendahului pembangkitan setiap kalimatnya.

   Bahasa manusia bersifat sebaliknya; grammar diciptakan untuk meresmikan kata-kata yang hidup di masyarakat.

B. Automata

Automata memiliki beberapa definisi,antara lain :

1. Mesin abstrak - ~~Mesin fisik~~
2. Model matematika
3. Sistem yang menerima input dan menghasilkan output, serta terdiri dari sejumlah berhingga state.

Mesin abstrak yang dapat mengenali (recognize), menerima (accept), atau membangkitkan (generate) sebuah kalimat dalam bahasa tertentu.

C. Teori Bahasa dan Automata

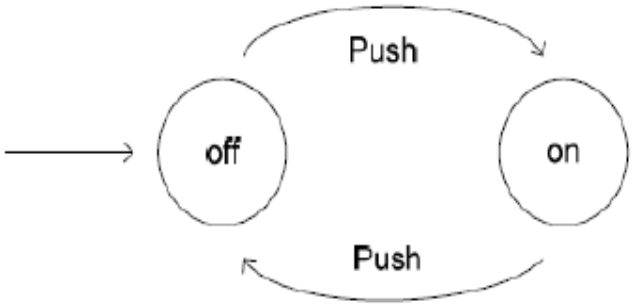
Bahasa sebagai input oleh suatu mesin otomata, selanjutnya mesin otomata akan membuat keputusan yang mengindikasikan apakah input itu diterima atau tidak.

D. Objektif

Membahas model komputasi sebagai mesin abstrak yang dapat didefinisikan secara matematis, mulai dari yang paling sederhana sampai yang *powerfull.*

1.3. Contoh Terapan Teori Automata

A. Model *Switch On/Off*

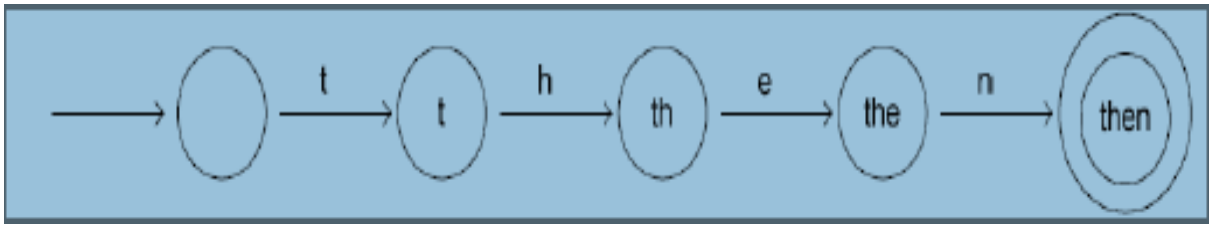


Gambar 1. Model Switch

Keterangan :

1. Mengingat apakah switch berada dalam state ‘on’ atau state ‘off’
2. Switch berada dalam state “off” maka setelah tombol ditekan state berubah menjadi “on”.
3. Jika switch berada dalam state “on” maka setelah tombol ditekan state berubah menjadi “off”
4. State >> lingkaran
5. Arc diantara state >> input >> push
6. Start state >> off
7. Final state >> on >> lingkaran ganda

B. Finite Automata – Lexical Analyzer

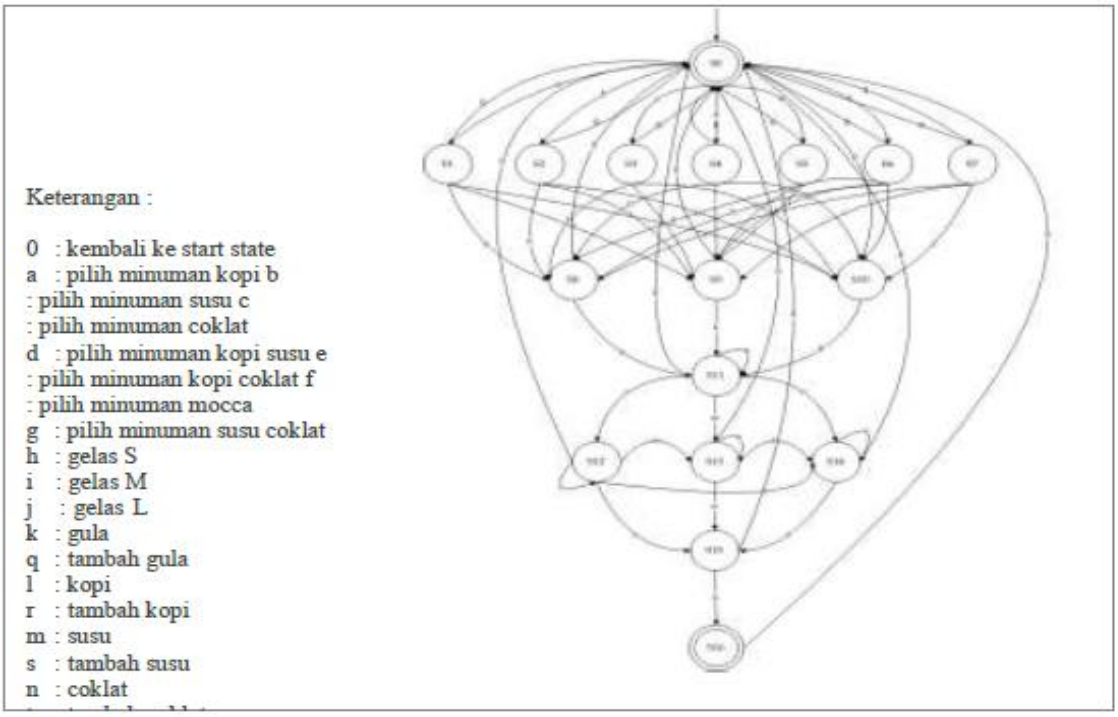


Gambar 2. Lexical Analyzer

Keterangan :

1. Mengenali *keyword* “then”
2. Diperlukan lima *state*
3. Input dinyatakan oleh huruf.
4. *Start state* merupakan string kosong, dan setiap *state* memiliki transisi pada huruf selanjutnya
5. *State* yang diberi nama “then” dimasuki ketika input mengeja kata “then”.
6. *state* “then”dinyatakan sebagai *accepting state*.

C. Automata - Mesin Pembuat Minuman Kopi Otomatis



Gambar 3. FSA Pembuat Kopi

Gambar 4. Penerapan Automata

D. Penelitian Terkait

Penerapan Konsep Finite State Automata (FSA) pada Mesin Pembuat Minuman Kopi Otomatis

[1]Wamiliana, [2]Didik Kurniawan [2]Rizky Indah Melly E.P

[1] Jurusan Matematika FMIPA Universitas Lampung
[2] Jurusan Ilmu Komputer FMIPA Universitas Lampung

Abstract

This research discusses about a simulating application for automatic coffee machine which can do processes of making 7 types of choices of drinks; coffee, milk, chocolate, coffee milk, coffee brown, mocha, and chocolate milk automatically. In this application, the concept of Finite State Automata (FSA) was applied to recognize and to capture the pattern on the process for making coffee drinks. In this application, Finite State Automata (FSA) was applied to read input symbols given from the starting state to the final state in order to get the language recognized by the machine. Further, the process will be done accordance with the read language.

Keywords: *Coffee Machine, Finite State Automata (FSA), Language and Automata Theory.*

Gambar 5. Penelitian terkait teori Automata

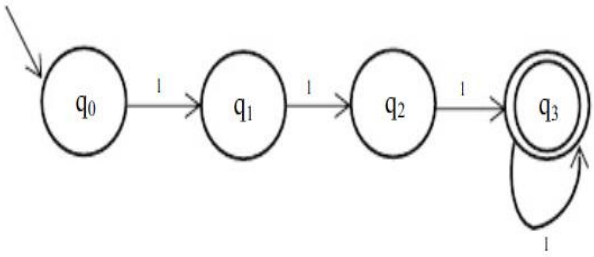
Aplikasi simulasi mesin pembuat minuman kopi otomatis ini dibangun dengan menggunakan *tools Microsoft Visual Studio* 2008. Pada aplikasi akan digunakan 8 bahan dasar pembuat minuman yaitu : gelas dengan tiga jenis ukuran yaitu *small, medium* dan *large*, air, gula, bubuk kopi, bubuk susu, dan bubuk coklat. Selanjutnya, dari bahan tersebut mesin dapat membuat 7 jenis produk minuman yaitu : kopi, susu, coklat, kopi susu, kopi coklat, kopi susu coklat dan susu coklat. Aplikasi ini juga dirancang hanya sebagai mesin yang dapat melakukan proses pembuatan minuman kopi secara otomatis dan tidak melakukan proses transaksi penjualan, sehingga tidak membutuhkan *input* berupa uang koin maupun uang kertas.

Gambar 6. Deskripsi penelitian

E. Automata – Mesin Karaoke

PENERAPAN DETERMINISTIC FINITE AUTOMATA
PADA MESIN KARAOKE

Gambar 7. Mesin Karaoke



Gambar 8. Finite State Diagram Mesin Karaoke

F. Teori Pendukung

1. Himpunan

Himpunan adalah koleksi dari beberapa elemen, yaitu :

$A = \{1, 2,3\}$

$B = \{train,bus,bicycle, airplane\}$

Bisa diulis :

$1 \in A$ *1 anggota bagian A*

Ship $\notin$ B *Ship bukan anggota bagian dari B*

a. Representasi dari Himpunan

C = { a, b, c, d, e, f, g, h, i, j, k }

C = { a, b, …, k } $\rightarrow$ *set yang mempnyai akhir*

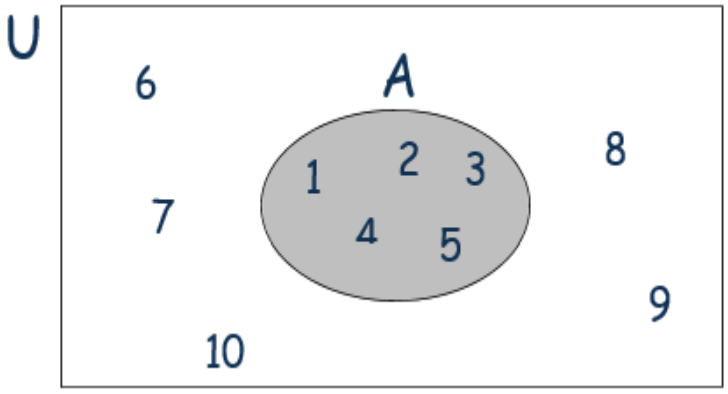
S = { 2, 4, 6, … } $\rightarrow$ *set yang tidak mempunyai akhir*

S = { j : j > 0, dan j = 2k untuk semua k>0 }

S = { j : j bukan bilangan negatif dan ada }

Contoh :

A = { 1, 2, 3, 4, 5 }



Gambar 9. Himpunan

Himpunan Universal : seluruh elemen yang mungkin ada

U = { 1 , … , 10 }

b. Operasi Himpunan

A = { 1, 2, 3 }

B = { 2, 3, 4, 5}

- Union

  A ∪ B = { 1, 2, 3, 4, 5 }

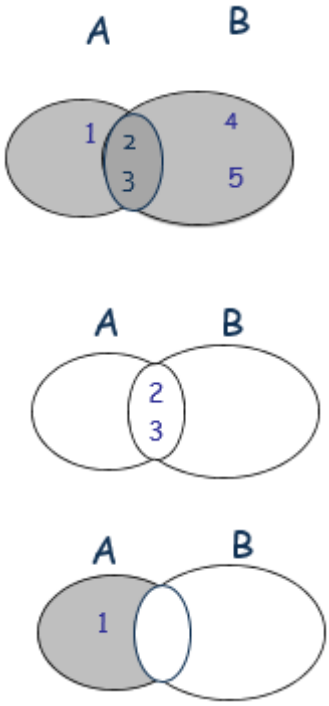
- Intersection

  A ∩ B = { 2, 3 }

- Difference

  A - B = { 1 }

  B - A = { 4, 5 }

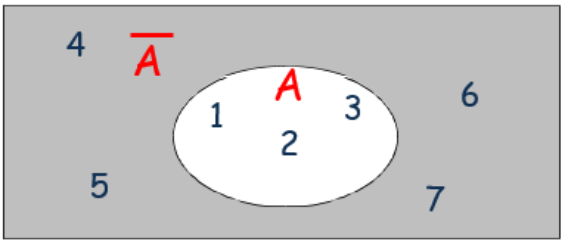


Gambar 10. Venn diagrams

- Complement

  Himpunan Universal = {1, …, 7}

  A = { 1, 2, 3 } $\bar{A}$= { 4, 5, 6, 7}



$\bar{\bar{A}} = A$

Gambar 11. Operasi Complement

- Hukum DeMorgan's

  $\overline{A \cup B} = \bar{A} \cap \bar{B}$

  $\overline{A \cap B} = \bar{A} \cup B$

- Himpunan Kosong: Ø

  Ø = { } $\bar{O}$ = Universal Set

  S ∪ Ø = S

  S ∩ Ø = Ø

  S – Ø = S

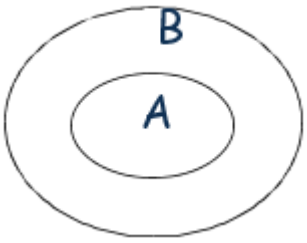
  Ø – S = Ø

- Himpunan Bagian

  A = { 1, 2, 3} B = { 1, 2, 3, 4, 5 }

  $A \subseteq B$

  Himpunan Bagian yang Sesuai: A ⊂ B

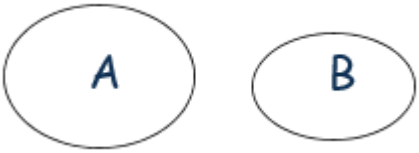


Gambar 12. Himpunan Bagian

- Himpunan Disjoint

  A = { 1, 2, 3} B = { 5, 6}

  $A \cap B$



Gambar 13. Operasi Disjoint

- Cardinalitas Himpunan

  Untuk set yang mempunyai nilai akhir A = { 2, 5, 7 }

  |A| = 3

  (ukuran set)

- Powersets

  Powersets adalah himpunan dalam himpunan

  S = { a, b, c }

  Powerset dari S = himpunan dari seluruh subsets S

  2^S = { ø {a}, {b}, {c}, {a, b}, {a, c}, {b, c}, {a, b, c} }

  Observasi: $| 2^S | = 2^{|S|} = 2^3 = 8$

- Produk Cartesius

  A = { 2, 4 } B = { 2, 3, 5 }

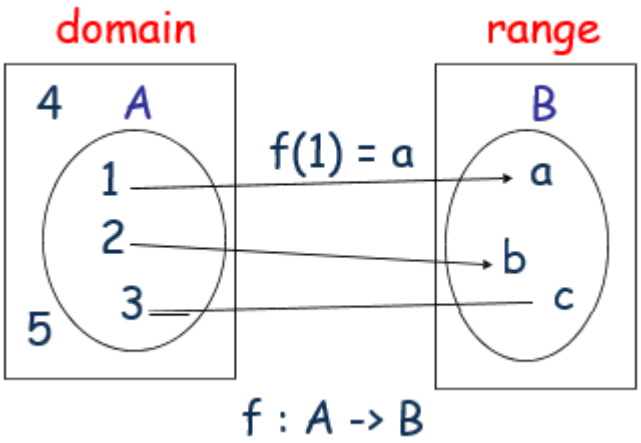
  A X B = { (2, 2), (2, 3), (2, 5), (4, 2), (4, 3), (4, 5) }

  |A X B| = |A| |B|

  Secara Umum untuk dua himpunan atau lebih

  A X B X … X Z

- Fungsi



Gambar 14. Fungsi

Jika A = domain

maka f adalah funsi total

Jika tidak f adalah function parsial (bagian)

- Relasi

$R = \{(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots\}$

$x_i \; R \; y_i$

e. g. if R = '>': 2 >1 , 3>2 , 3>1

- Equivalensi Relasi

Refleksi: x R x

Simetrik: x R y $\rightarrow$ y R x

Transitif: x R y dan y R z $\rightarrow$ x R z

Contoh: R = '='

$x = x$

$x = y \rightarrow y = x$

$x = y$ dan $y = z \rightarrow x = z$

- Equivalensi Pada Class

Untuk equivalensi relasi R

equivalensi class adalah $x = \{y : x \; R \; y\}$

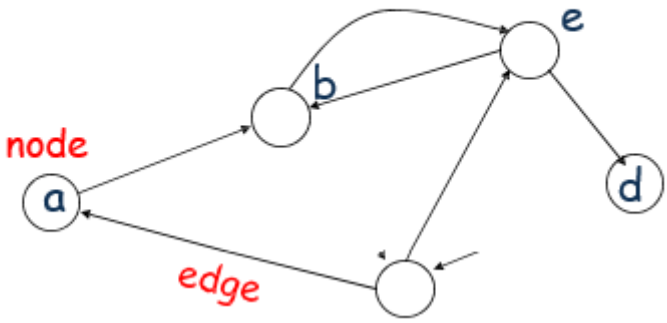
Contoh:

R = { (1, 1), (2, 2), (1, 2), (2, 1),
(3, 3), (4, 4), (3, 4), (4, 3) }

Equivalensi class dari 1 = {1, 2}

Equivalensi class dari 3 = {3, 4}

- Graff

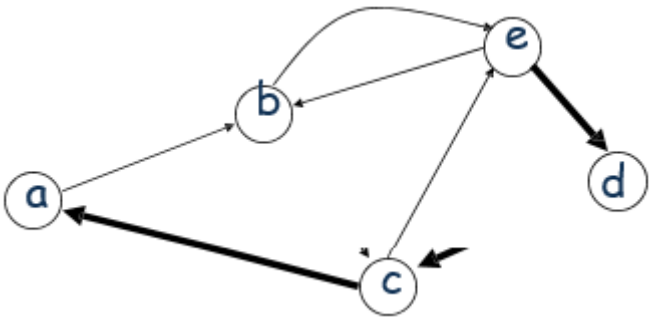


Gambar 15. Graf

- Lintasan

  Lintasan adalah urutan dari edges yang berdekatan

  (e, d), (d, c), (c, a)

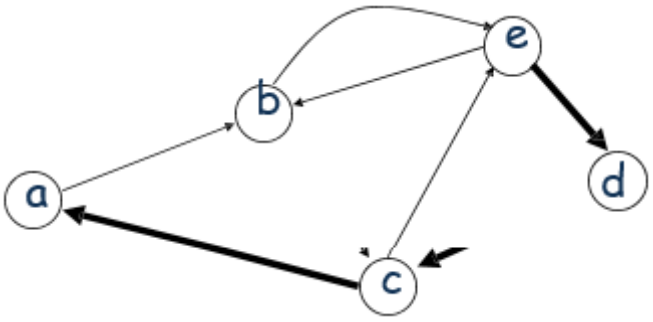


Gambar 16. Lintasan

- Path

  Path adalah lintasan dimana tidak ada edge yang diulang Path

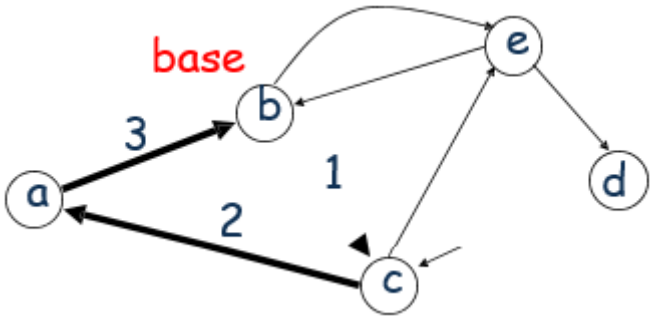
  sederhana : tidak ada node yang berulang



Gambar 17. Path

- Cycle

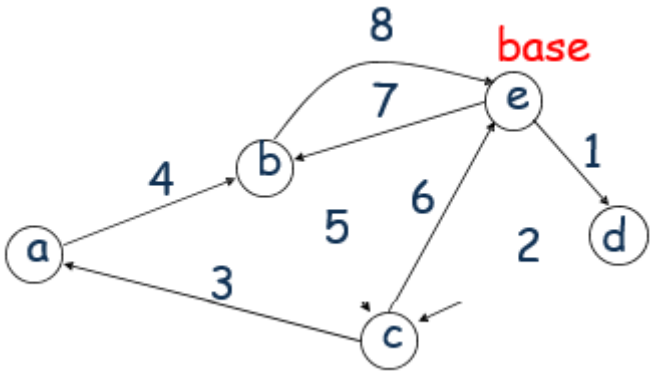
  Cycle adalah lintasan dari node awal kembali ke node awal lagi Cycle sederhana : hanya node awal aja yang berulang



Gambar 18. Cycle

- Lintasan Euler

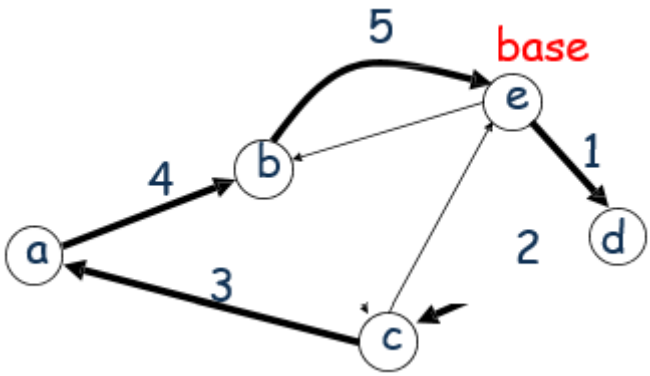
  Sebuah cycle yang melintasi edge satu kali



Gambar 19. Lintasan Euler

- Hamiltonian Cycle

  Lintasan sederhana yang melintasi seluruh node



Gambar 20. Hamiltonian Cycle

BAB II

TEORI BAHASA

2.1 Tujuan Instruksional

A. Tujuan Instruksional Umum

Mahasiswa memahami konsep bahasa secara umum

B. Tujuan Instruksional Khusus

Mahasiswa memahami operasi bahasa

2.2 Obyektif TBA Secara Umum

Membahas model komputasi sebagai mesin abstrak yang dapat didefinisikan secara matematis, mulai dari yang paling sederhana hingga yang powerfull. Model tersebut digambarkan sebagai mesin untuk menjawab masalah keputusan. Masukan string x, Keluaran berupa : ya atau tidak, misalnya : Apakah x adalah bilangan prima? Apakah suatu string s anggota dari bahasa B?

2.3 Konsep Bahasa

A. Simbol

Elemen unik terkecil dari bahasa. Dari suatu bahasa terdapat sejumlah berhingga simbol yang digunakan.

B. Alfabet Simbol

Himpunan berhingga simbol suatu bahasa sebagai alfabet simbol dari bahasa tersebut

C. String

Deretan simbol sebagai suatu entitas. Dalam suatu string bisa ada simbol yang muncul lebih dari 1 kali dan ada simbol yang tidak digunakan. Kumpulan Huruf.

D. Bahasa

Bahas merupakan himpunan dari sejumlah string dari alfabet bahasa tersebut. Subset dari himpunan seluruh kemungkinan string yang dapat dibentuk dari alfabet bahasa tersebut. Himpunan bisa berhingga atau tak berhingga.

E. Grammer

Himpunan dari aturan struktural yang didefinisikan yang berlaku dalam suatu kalimat.

2.4 Notasi Alfabet dan Variabel Simbol/String

Alfabet ditulis dengan lambang Σ. Untuk penyederhanaan pembahasaan alfabet, hanya berisi dua bahkan mungkin satu simbol. Namun secara umum berlaku untuk alfabet yang lebih besar.

Variabel string tertentu sering direpresentasikan sebagai r,s,t, ..., x,y,z. (Huruf kecil di urutan belakang dalam abjad latin). Variabel simbol sering direpresentasikan sebagai a,b,c ..... yaitu huruf kecil pada urutan terdepan abjad. Notasi Variabel Himpunan String dinotasikan dengan huruf kapital pada urutan awal abjad.

L = { 001, 1100, 01, 000001} Contoh String :

"cat", "dog", "house", … terdefinisi pada alphabet:

Σ = { *a,b, c .....z*}

Dalam Konteks Bahasa Pemrograman, sebuah program lengkap adalah string. Keyword, nama variabel, nama fungsi, tanda operasi dan notasi adalah simbol bahasa.

A. Alphabets and Strings

1. Alphabet menggunakan huruf kecil

   S = {*a,b*}

2. String

   a

   ab u = ab

   abba v = bbbaaa

   baba w = abba

   aaabbbaabab

2.5 Operasi String

1. Panjang String

   Suatu string tersusun atas sejumlan n simbol, n ≥ 0.

   Panjang string x ditulis |x|

   Misal x = aabaa

   |x| = 5

   |aab| = 3

   |aaaabbb| = 7

2. String Kosong

   Jika |x| = 0 maka disebut String kosong

   Biasa dinyatakan dengan ε atau λ

   |ε| = 0

3. Konkatenasi

Artinya adalah penyambungan. String x dan y apabila dikonkatenasi akan menjadi string xy

Contoh : x = aaba

y = aa

xy = aabaaa

x = ab

y = ε

xy = ab

Konkatenasi Berulang pada String

Menggunakan notasi pangkat

Contoh : x = ab

Maka x^4 = abababab

$(abba)^{2 =}$ abbaabba

Konkatenasi pada Himpunan String

L1 = {aa, bab}

L2 = {bb,a}

L1L2 = {aabb, aaa, babbb, baba}

4. Reverse

   Merupakan pembalikan string

   Contoh : x = ab x^R = ba

5. Substring

   Substring dari string

   Subsequen dari karakter yg berurutan

   u = abbab

   Substring u :

   ab

   abba

b

bbab

6. Prefix and Suffix

   Prefix - substring di awal

   Suffix - substring di akhir

   String u = abbab

Prefixes	Suffixes
λ	*abbab*
a	*bbab*
ab	*bab*
abb	*ab*
abba	*b*
abbab	λ

7. Operasi (*)

   Himpunan seluruh string yg mungkin dari alphabet/Σ

   Σ^*

   Contoh :

   $\Sigma = \{a,b\}$

   maka :

   $\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots.\}$

8. Operasi (+) Σ^+

   Himpunan seluruh string yang mungkin dari alphabet Σ kecuali λ

   $\Sigma = \{a,b\}$

   $\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa, aab, \ldots.\}$

   $\Sigma^+ = \Sigma^* - \lambda$

   $\Sigma^+ = \{a,b,aa, ab,ba,bb, aaa, aab,....\}$

BAB III

FINITE STATE AUTOMATA

3.1 Tujuan Instruksional

A. Tujuan Instruksional Umum

Mahasiswa memahami konsep FSA

B. Tujuan Instruksional Khusus

Mahasiswa dapat membuat FSA

3.2 Definisi Finite State Automata

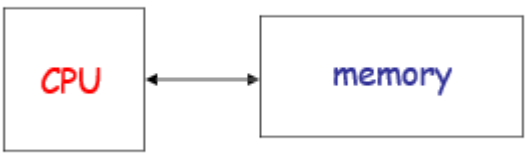
3.3 Sifat Automata

1. Kelakuan mesin bergantung pada rangkaian masukan yang diterima mesin.
2. Setiap saat mesin dapat berada pada status tertentu dan dapat berpindah ke status baru karena ada perubahan input.
3. Rangkaian input dapat dianggap sebagai suatu bahasa yang harus dikenali.
4. Setelah pembacaan input selesai maka mesin akan membuat keputusan.

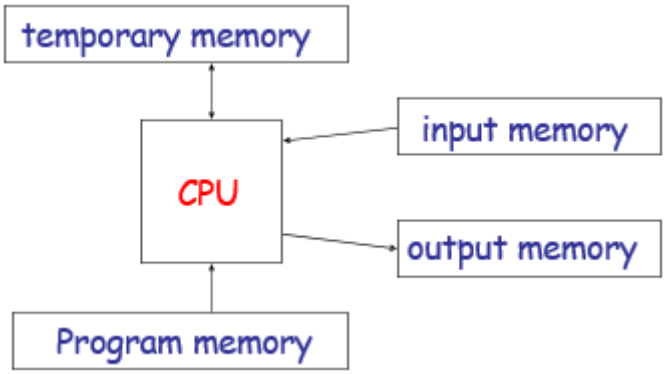
3.4 Komputasi

Membahas model komputasi sebagai mesin abstrak yang dapat didefinisikan secara matematis, mulai dari yang paling sederhana sampai yang *powerfull.*

A. Komputasi

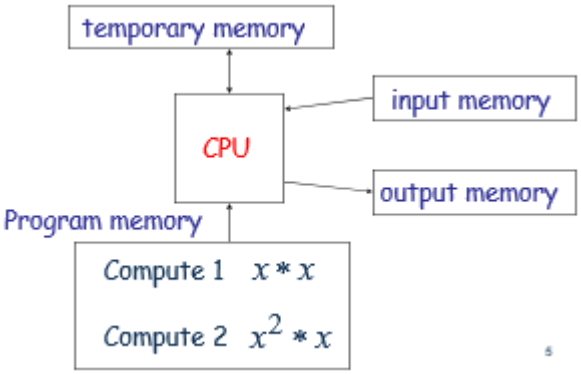


Gambar 21. Proses Komputasi

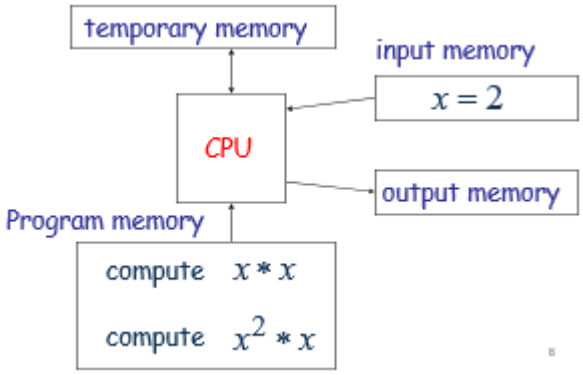


Gambar 22. Komputasi dengan Temporary Memory

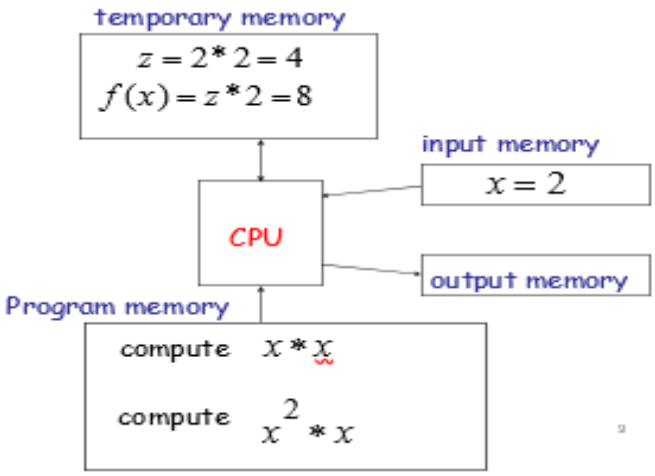
Contoh : $f(x) = x^3$



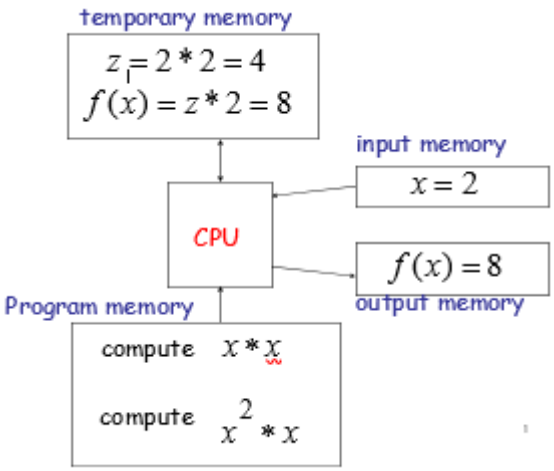
Gambar 23. Proses Komputasi f(x)



Gambar 24. Proses Komputasi f(x), x = 2

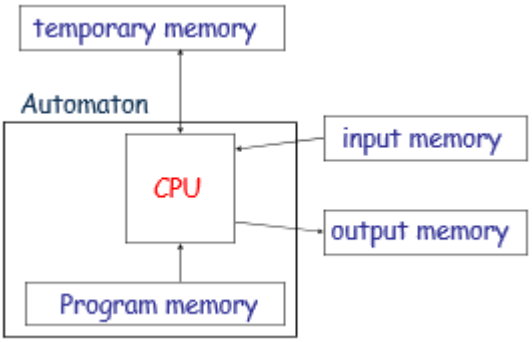


Gambar 25. Proses Komputasi



Gambar 26. Hasil Komputasi

3.5 Automaton



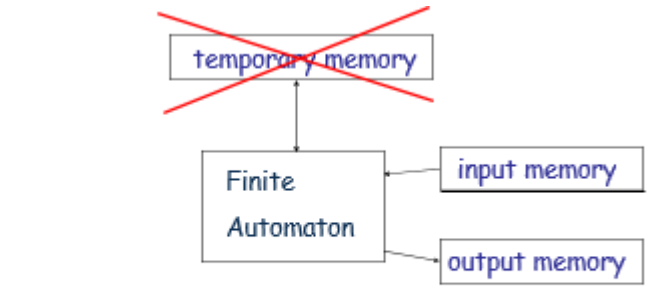
Gambar 27. Automaton

Perbedaan dari beberapa Automata

Tabel 1. Perbedaan Automata berdasarkan temporary memory

Berdasarkan temporary memory	
Finite State Automata	Tidak mempunyai
PushDown Automata	Bentuknya Stack
Turing Machines	Pengaksesan memory random

1. Finite Automaton (FA)

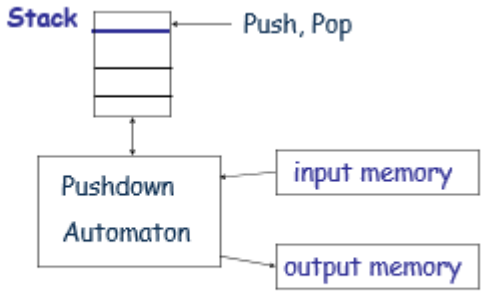


Contoh: Mesin Pencari Kata

(Kekuatan komputasi kecil)

Gambar 28. Finite Automaton

2. Pushdown Automaton (PDA)

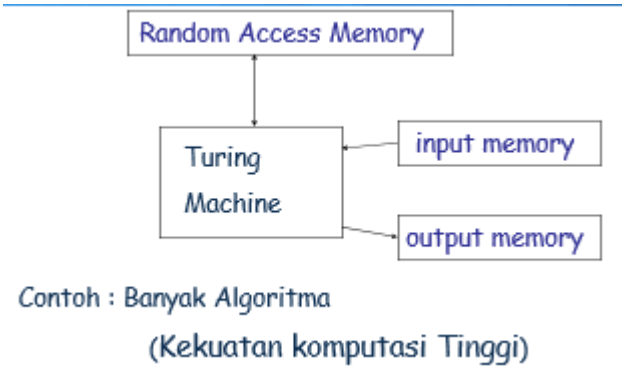


Contoh : Compilers pada bahasa pemrograman

(Kekuatan komputasi medium)

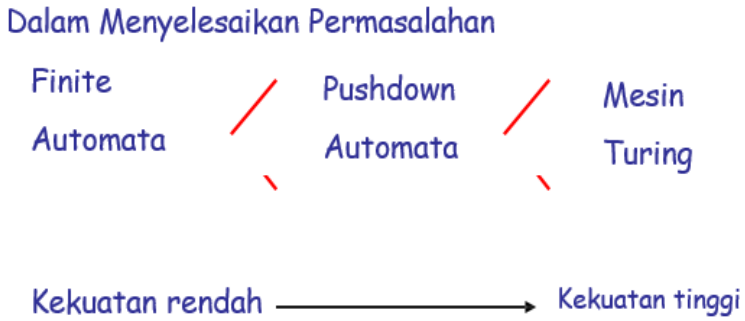
Gambar 29. PDA

3. Mesin Turning



Gambar 30. Mesin Turing

4. Power of Automata



Gambar 31. Power Automata

3.6 Hirarki Chomsky

Penggolongan bahasa menjadi 4 tingkatan :

Tabel 2. Penggolongan Bahasa

Bahasa	Mesin Automata
Reguler	FSA
Bebas Konteks/Context Free	PDA
Context Sensitive	LBA
Unrestricted/Phase Structure/Natural Language	Mesin Turing

3.7 Finite Automata (FA)

A. Jenis FSA

1. Deterministic Finite Automata (DFA)

   Dari suatu state ada tepat **satu** state berikutnya untuk setiap symbol masukan yang diterima.

2. Non-deterministic Finite Automata (NFA)

   Dari suatu state ada **0, 1 atau lebih** state berikutnya untuk setiap simbol masukan yang diterima.

Suatu *FSA* memiliki *state* yang banyaknya berhingga, dan dapat berpindah-pindah dari suatu *state* ke *state* lain.

Konfigurasi FA $M = (Q, S, d, q_0, F)$

Q : Himpunan states/kedudukan

S : Himpunan symbol input/masukan/ alphabet

d : Fungsi Transisi

q_0 : State Awal

F : State akhir

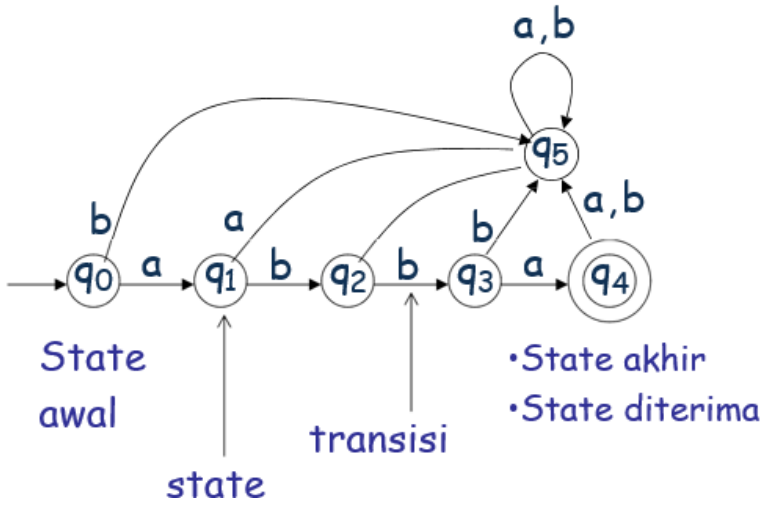
B. Cara kerja Finite Automata

1. Mesin membaca memori masukan berupa tape yaitu 1 karakter tiap saat (dari kiri ke kanan) menggunakan head baca yang dikendalikan oleh kotak kendali state berhingga dimana pada mesin terdapat sejumlah state
2. Finite Automata selalu dalam kondisi yang disebut state awal pada saat Finite Automata mulai membaca tape.
3. Perubahan state terjadi pada mesin ketika sebuah karakter berikutnya dibaca.
4. Ketika head telah sampai pada akhir tape dan kondisi yang ditemui adalah state akhir, maka string yang terdapat pada tape dikatakan diterima Finite Automata (String-string merupakan milik bahasa bila diterima Finite Automata bahasa tersebut). FA untuk mengenali Bilangan Cacah

C. Graph Transisi

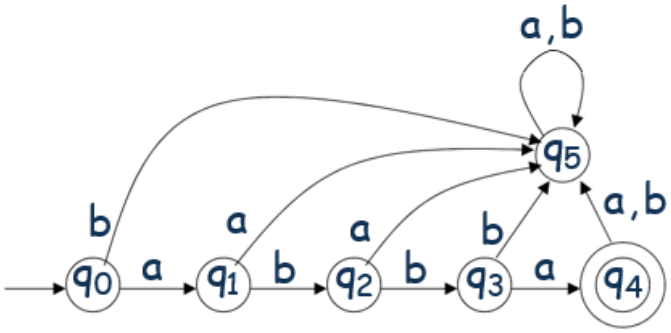
Input Alphabet Σ

Σ = {a,b}

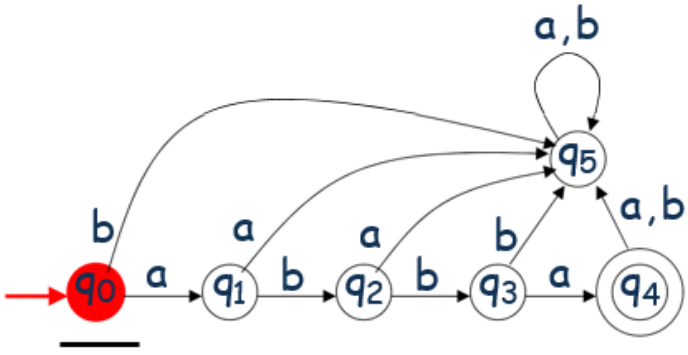


D. Himpunan State = *Q*

$Q = \{q_0, q_1, q_2, q_3, q_4, q_5\}$

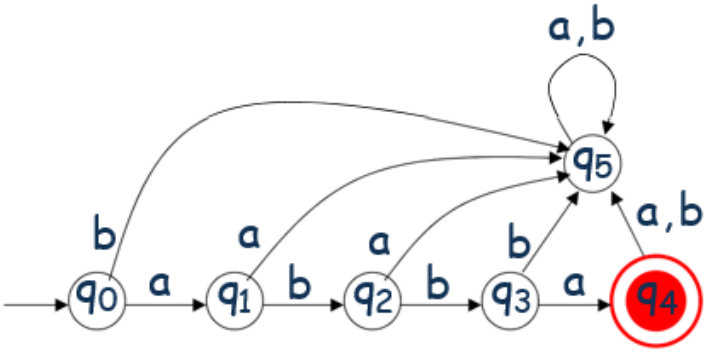


1. State awal = q_0



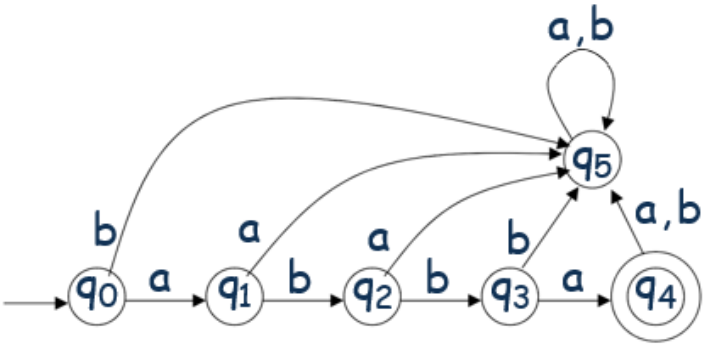
2. State akhir = F

F = {q_4}



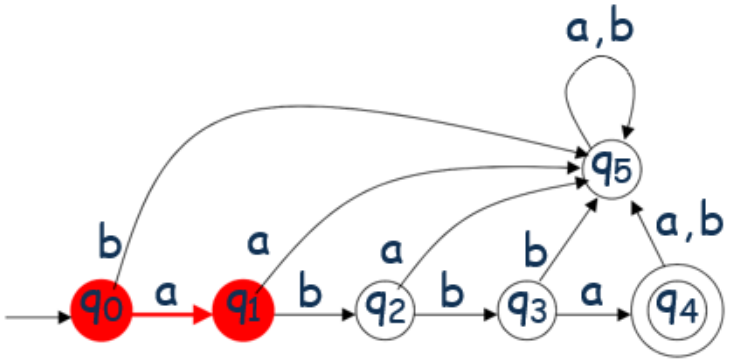
3. Fungsi Transisi = δ

δ : Q x Σ → Q



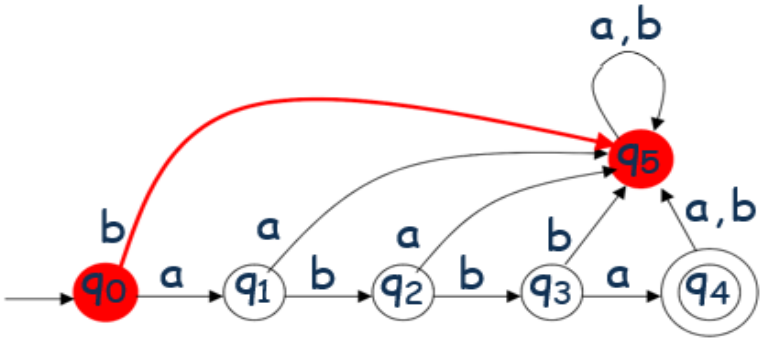
δ (q_0 , a) = q_1

Dibaca : State qo mendapat inputan a menuju ke q_1

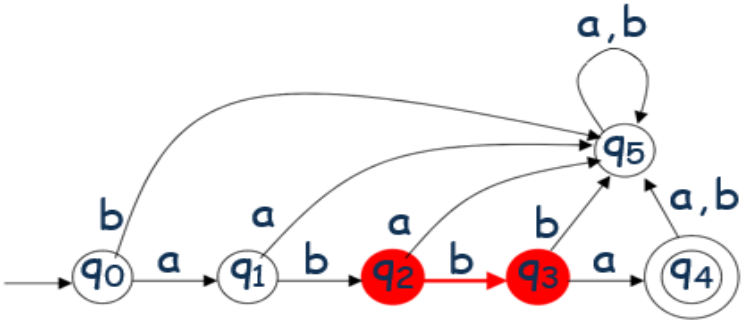


δ (q_0 ,b) = q_5

Dibaca : State qo mendapat inputan a menuju ke q_5

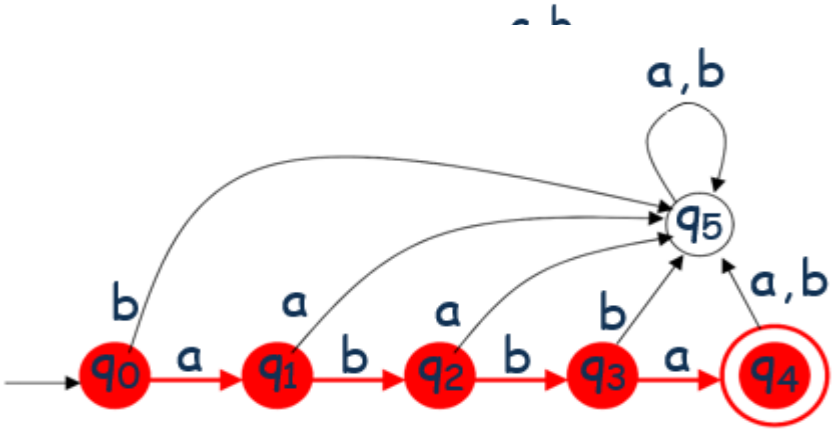


$\delta\ (q_2, b) = q_3$



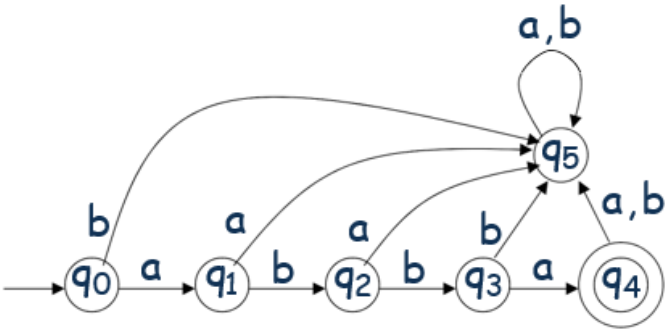
4. Fungsi Transisi δ (Digambarkan dengan tabel transisi)

δ	a	b
q_0	q_1	q_5
q_1	q_5	q_2
q_2	q_5	q_3
3	4	5
q_4	q_5	q_5
q_5	q_5	q_5

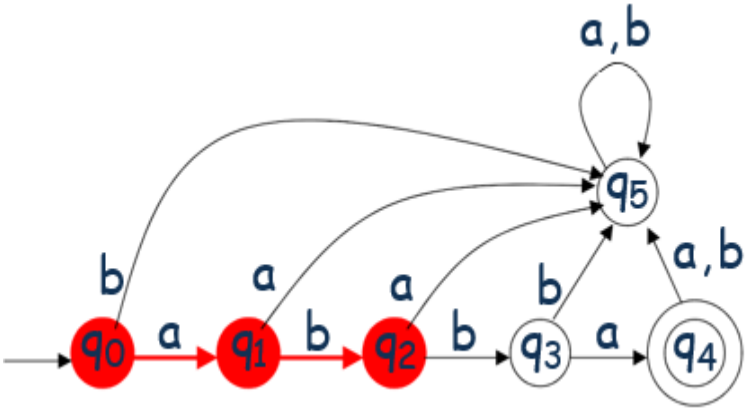


5. Memperpanjang Fungsi Transisi δ*

δ * : Q x Σ* → Q

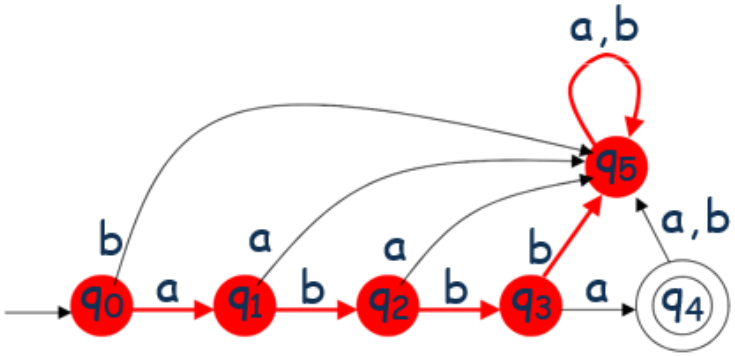


δ * (q_0 , ab) = q_2



δ * (q_0 , $abba$) = q_4

d * (q0 , $abbbaa$) = q5



E. Bahasa yang diterima oleh FA

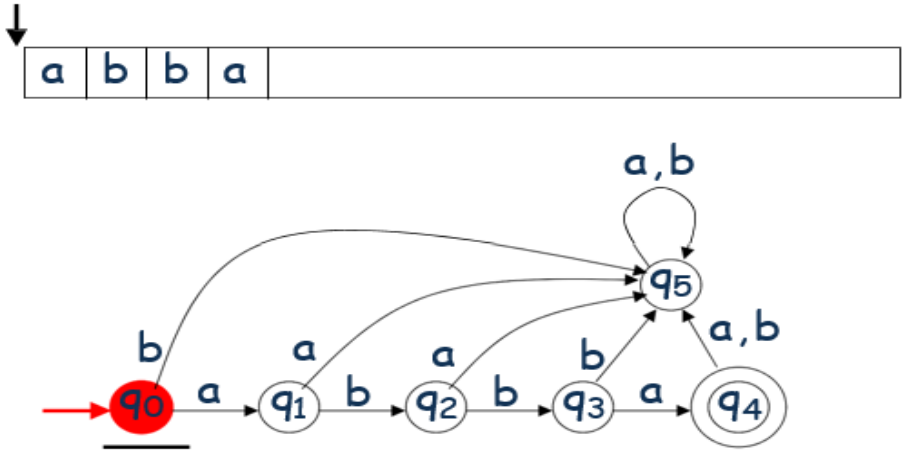
FA =*M*

Bahasa L(M) yang terdiri dari seluruh inputan string yang diterima oleh M .

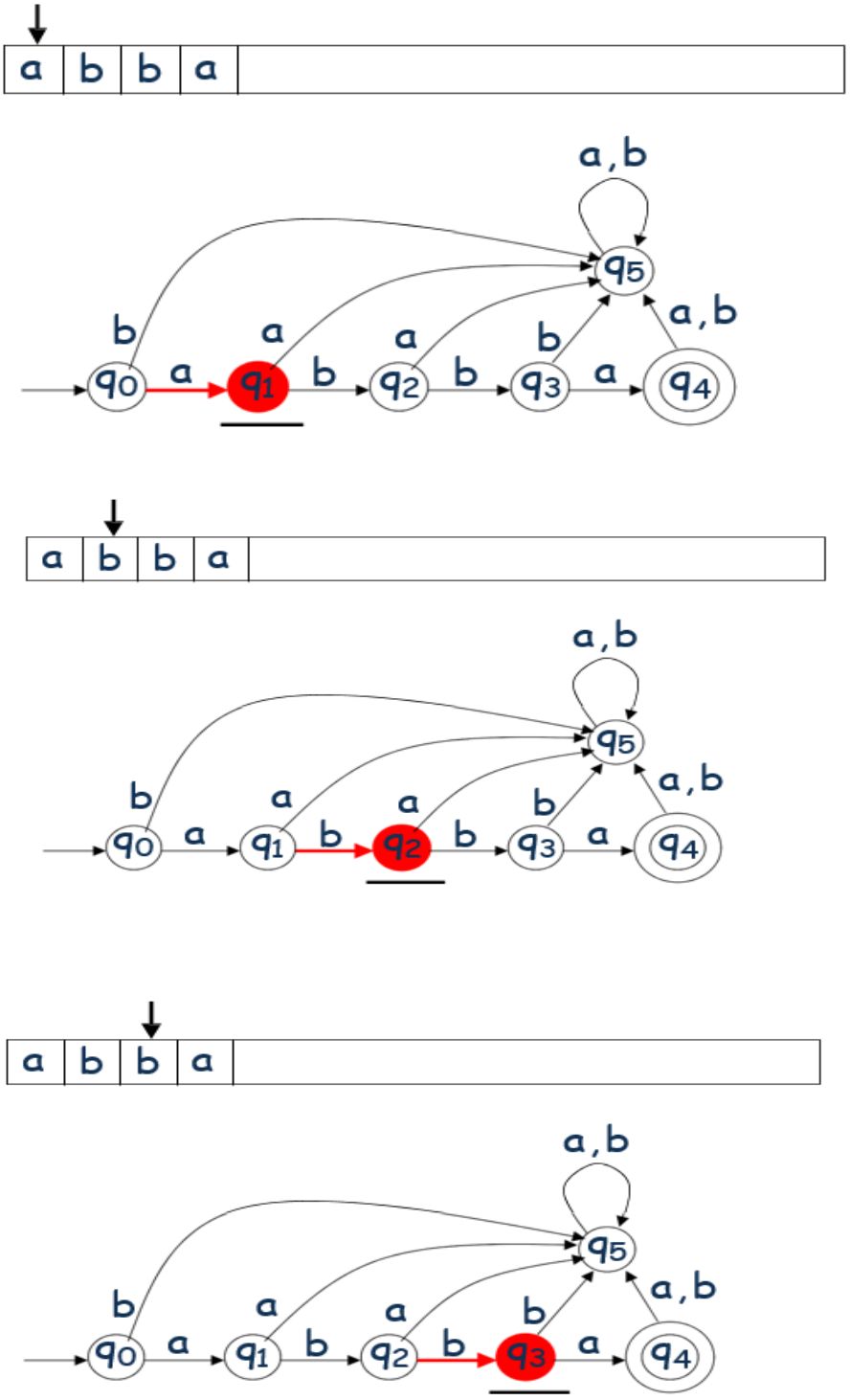
L(M) = string yang dibawa oleh M untuk dapat diterima oleh state.

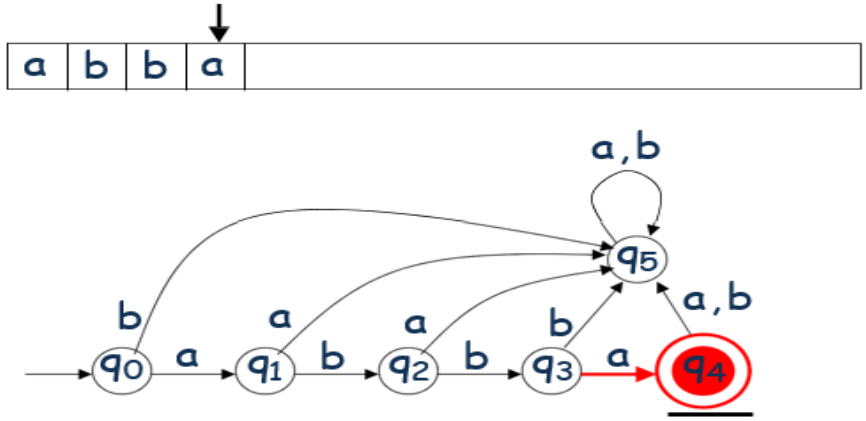
1. Inisial Configurasi (Pertama Kali)

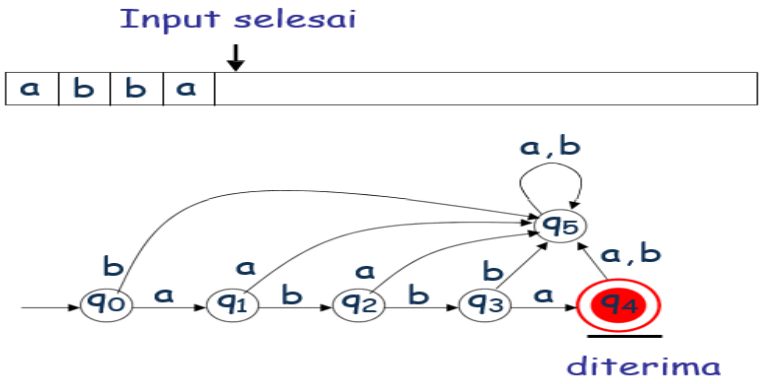
Input String



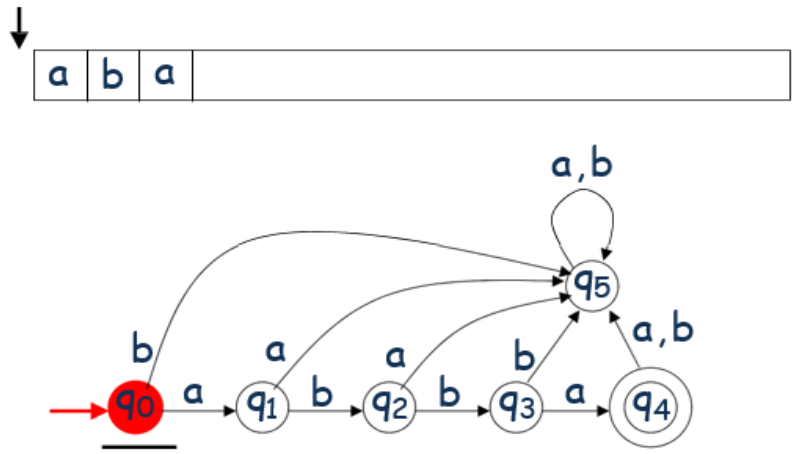
Membaca Input

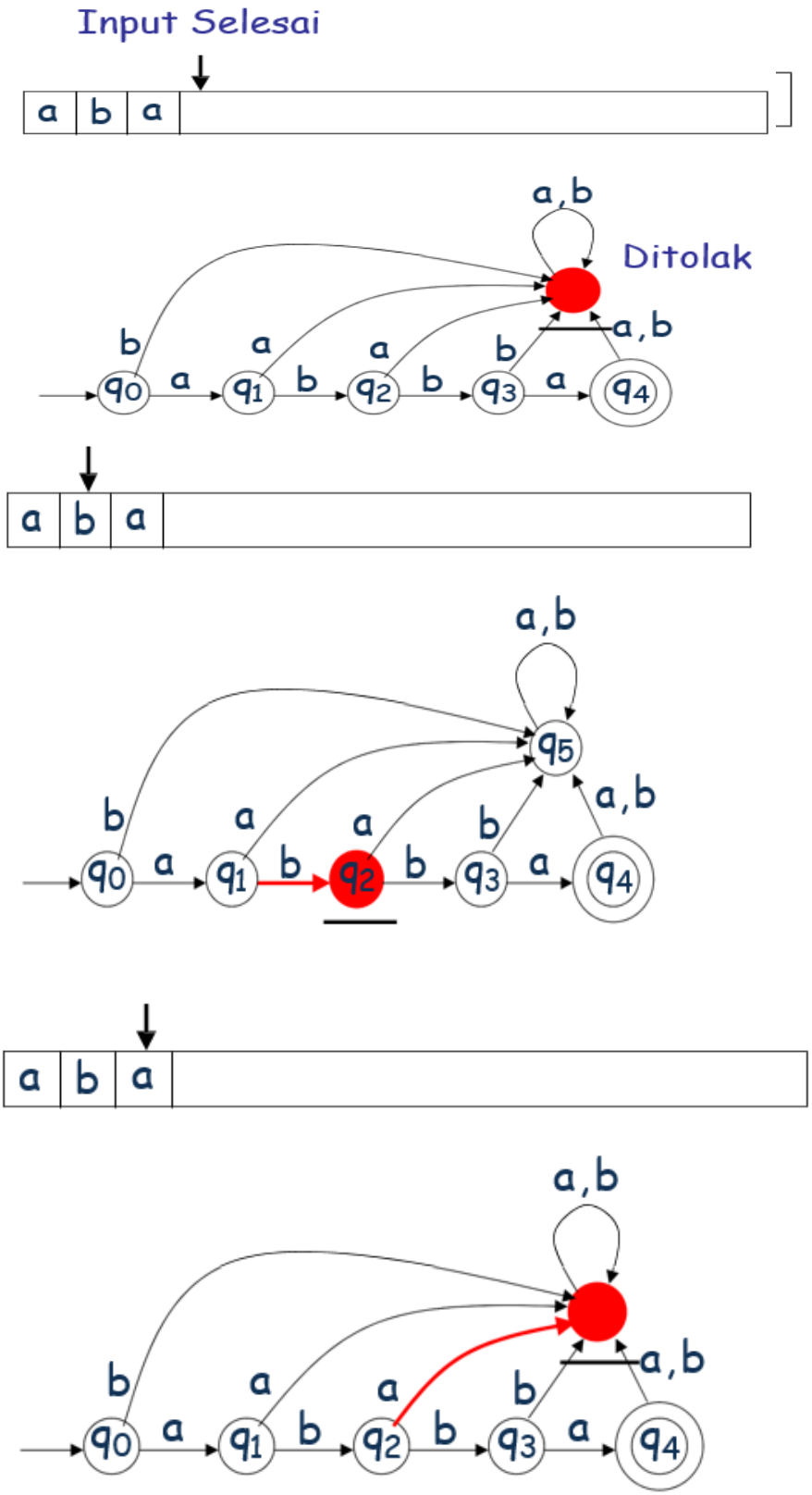

a b b a
a,b
q5
b
a
a
b
a,b
q0
a
q1
b
q2
b
q3
a
q4
a b b a
a,b
q5
b
a
a
b
a,b
q0
a
q1
b
q2
b
q3
a
q4
a b b a
a,b
q5
b
a
a
b
a,b
q0
a
q1
b
q2
b
q3
a
q4





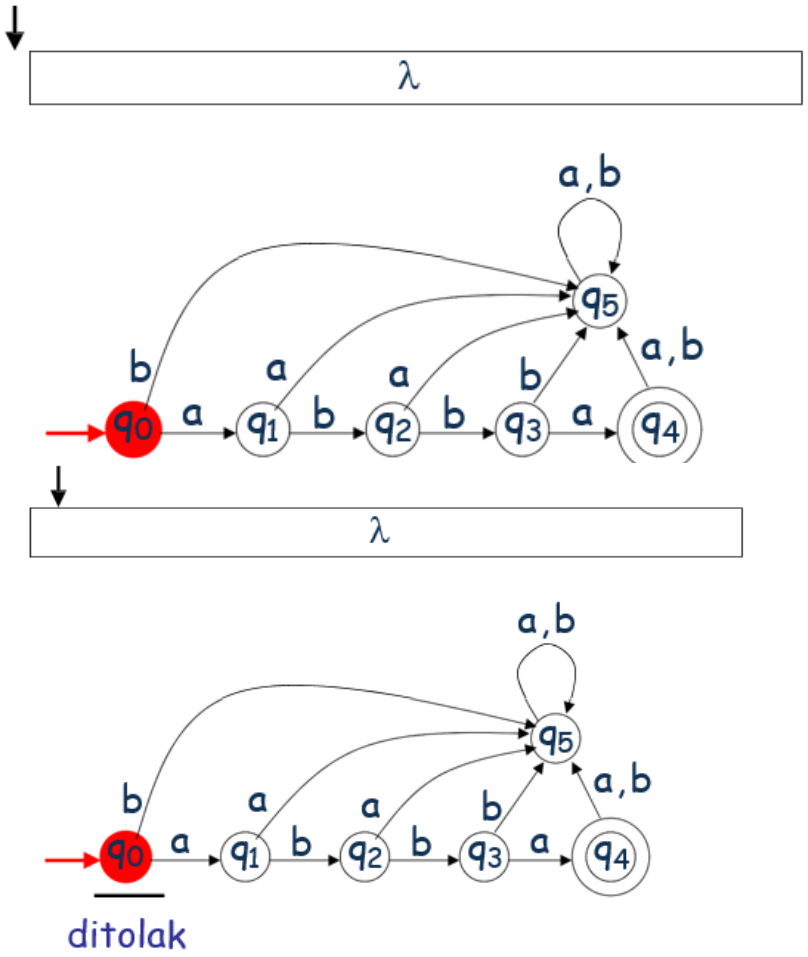
F. Inputan ditolak



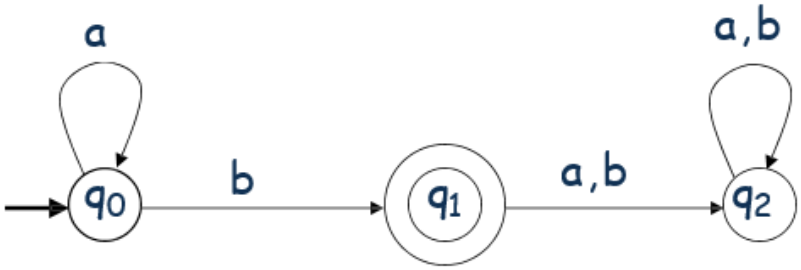

Input Selesai
a b a
a,b
Ditolak
b
a
a
b
a,b
q0
q1
q2
q3
q4
a b a
a,b
q5
b
a
a
b
a,b
q0
q1
q2
q3
q4
a b a
a,b
b
a
a
b
a,b
q0
q1
q2
q3
q4

G. Diterima atau Ditolak? Jika Inputannya λ.

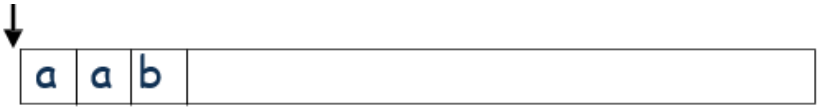
State awal

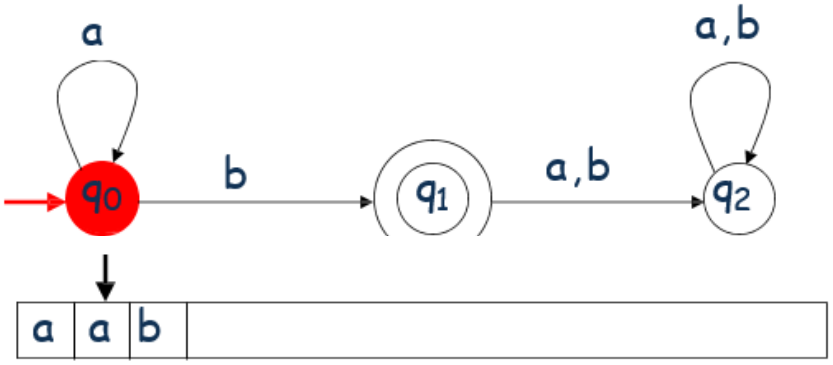


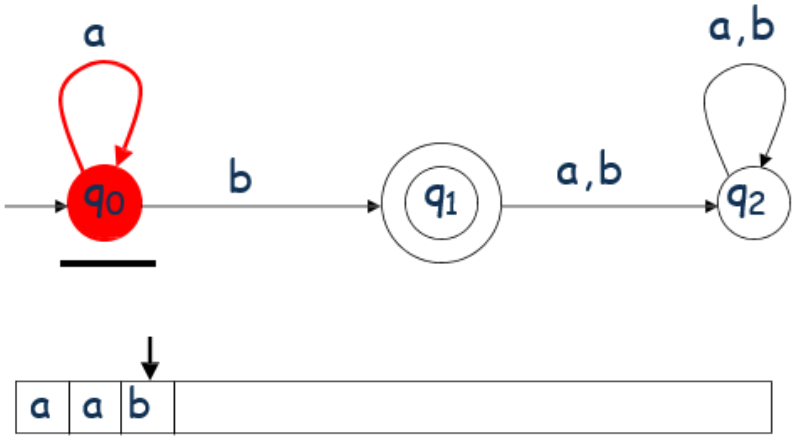
H. Bahasa yang digunakan ?

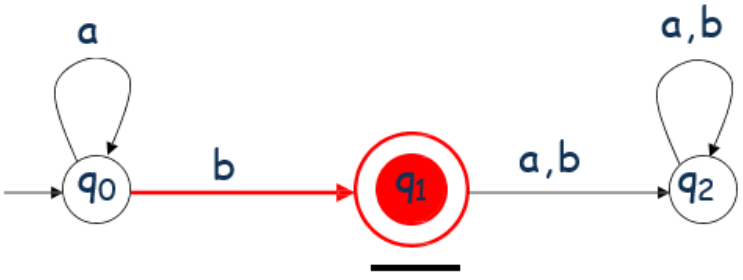


Contoh diterima:

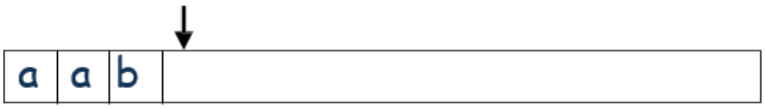


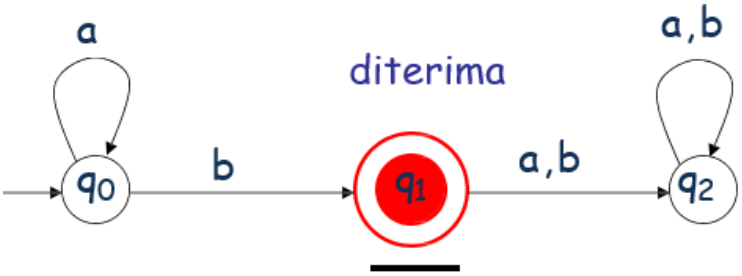




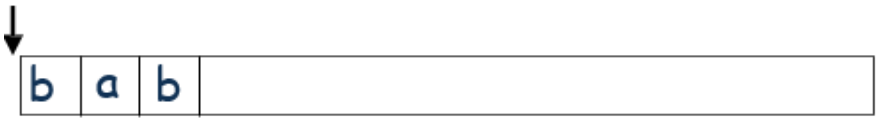


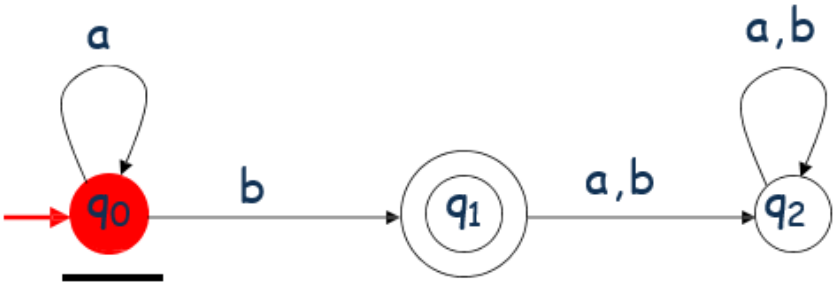
Input selesai

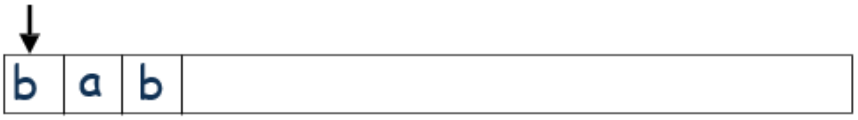


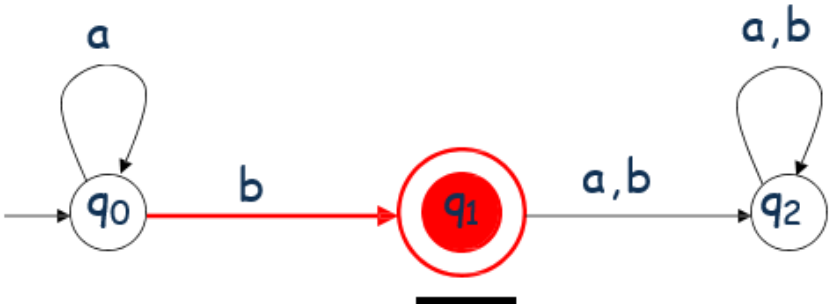


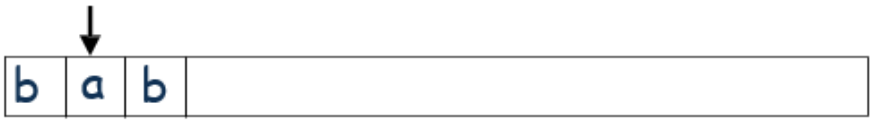
Contoh ditolak :

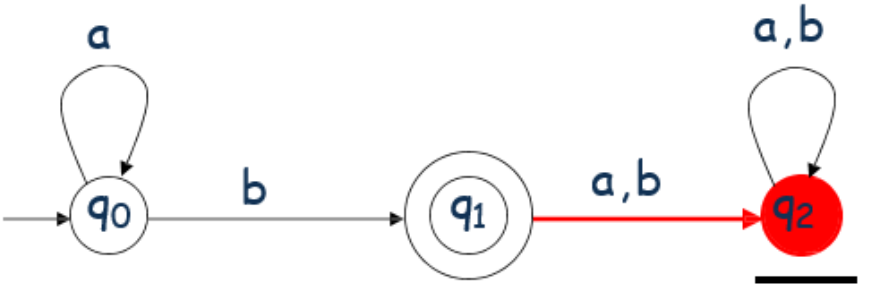


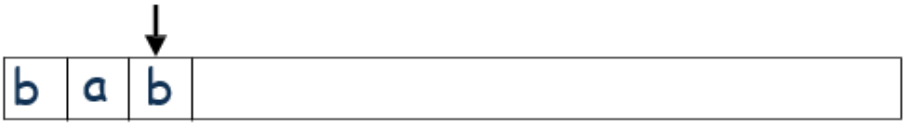


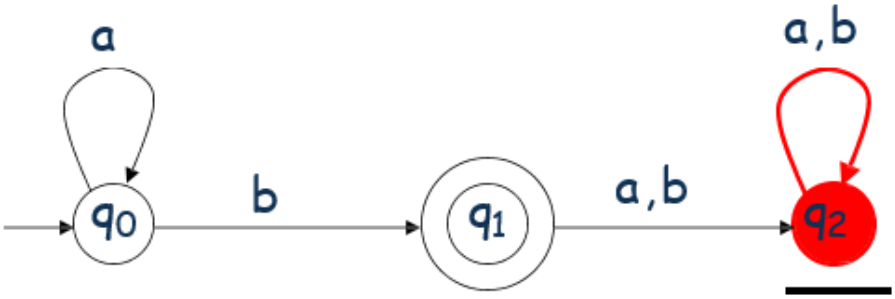




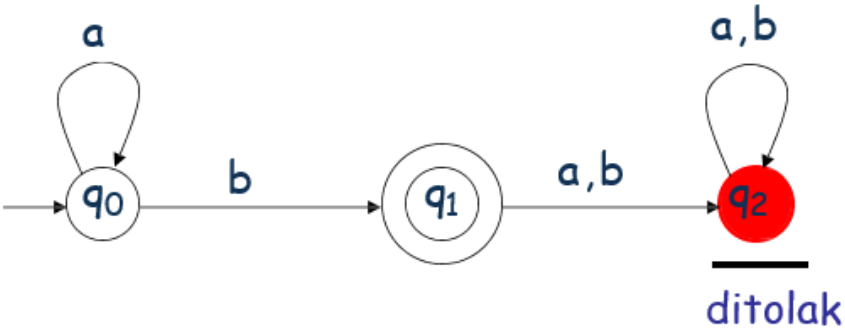






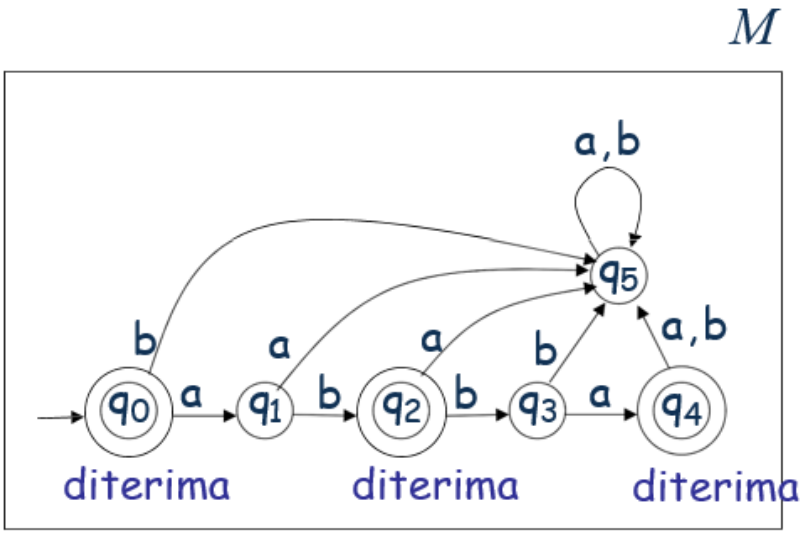


Input selesai

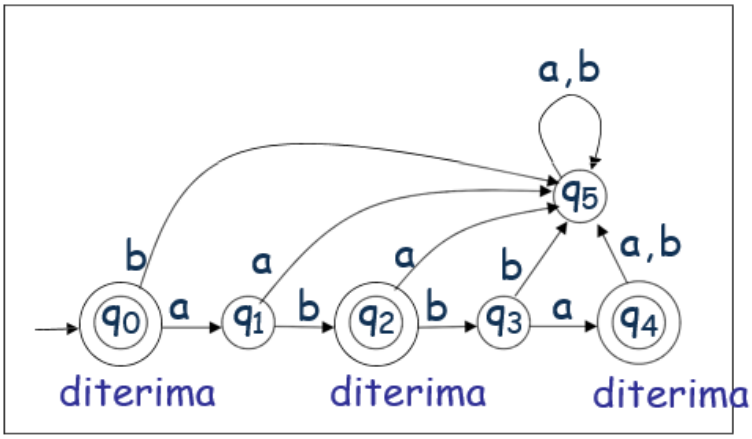


ditolak

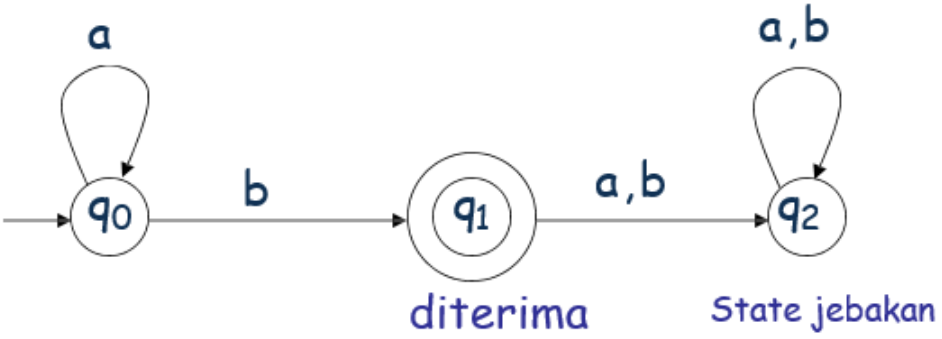
Contoh 1 L(M) = ?



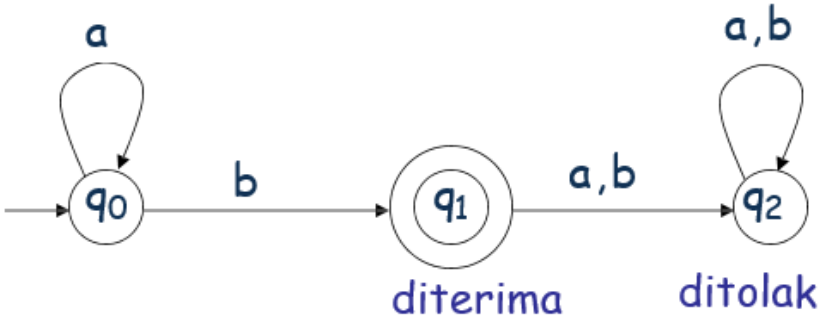
$L(M) = \{\lambda, ab, abba\}$



Contoh 2 L(M) = ?



Contoh 3 $L(M) = \{a^n b : n \geq 0\}$



CERMATILAH :

Pengamatan :jika berjalan dari q ke q', diberi nama W. maka :

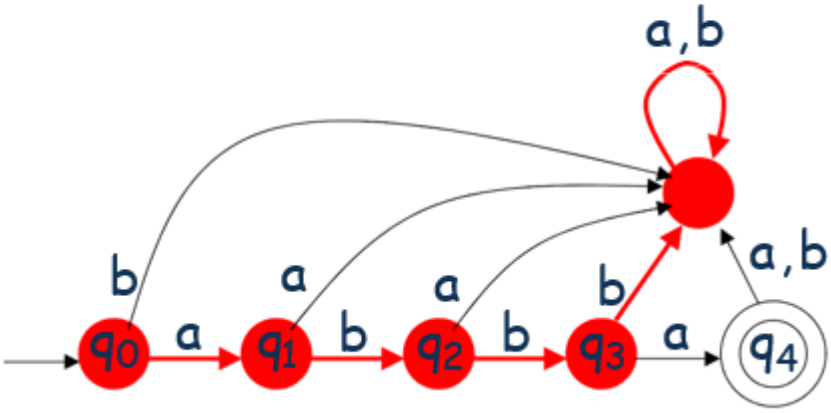
$$\delta^*(q, w) = q'$$

$$w = \sigma_1\sigma_2\cdots\sigma_k$$

Contoh : jika berjalan dari q_0 ke q_5, maka diberi nama *abbbaa*

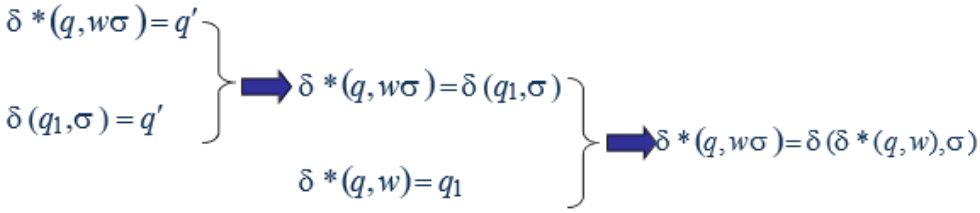
$$\delta^*(q_0, abbbaa) = q_5$$

$$\delta^*(q_0, abbbaa) = q_5$$

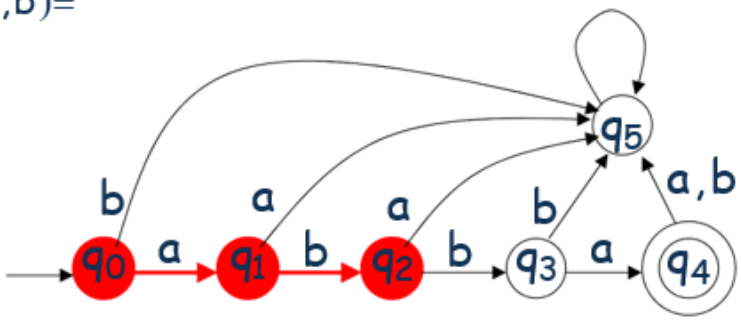


I. Definisi Rekursif

$$\delta^*(q,\lambda) = q$$
$$\delta^*(q, w\sigma) = \delta(\delta^*(q, w), \sigma)$$



$\delta^*(q_0, ab) =$
$\delta(\delta^*(q_0, a), b) =$
$\delta(\delta(\delta^*(q_0, \lambda), a), b) =$
$\delta(\delta(q_0, a), b) =$
$\delta(q_1, b) =$
q_2



Bahasa yang bisa diterima oleh FA

Untuk sebuah FA $M = (Q, \Sigma, \delta, q_0, F)$

Bahasa yang diterima : M

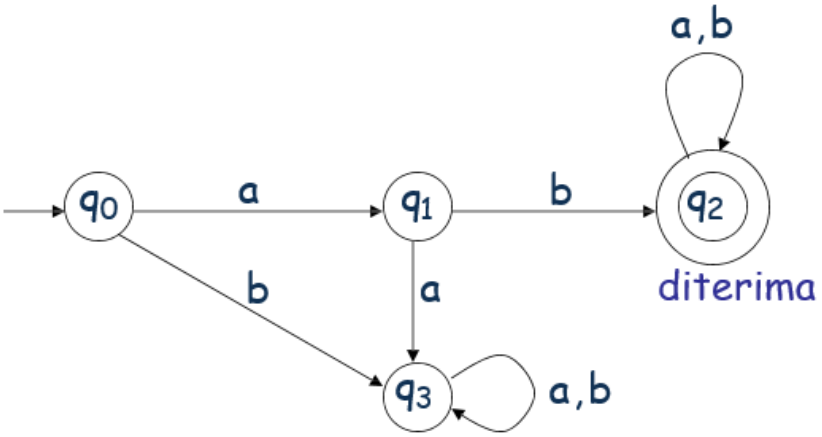
$$L(M) = \{w \in \Sigma^* : \delta^*(q_0, w) \in F\}$$

Bahasa yang ditolak oleh M adalah :

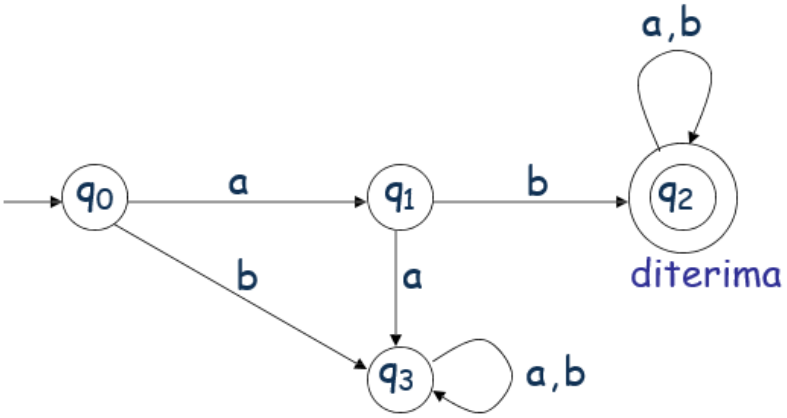
$$\overline{L(M)} = \{w \in \Sigma^* : \delta^*(q_0, w) \notin F\}$$

L(M) ?

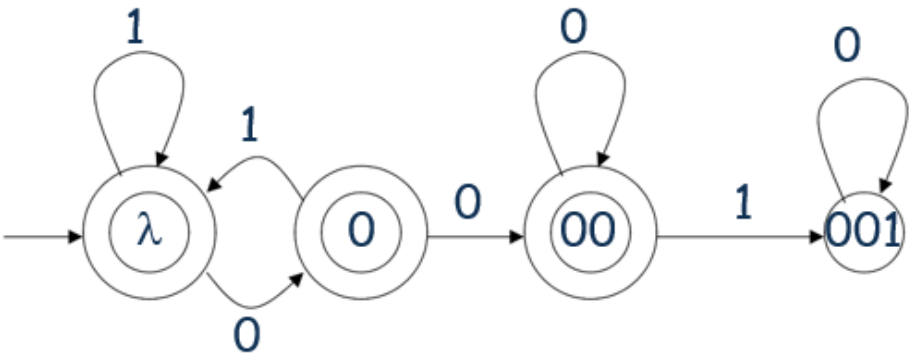
a b



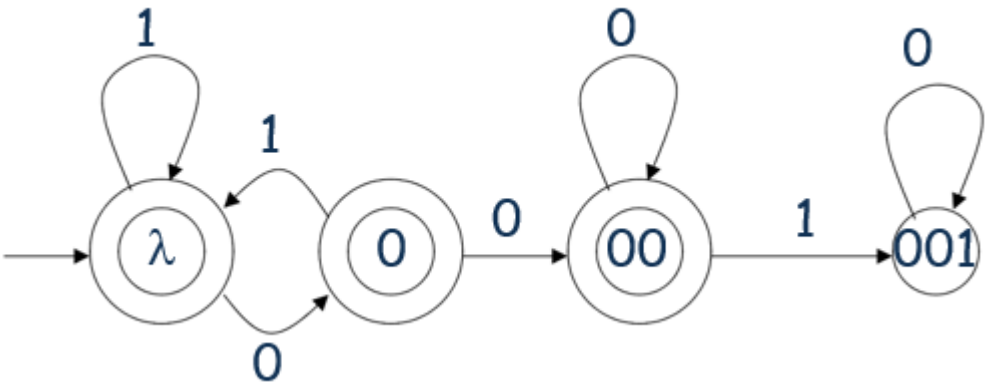
Contoh : L(M) = { seluruh strings dengan prefik ab }



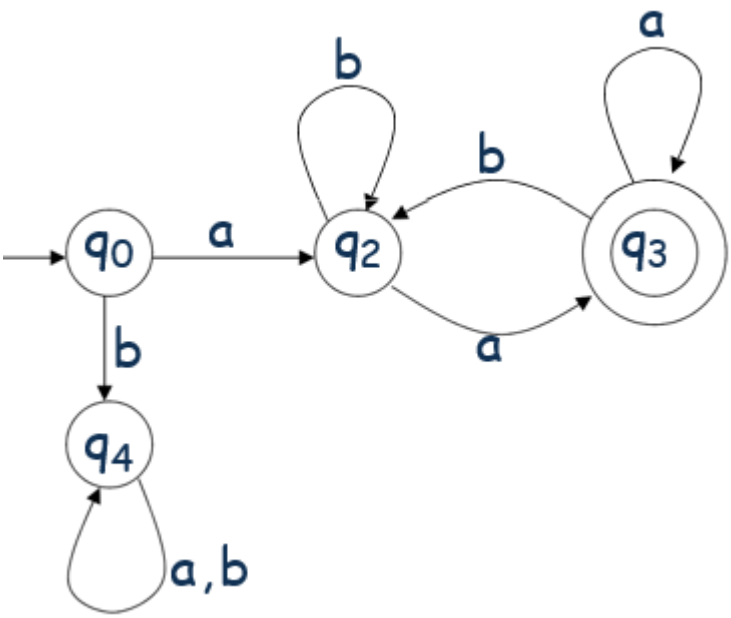
L(M) = ?



Contoh L(M) = { seluruh string tanpa Substring 001}



Contoh $L(M) = \{a\ w\ a : w \in \{a,b\}^*\}$



BAB IV

NON-DETERMINISTIC FINITE AUTOMATA

4.1 Tujuan Instruksional

A. Tujuan Instruksional Umum

Mahasiswa memahami konsep NFA

B. Tujuan Instruksional Khusus

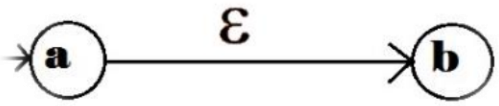
Mahasiswa dapat membuat NFA

4.2 Definisi NFA

4.3 NFA dengan ε-move

Merupakan mesin NFA dimana sebuah state diperbolehkan menuju state yang lain tanpa membaca input.

ε = transisi ε → dimana ε =“*empty*”



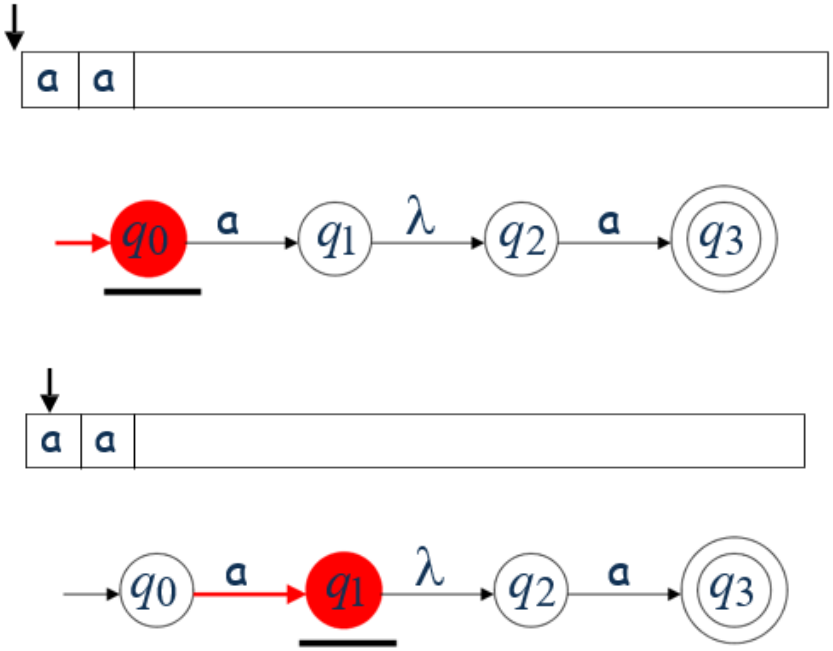
Penjelasan :

1) ε -move atau l (ε di sini bisa dianggap sebagai 'empty')
2) Pada mesin NFA kita bisa mengubah state state tanpa harus bergantung pada suatu inputan, cukup dengan menggunak ε move, kita bisa menuju ke state state yg ada,
3) ε move ini juga disebut dengan transisi
4) Tidak bergantung pada suatu input ketika melakukan transisi atau perpindahan.
5) Agar kita bisa mudah dalam mengkombinasikan finite state automata.

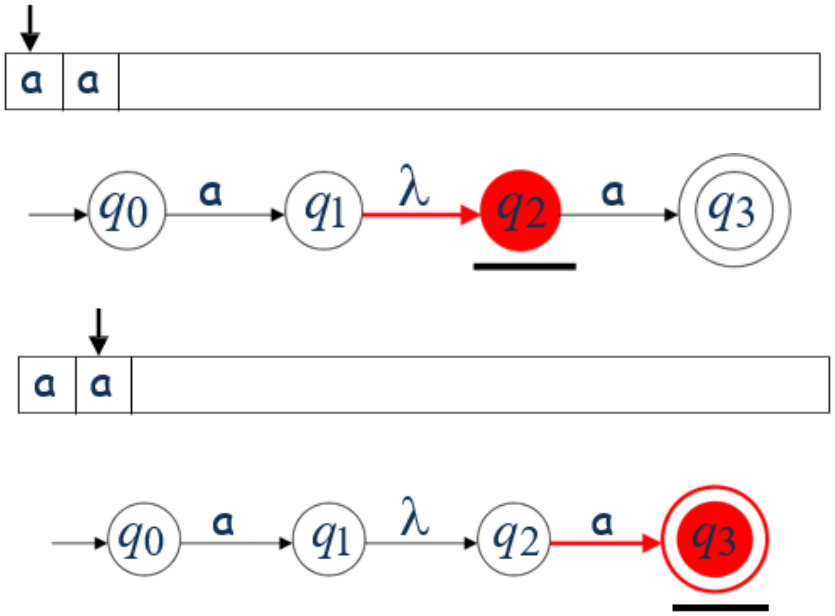
4.4 Transisi Lambda/Epsilon (λ/ε)

Adalah Transisi yang diperbolehkan melakukan perubahan state tanpa mendapatkan inputan.

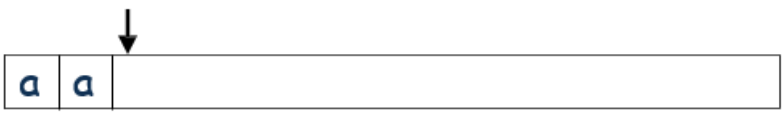
Contoh :

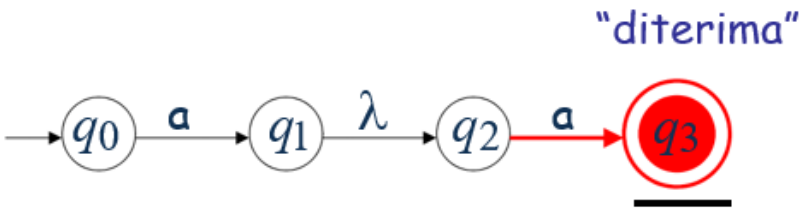


Ada pergerakan tapi tidak ada inputan



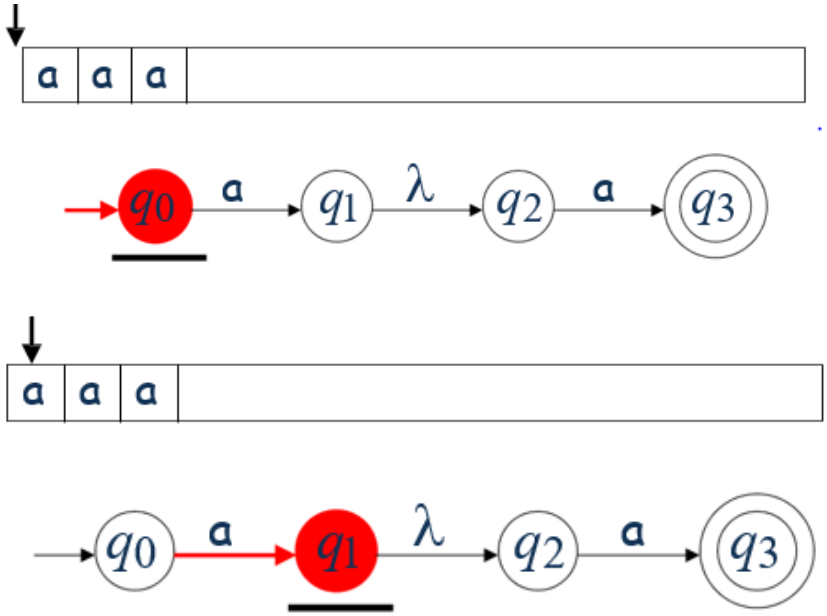
Inputan terselesaikan



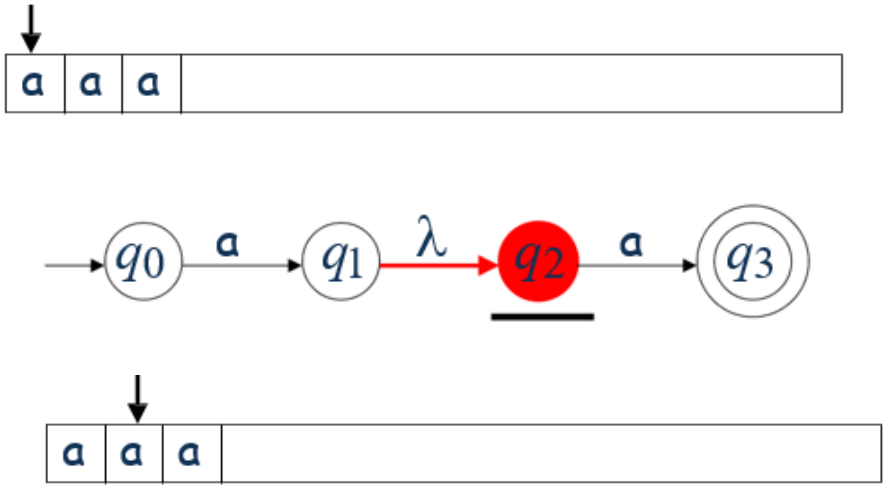


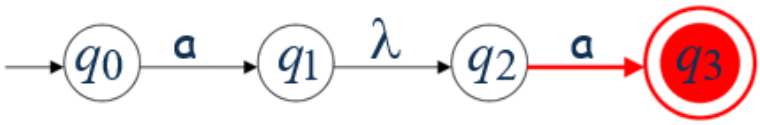
String aa diterima.

Contoh 2:



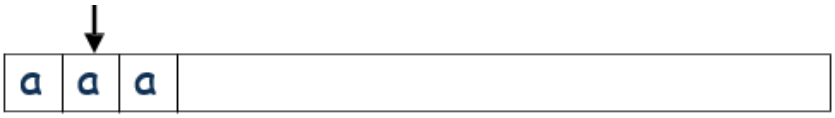
Ada pergerakan namun tidak ada inputan





Tidak ada Transisi:
automata error

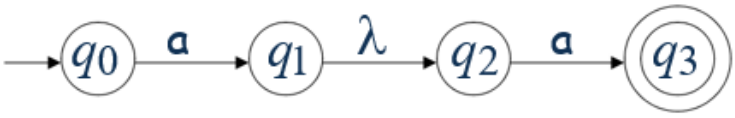
Inputan tidak terselesaikan



"ditolak"

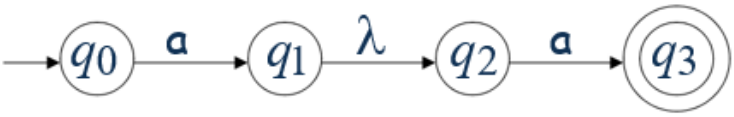
String aaa ditolak

L (M) ?

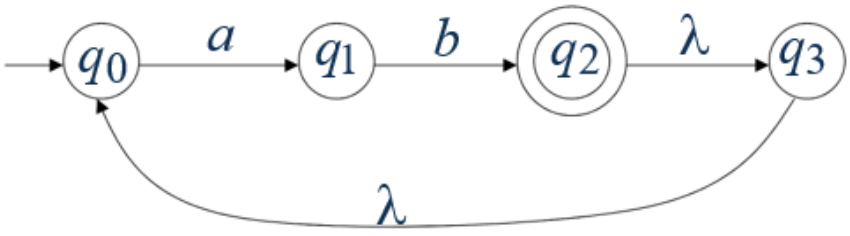


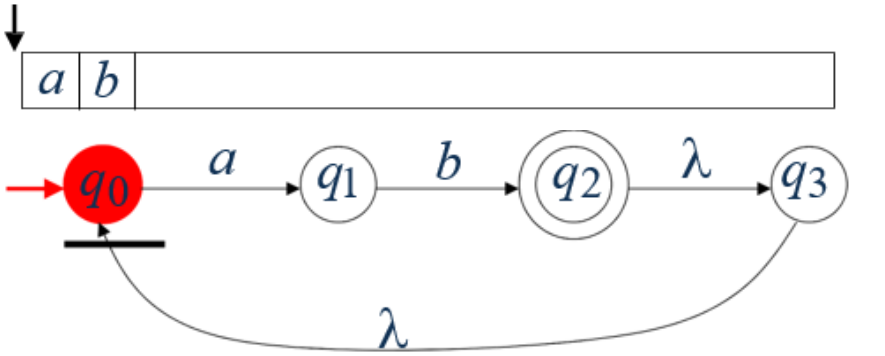
Bahasa yang bisa diterima :

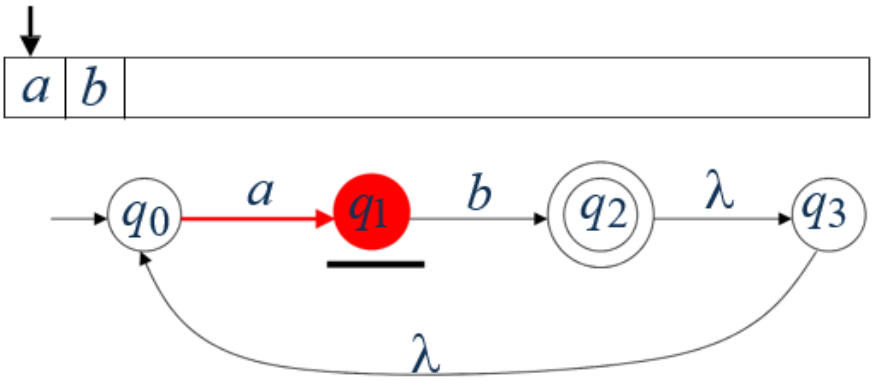
L = {aa}

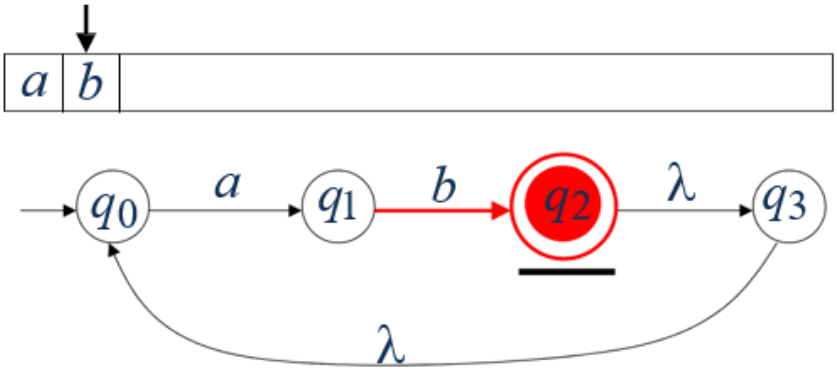


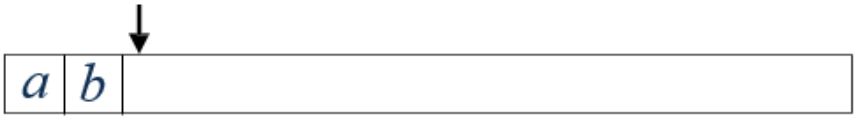
Contoh 3 :



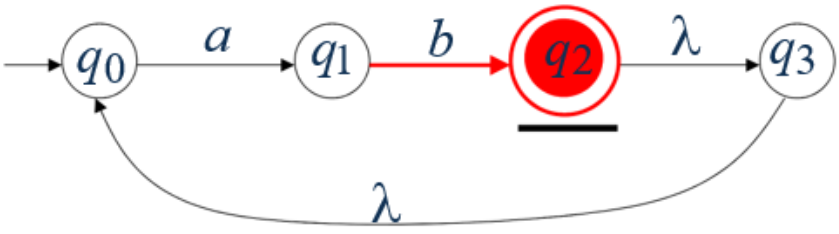

a b
q0
a
q1
b
q2
λ
q3
λ


a b
q0
a
q1
b
q2
λ
q3
λ

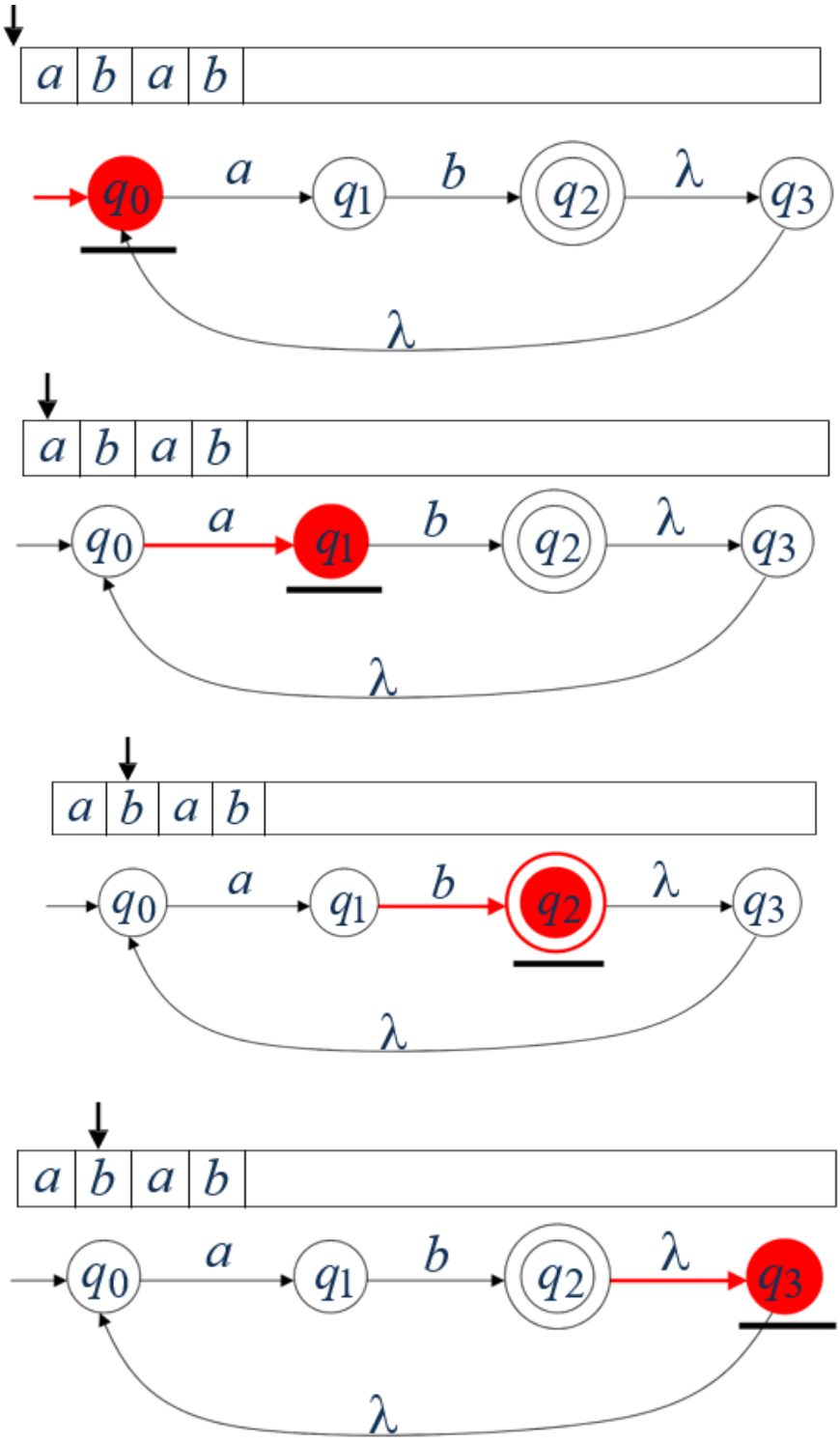

a b
q0
a
q1
b
q2
λ
q3
λ

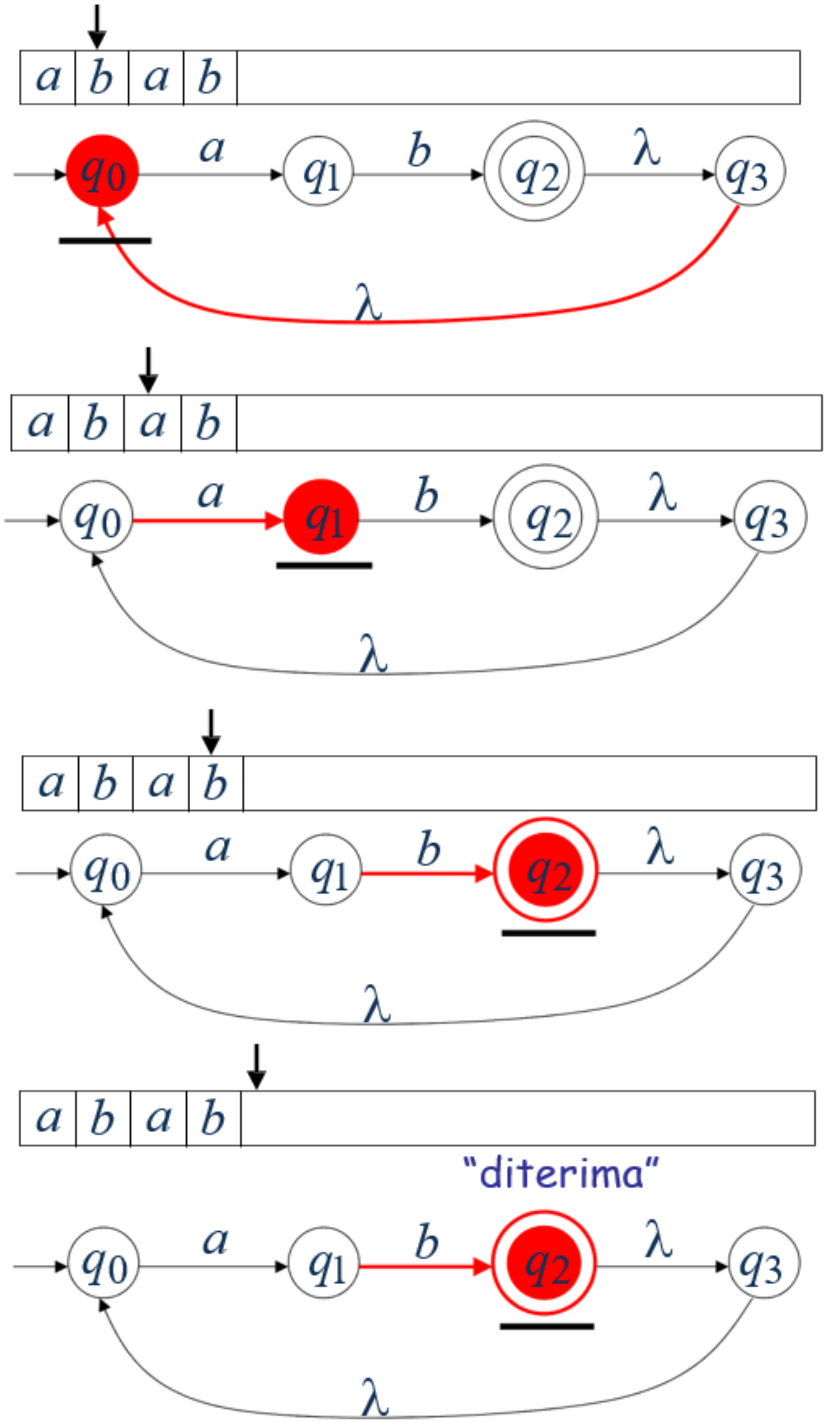

a b

"diterima"


q0
a
q1
b
q2
λ
q3
λ

Inputan String Lain :

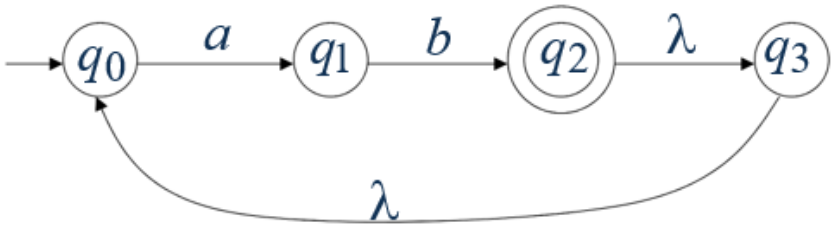




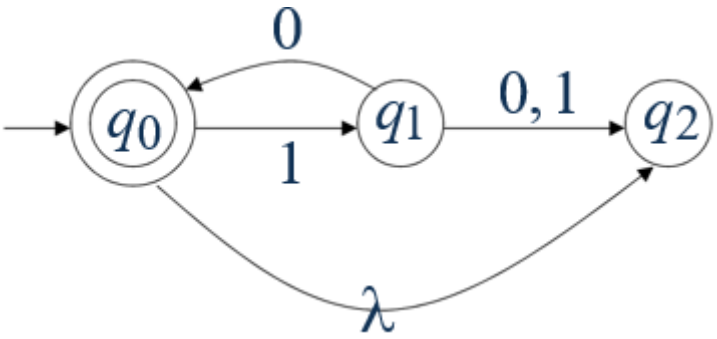
Bahasa yang diterima :

$L = \{ab, abab, ababab, ...\}$

$= \{ab\}^+$



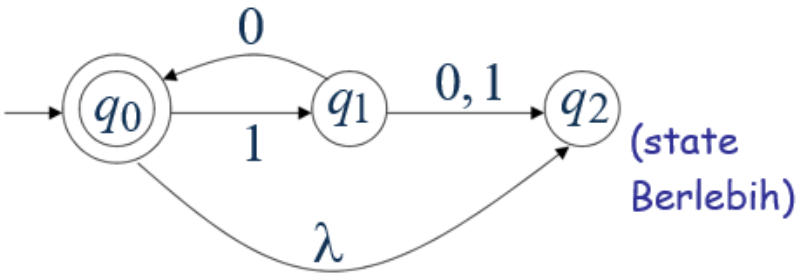
Contoh 4 :



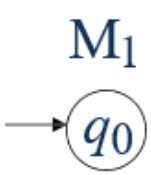
Bahasa yang diterima

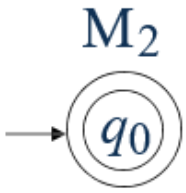
$L(M) = \{\lambda, 10, 1010, 101010, \ldots\}$

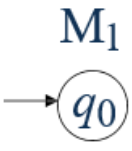
$= \{10\}^*$

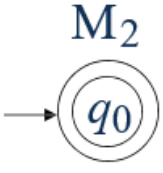


Otomata sederhana : Bahasa yang diterima?







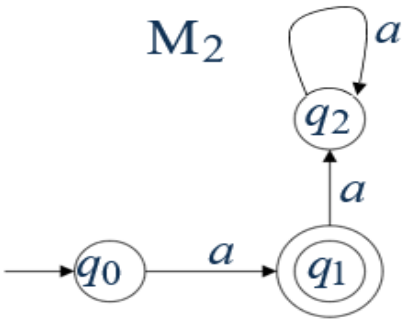


$L(M_1) = \{\}$

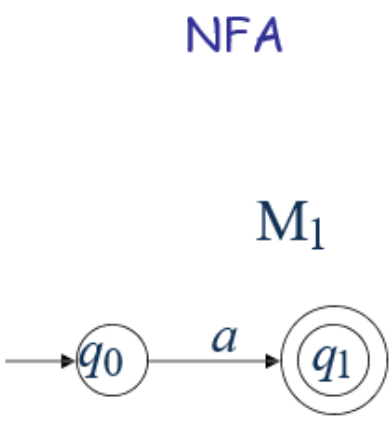
$L(M_2) = \{\lambda\}$

5. λ-transisi pada deterministik automata?

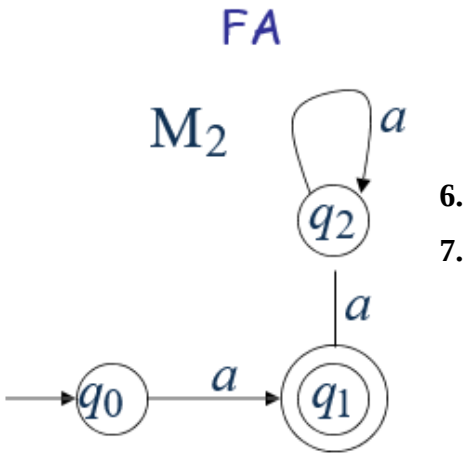
NFA lebih menarik karena : Kemudahan dalam mengekspresikan bahasa disbandingkan dengan FA



$$L(M_2) = \{a\}$$



$$L(M_1) = \{a\}$$



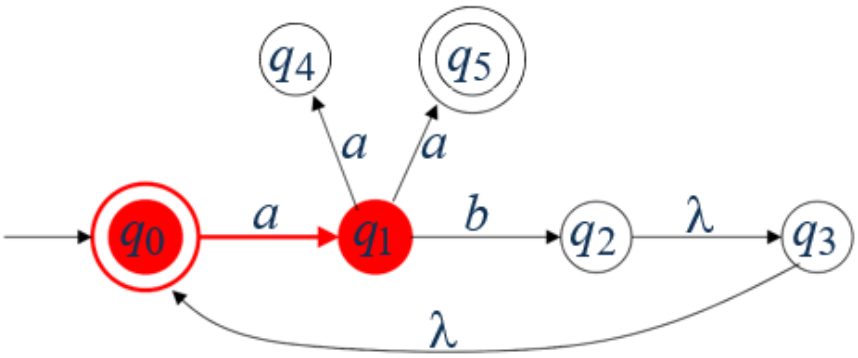
$$L(M_2) = \{a\}$$

6.

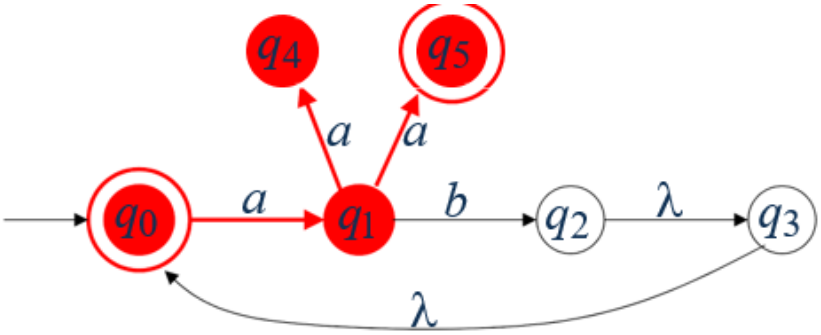
7.

Fungsi Transisi Lanjut δ*

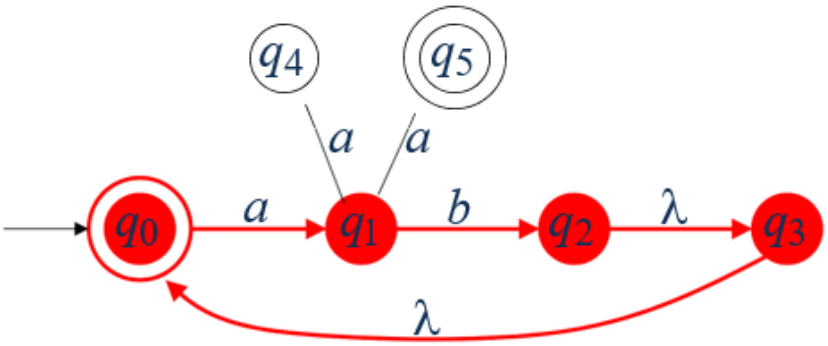
δ* (q_0 , a) = {q_1}



δ*(q_0 ,*aa*) = {q_4 , q_5}



δ * (q_0 , *ab*) = {q_2 , q_3, q_0}

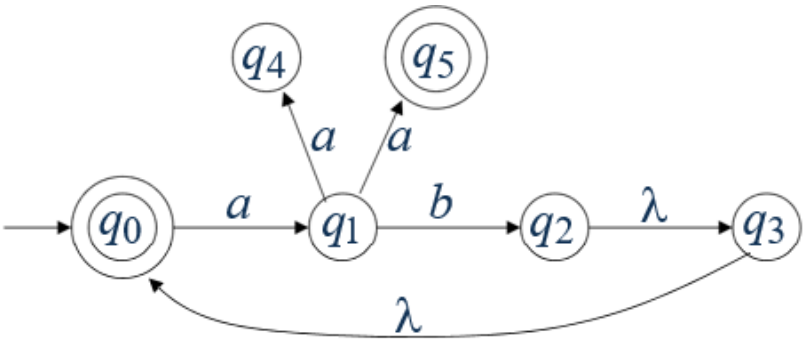


8. Secara formal

$q_j \in \delta^*(q_i, w)$: Perjalanan dari q_i ke q_j dengan label w

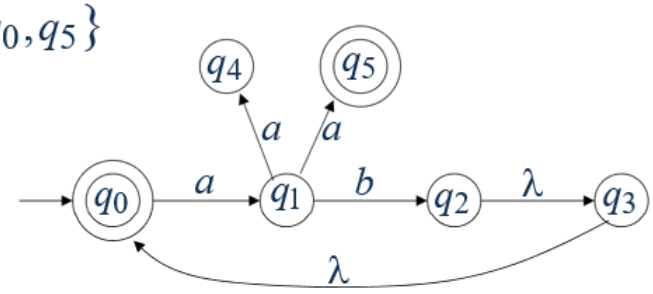
$w = \sigma_1 \sigma_2 \cdots \sigma_k$

L(M) ?



9. Bahasa dari NFA *M*

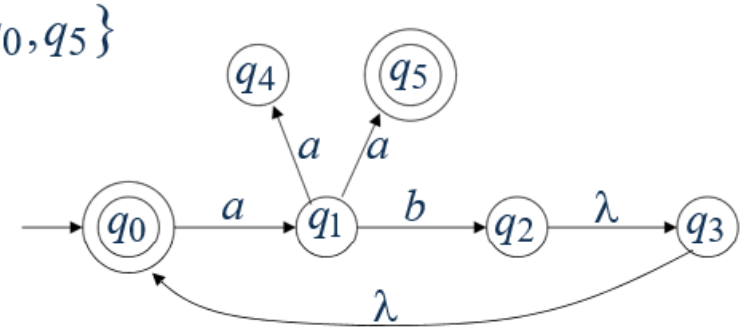
$F = \{q_0, q_5\}$



$\delta^*(q_0, aa) = \{q_4, q_5\}$ $\quad aa \in L(M)$

$\in F$

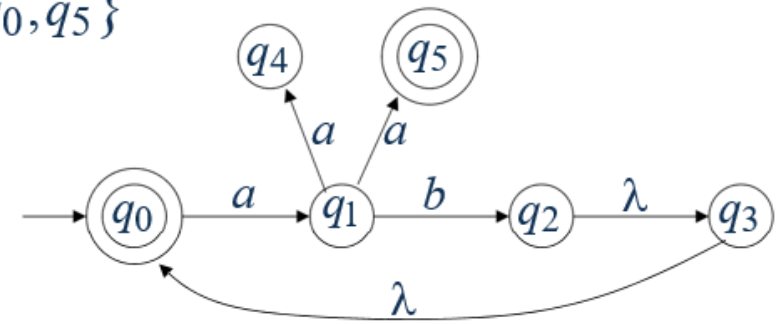
$F = \{q_0, q_5\}$



$\delta^*(q_0, ab) = \{q_2, q_3, \underline{q_0}\} \quad ab \in L(M)$

$\underline{q_0} \in F$

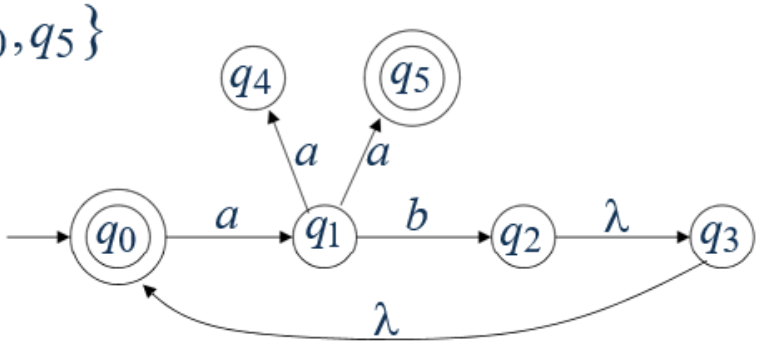
$F = \{q_0, q_5\}$



$\delta^*(q_0, abaa) = \{q_4, \underline{q_5}\} \quad aaba \in L(M)$

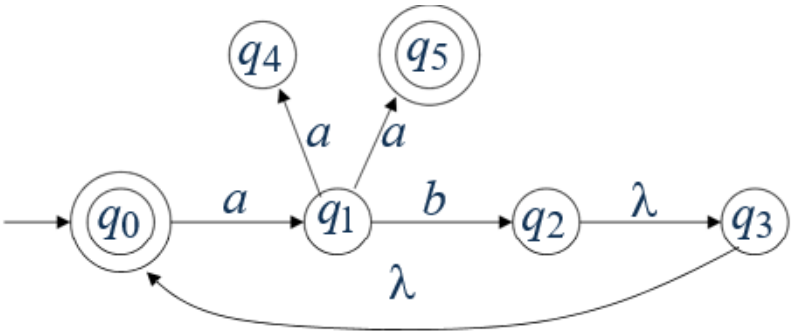
$\underline{q_5} \in F$

$F = \{q_0, q_5\}$



$\delta^*(q_0, aba) = \{q_1\} \quad aba \notin L(M)$

$q_1 \notin F$



$$L(M) = \{\lambda\} \cup \{ab\}^* \{aa\}$$

Secara formal

Bahasa yang diterima oleh NFA *M* adalah :

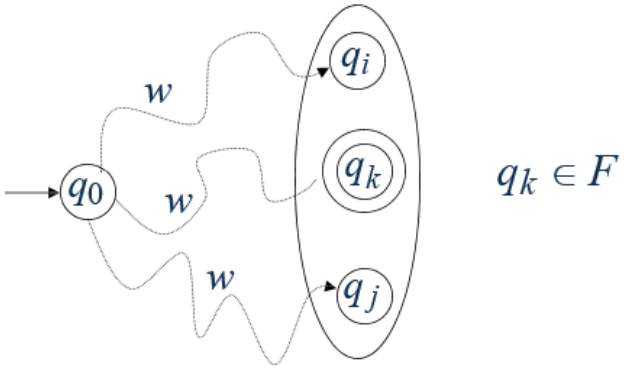
Dimana L (*M*) = {w_1, w_2 , w_3,...}

$$\delta^*(q_0, w_m) = \{q_i, q_j, \ldots, q_k, \ldots\}$$

dan

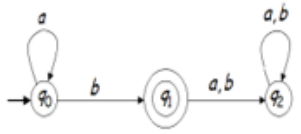
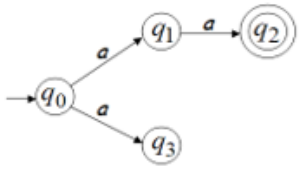
$q_k \in F$ (state yg diterima)

$w \in L(M)$ $\delta^*(q_0, w)$

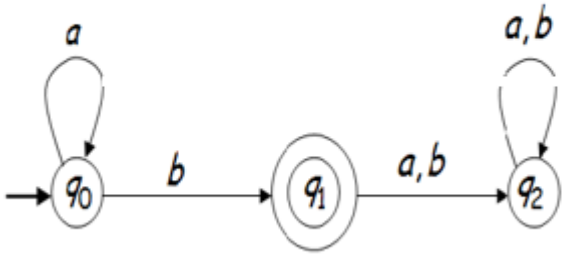


10. F A (Finite Automata)

Tabel 4.1 Penjelasan DFA dan NFA

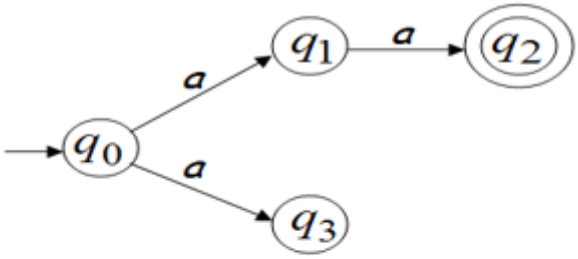
Jenis FA	Definisi	Bentuk Transisi
1. DFA *(Deterministic Finite Automata)*	FA di dalam menerima input mempuyai tepat satu busur keluar.	
2. NFA *(Non Deterministic Finite Automata)*	FA di dalam menerima input mempuyai lebih dari satu busur keluar atau tidak punya busur keluar.	

A. Graph Transisi DFA/NFA?



δ	a	b
q0	q0	q1
q1	q2	q2
q2	q2	q2

B. Graph Transisi DFA/NFA?



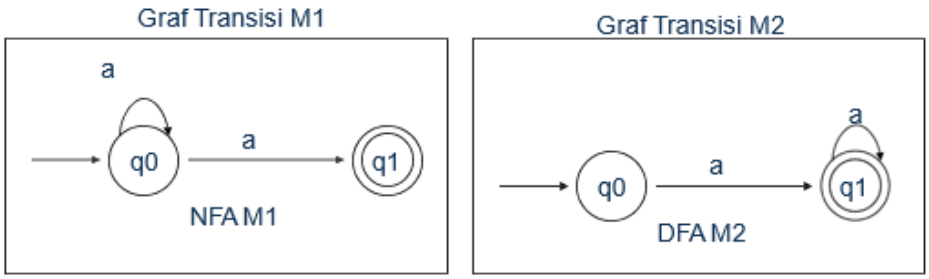
δ	a
q0	{q1,q3}
q1	q2
q2	-
q3	-

4.8 Ekuivalensi Antar FA

Diberikan dua mesin FA M1 dan M2. Masing-masing menerima bahasa L(M1) dan L(M2).

Kedua mesin tersebut disebut ekuivalen jika menerima bahasa yang sama yaitu :

L(M1) = L(M2)



δ	a
q0	{q0, q1}
q1	-

δ	a
q0	q1
q1	q1

L(M1) = $a^n a$

L(M2) = aa^n

4.9 Definisi Formal NFA

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

Q : Himpunan states/kedudukan

Σ : Himpunan symbol input/masukan/ alphabet

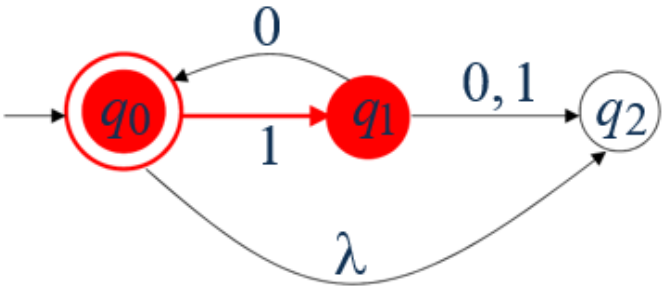
δ : Fungsi transisi

q_0 : state awal

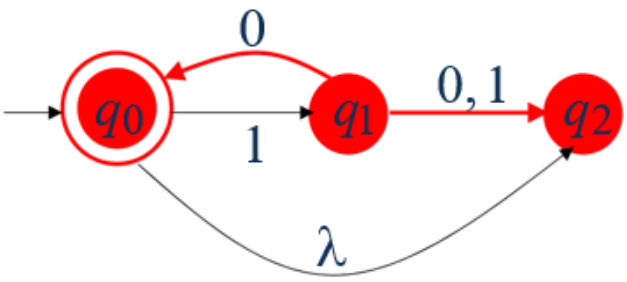
F : state akhir

A. Fungsi transisi δ

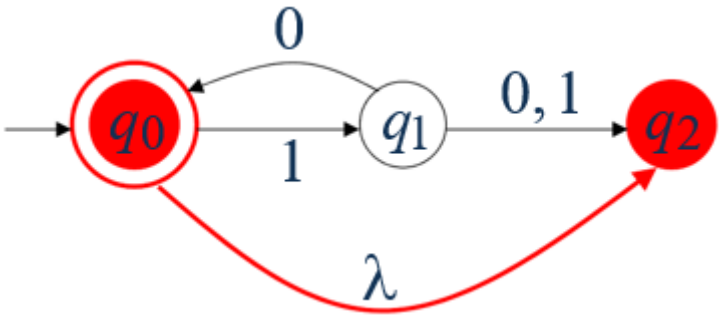
$\delta(q_0, 1) = \{q_1\}$



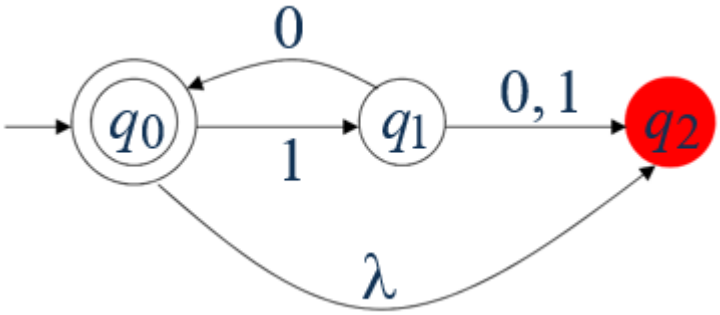
$\delta(q_1,0) = \{q_0, q_2\}$



$\delta(q_0, \lambda) = \{q_0, q_2\}$

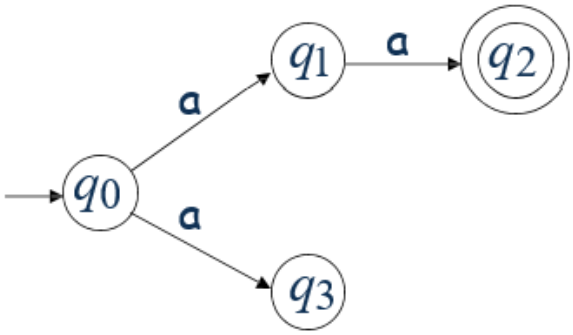


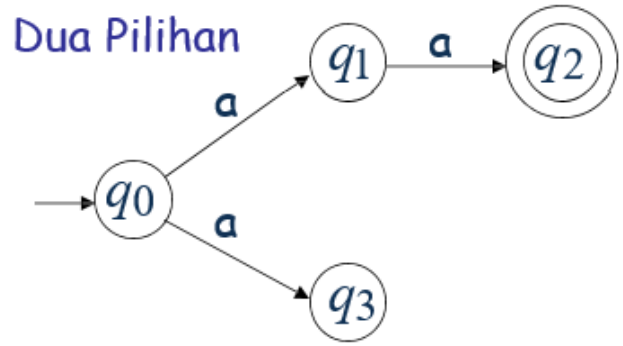
$\delta(q_2, 1) = \emptyset$

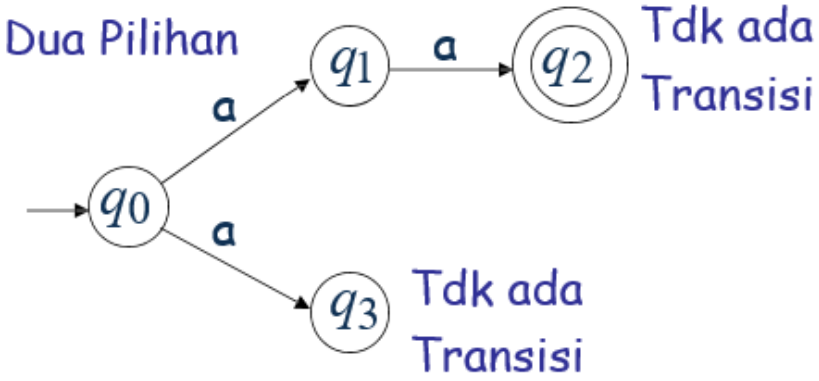


4.10 Nondeterministic Finite Automaton (NFA)

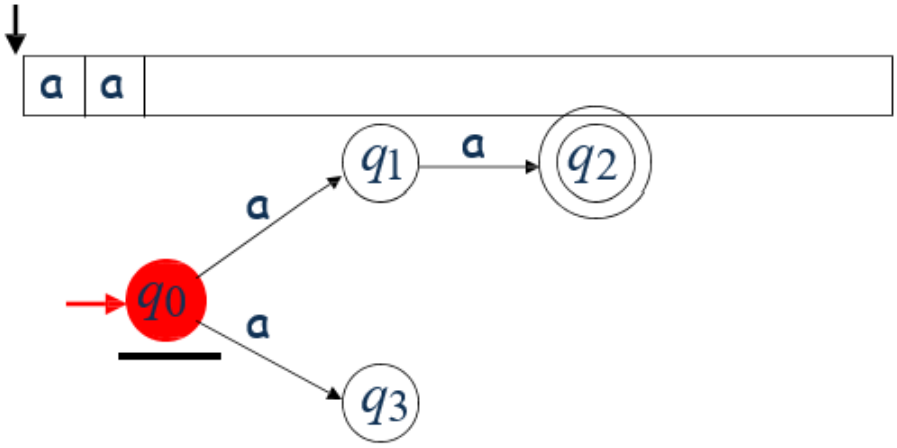
Alphabet = {a}

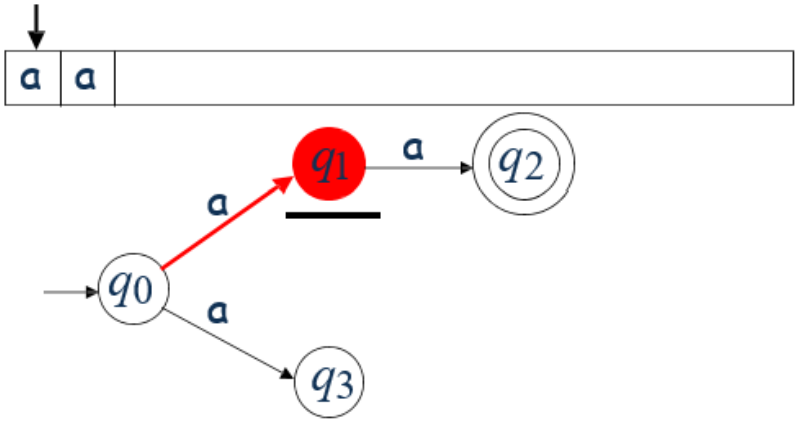


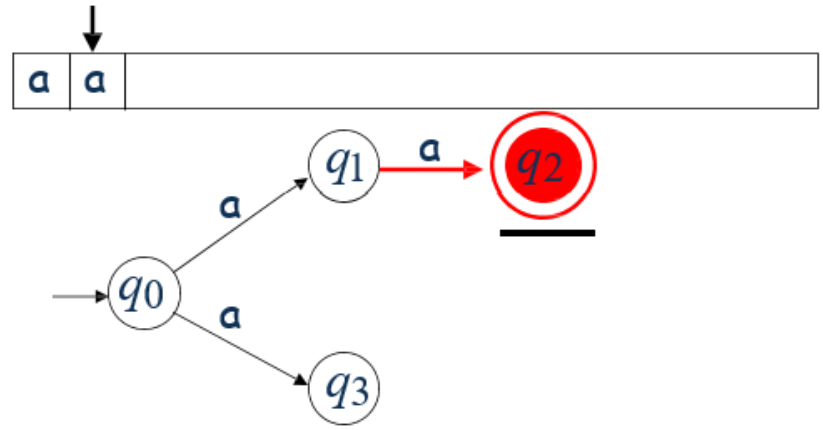


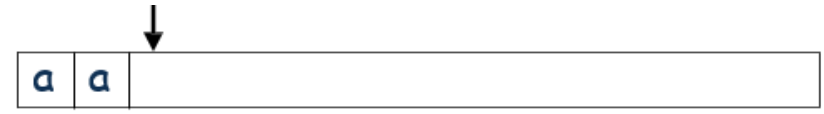


Pilihan Pertama :

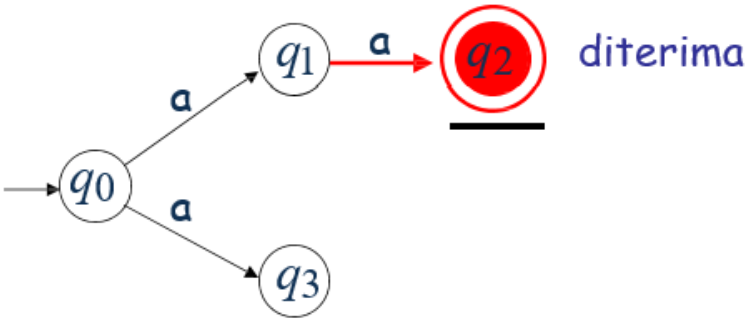




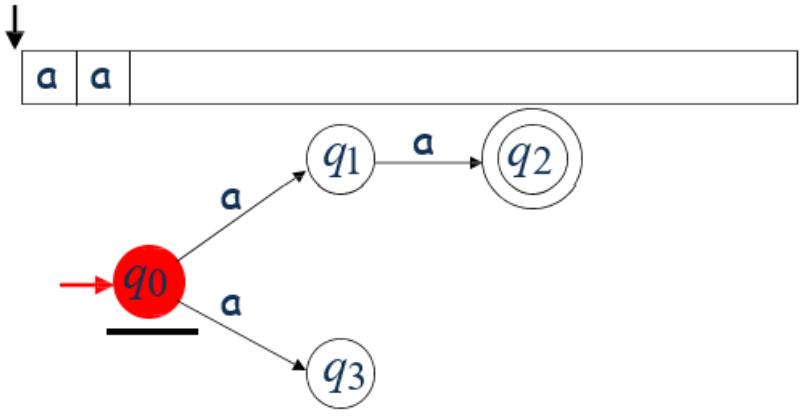


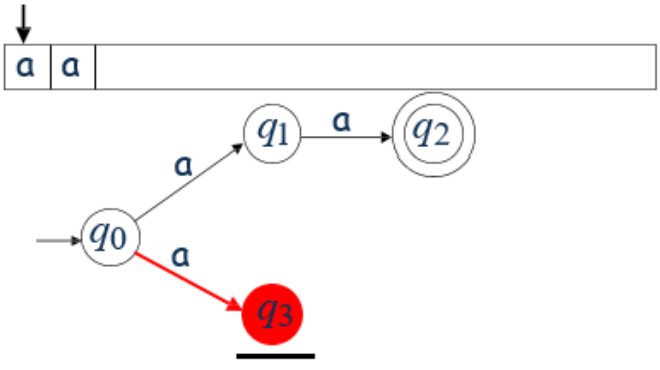


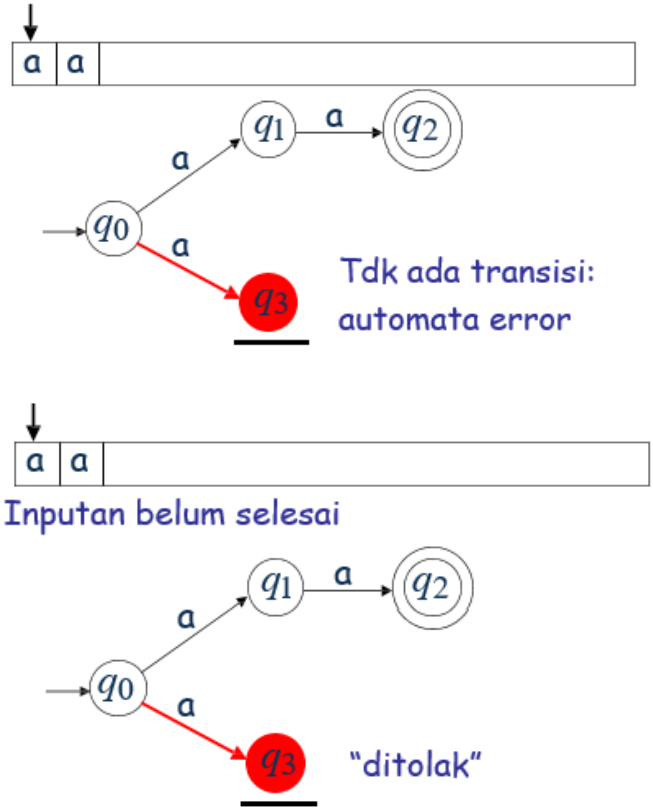
Inputan sudah selesai



Pilihan Kedua :







4.11 **Sebuah NFA menerima string, jika:**

Komputasi pada NFA menerima string

Komputasi adalah :

Seluruh inputan dimasukkan dan automata dimulai dari state awal menuju ke state akhir

Contoh 1 :

Apakah aa diterima oleh NFA?

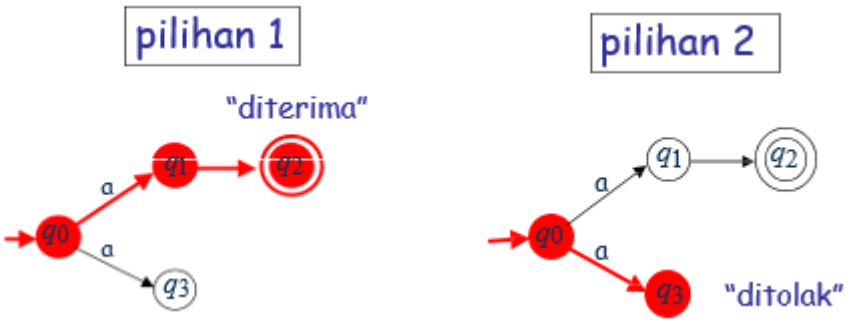
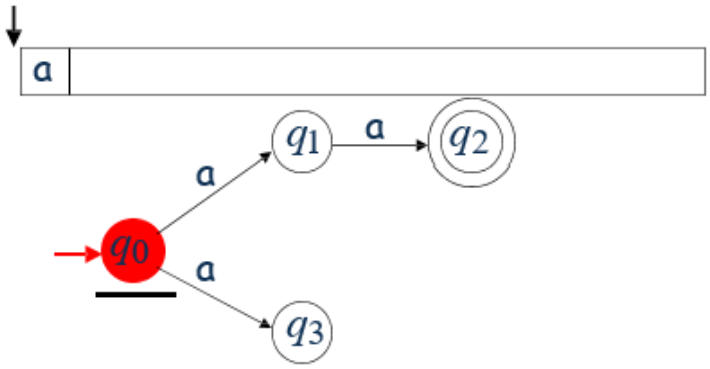
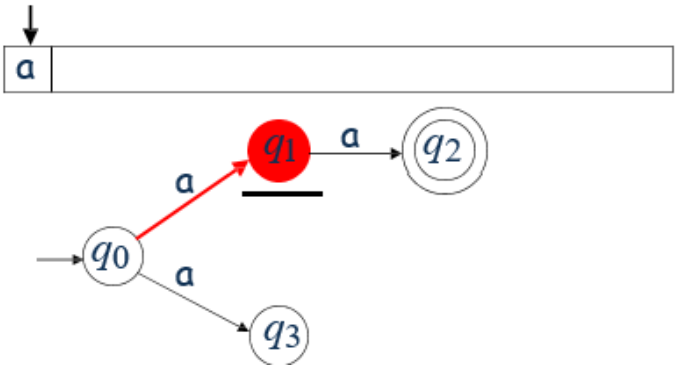


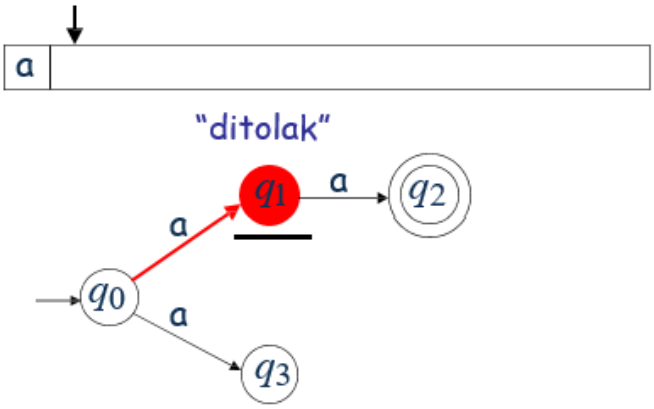
Diagram transisi yang dipilih adalah pilihan 1. Karena komputasi dapat diterima *aa*.

Diterima / Ditolak

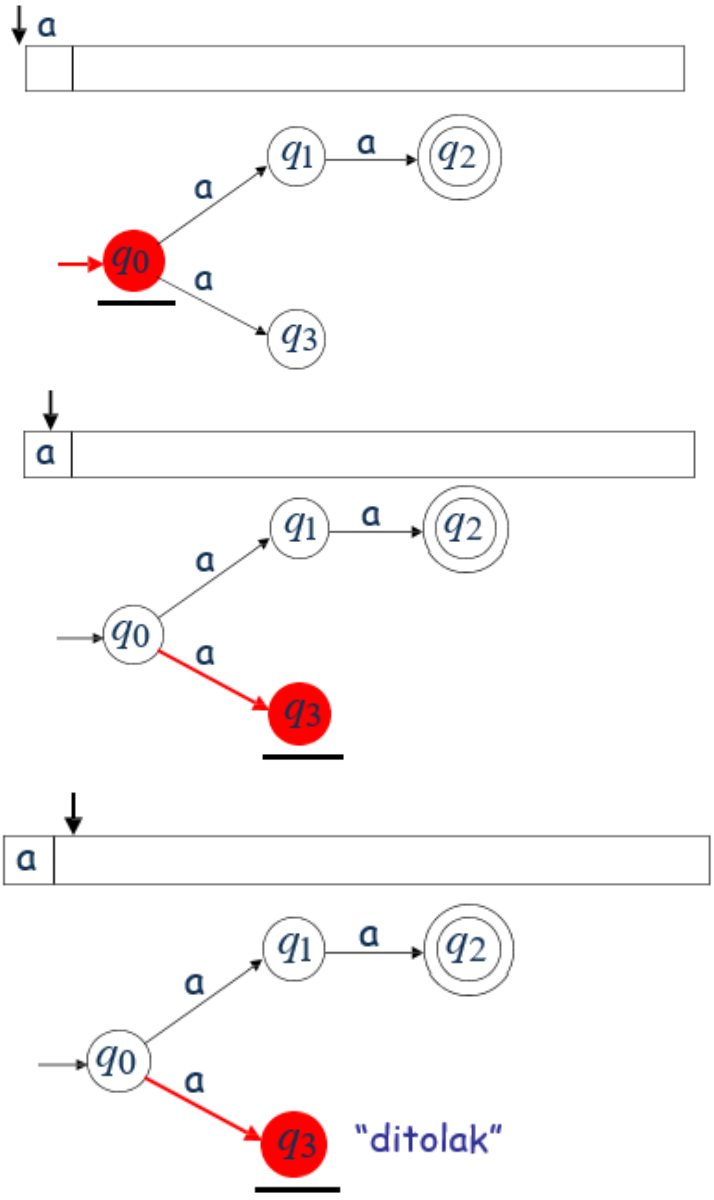


Pilihan Pertama





Pilihan Kedua



Sebuah NFA Menolak string, jika:

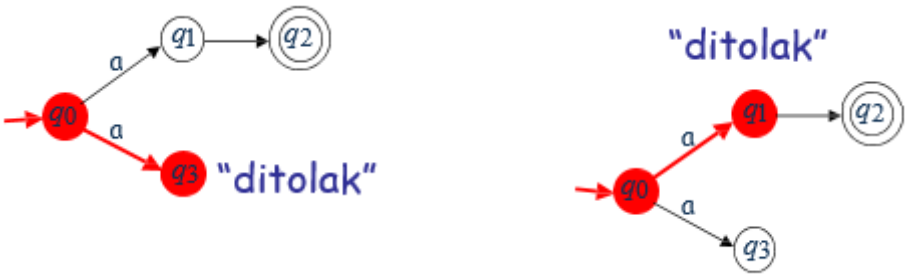
Tidak ada komputasi pada NFA yang menerima string.

Untuk setiap komputasi:

seluruh inputan selesai dimasukkan dan automata tidak sampai pada State akhir atau, inputan belum selesai dimasukkan.

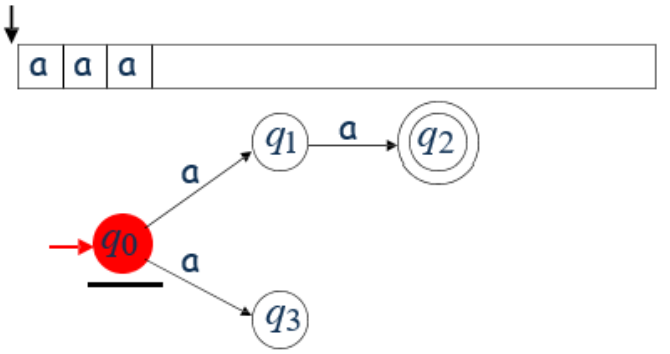
Contoh 2 :

Apakah *a* ditolak oleh NFA ?

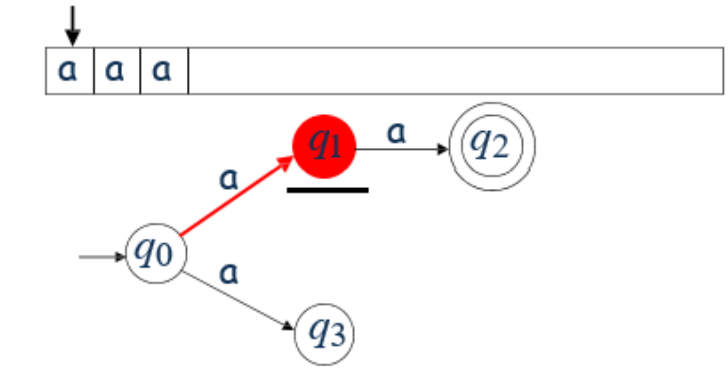


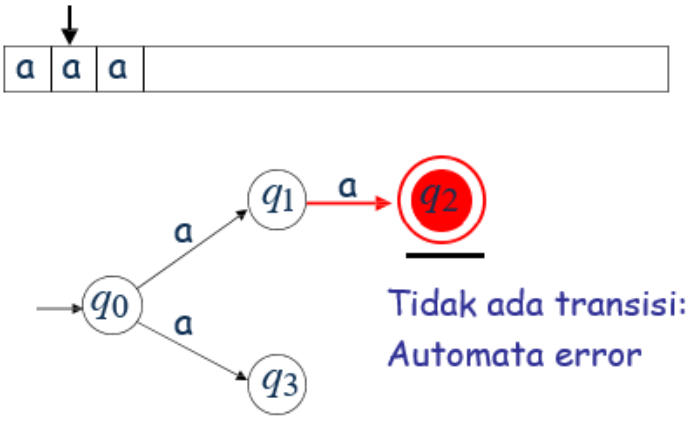
Seluruh komputasi yang mungkin "ditolak"

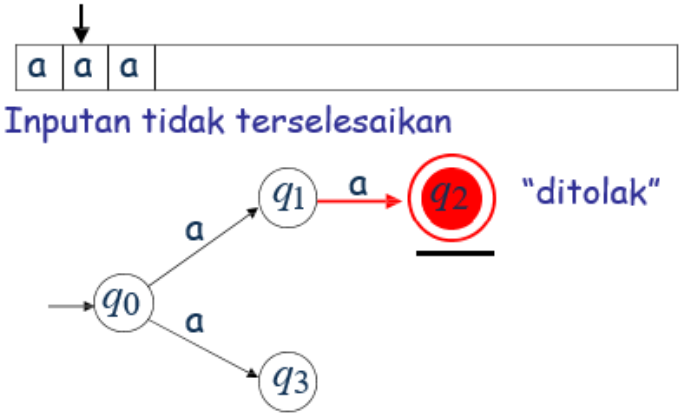
Diterima / ditolak?



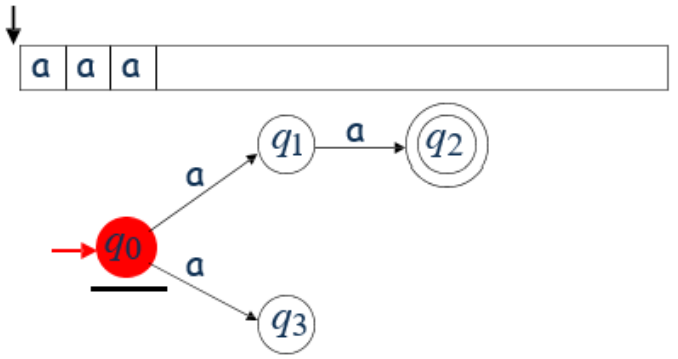
Pilihan Pertama

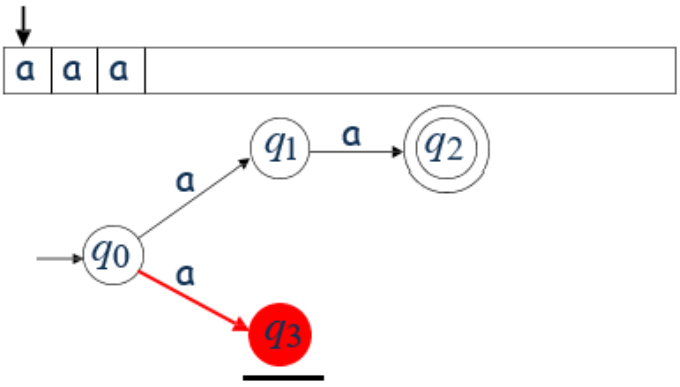


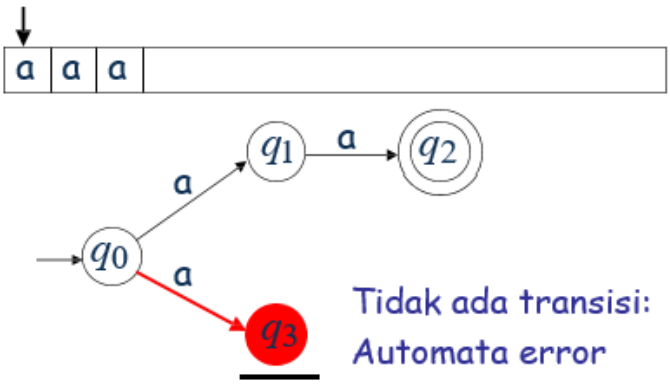


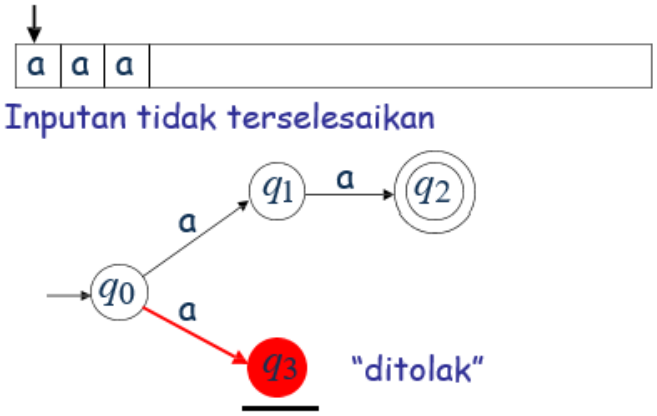


Pilihan Kedua

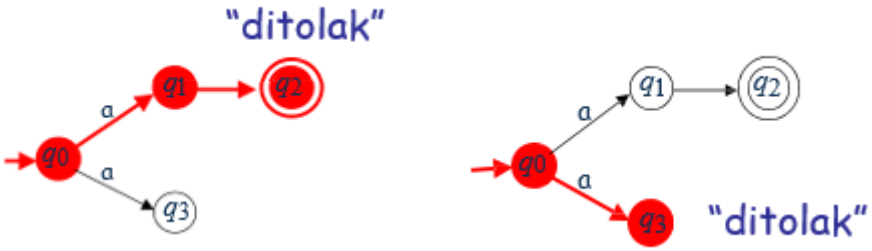






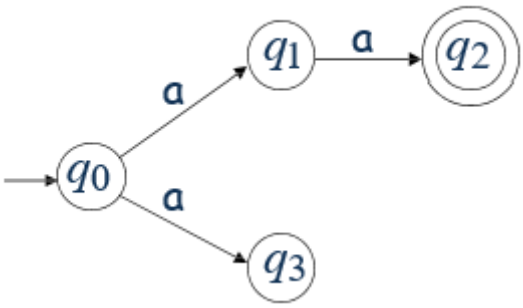


aaa ditolak oleh NFA

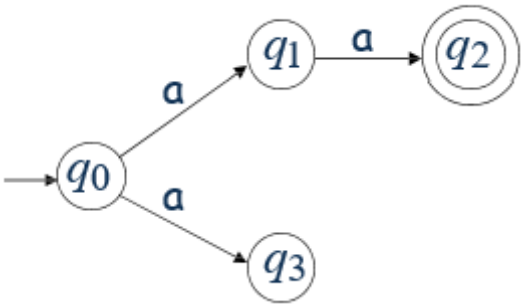


Seluruh komputasi yang mungkin "ditolak"

L(M) ?



Bahasa yng meneruma L = {*aa*}



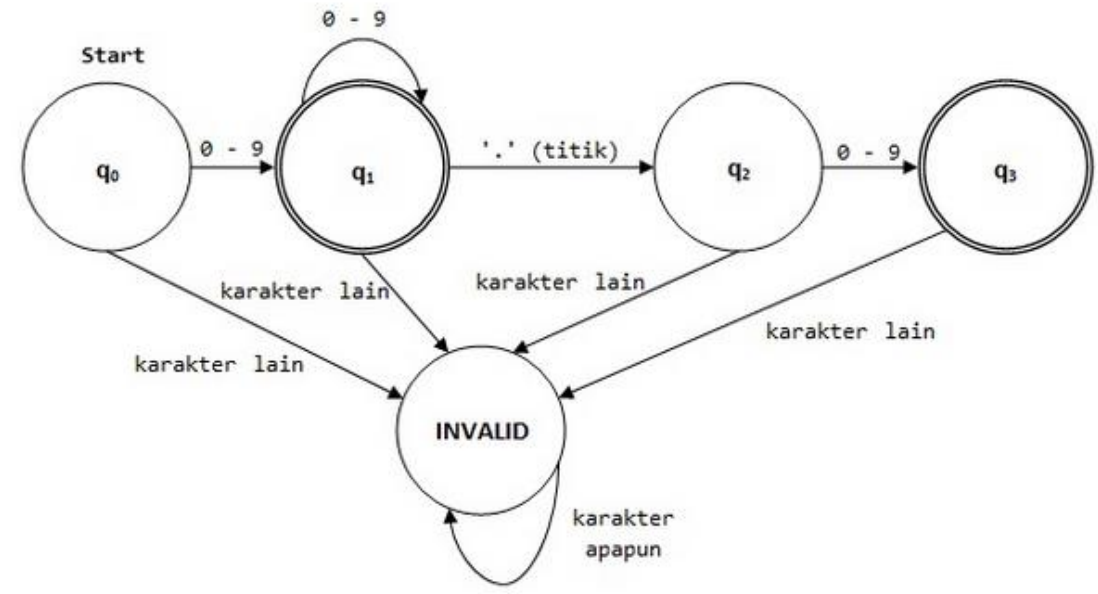
4.12 Hubungan DFA dan NFA

1) Semua DFA merupakan bagian dari NFA.
2) DFA dapat menuntun recognizer lebih cepat dibanding NFA.
3) Lebih mudah membangun NFA dibanding DFA untuk suatu bahasa
4) Namun lebih mudah mengimplementasikan DFA dibanding NFA

4.13 Implementasi FSA ke dalam Program Komputer C++

FSA untuk mengenali bilangan bulat

1. Diagram Transisi



Gambar 4.1. Diagram Transisi

2. Memodelkan state ke dalam kode program

```
#include <iostream>
#include <string.h>

using namespace std;

typedef enum {
    q0,
    q1,
    q2,
    q3,
    INVALID = -1
}
STATE;
```

Gambar 4.2. Implementasi state dalam kode program

3. Fungsi transisi FSA

   Implementasi Tabel : transisi dengan fungsi yang mengembalikan nilai state

```
STATE delta(STATE currentState, char readHead) {
    if (isdigit(readHead)) {
        switch (currentState) {
            case q0:
                return q1;
            case q1:
                return q1;
            case q2:
                return q3;
            case q3:
                return q3;
        }
    }
    else if (readHead == '.') {
        switch (currentState) {
            case q1:
                return q2;
            default:
                INVALID;
        }
    }
    else
        return INVALID;
}
```

4. Fungsi untuk mengecek input, diterima atau tidak

```
bool isStringAccepted(char input[50]) {
    int i = 0;
    STATE state = q0; //mengeset state mesin ke state awal.
    while (i < strlen(input) && state != INVALID) {
        state = delta(state, input[i++]);
    }
    if (state == q1 || state == q3)
        return true;
    else
        return false;
}
```

```
int main() {
    char input[50];
    cin >> input;
    if (isStringAccepted(input))
        cout << "Accepted!" << endl;
    else
        cout << "Rejected!" << endl;
    return 0;
}
```

BAB V
NFA DENGAN EPSILON DAN KONVERSINYA KE NFA TANPA EPSILON

5.1 Tujuan Instruksional

C. Tujuan Instruksional Umum

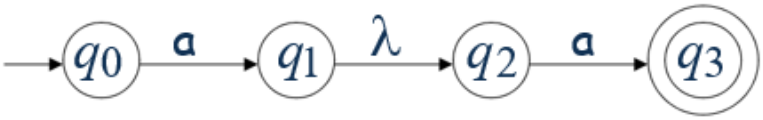
Mahasiswa memahami konsep NFA dengan epsilon dan konversinya

D. Tujuan Instruksional Khusus

Mahasiswa dapat membuat NFA dengan epsilon dan konversinya

5.2 Transisi Lambda/Epsilon (l/e)

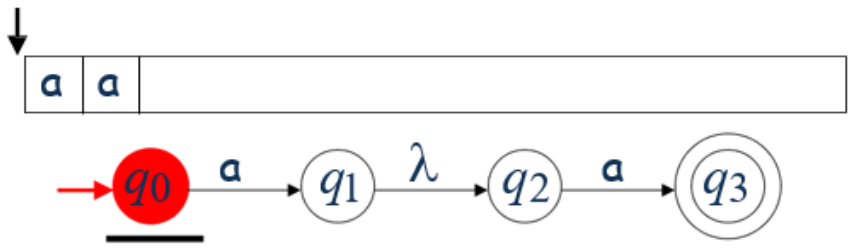
Adalah Transisi yang diperbolehkan melakukan perubahan state tanpa mendapatkan inputan.

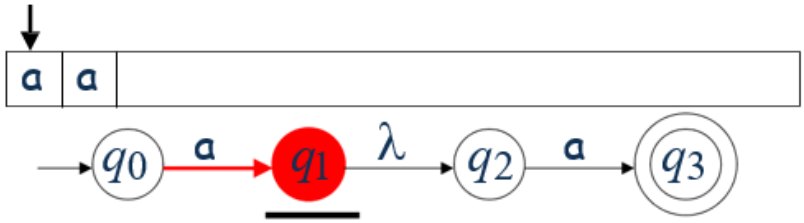


Penjelasan :

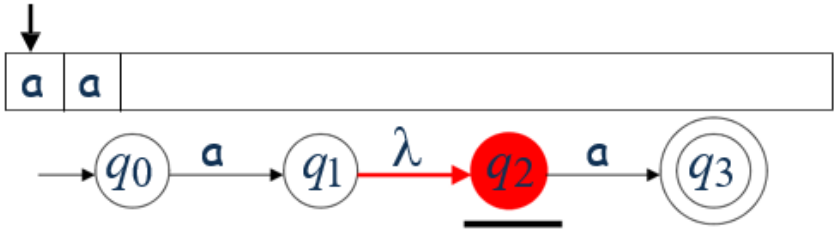
1. ε -move atau λ (ε di sini bisa dianggap sebagai 'empty')
2. Pada mesin NFA kita bisa mengubah state state tanpa harus bergantung pada suatu inputan, cukup dengan menggunak ε move, kita bisa menuju ke state state yg ada,
3. ε move ini juga disebut dengan transisi
4. Tidak bergantung pada suatu input ketika melakukan transisi atau perpindahan.
5. Agar kita bisa mudah dalam mengkombinasikan finite state automata.

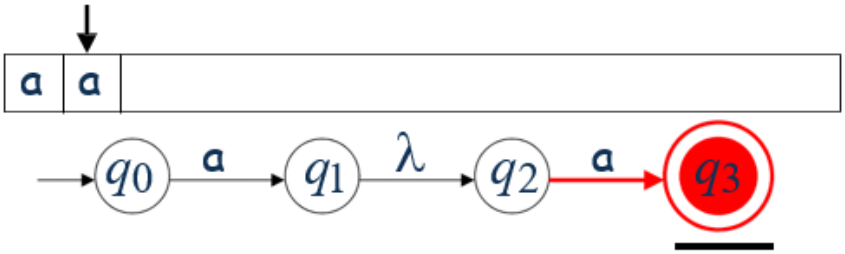
Contoh 1



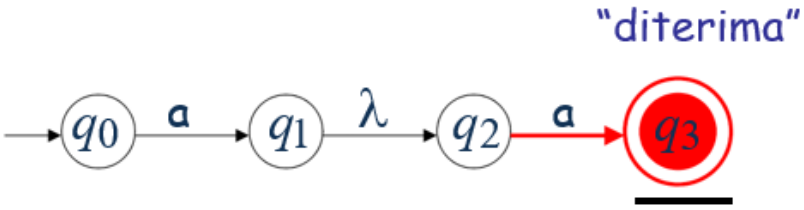


Ada pergerakan tapi tidak ada inputan



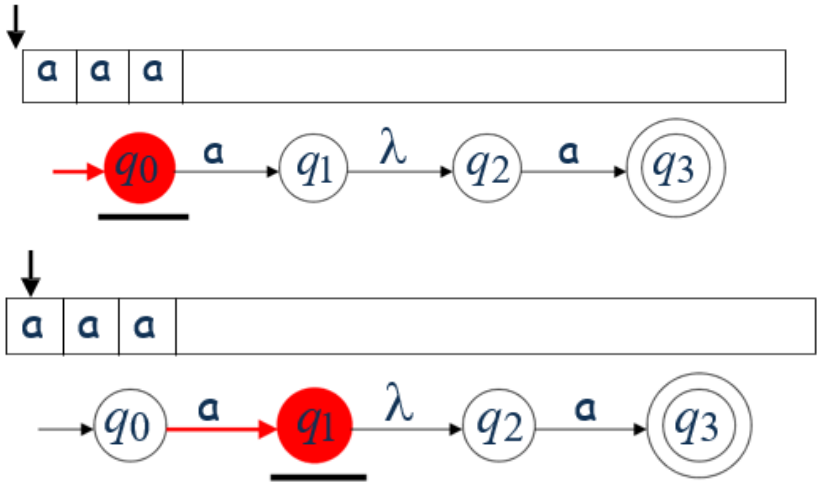


Inputan terselesaikan

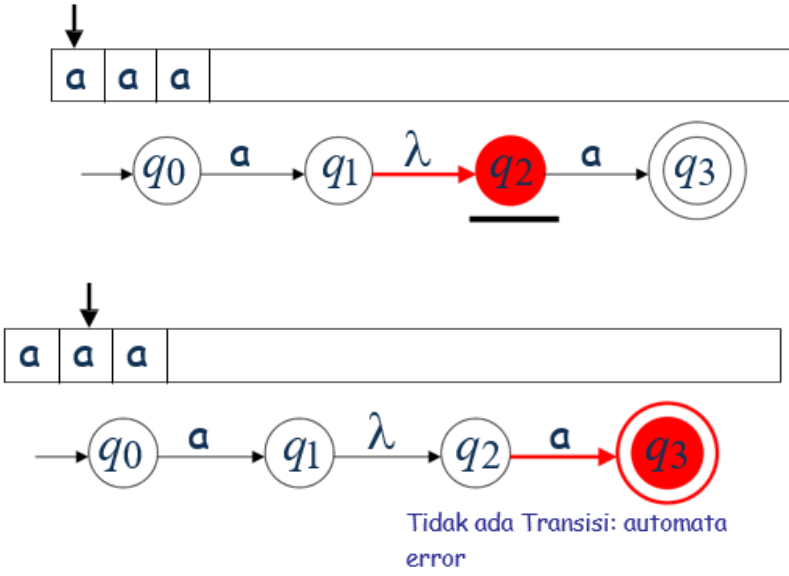


String *aa* diterima

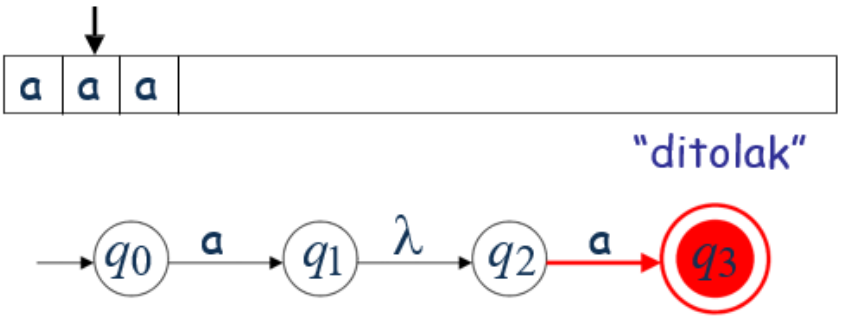
Contoh 2 :



Ada pergerakan tetapi tidak ada inputan

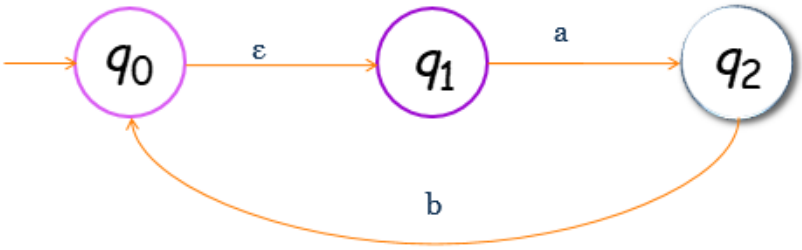


Inputan tidak terselesaikan

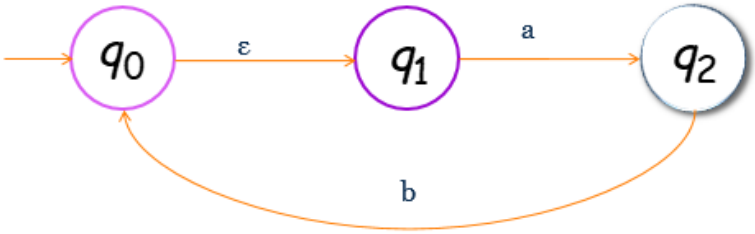


Pada NFA jenis ini diperbolehkan suatu status berubah secara sepontan tanpa membaca input

ε (epsilon) ----» string kosong



ε-closure adalah himpunan state yang dapat dicapai dari suatu state tanpa adanya input.

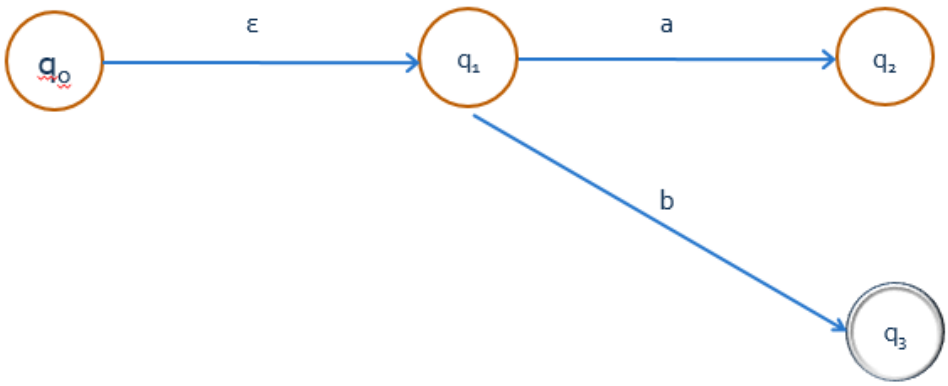


- Klosure-ε (q_o) = {q_o ,q_1 }
- Klosure-ε (q_1) = {q_1}
- Klosure-ε (q_2) = {q_2}

5.3 Ekuivalensi NFA dengan ε-move ke NFA tanpa ε-move

1. Buat tabel transisi NFA dengan ε-move
2. Tentukan ε-closure setiap state
3. Carilah fungsi transisi /tabel transisi yang baru, rumus :
4. δ'(state,input)=ε-closure(δ(ε-closure(state,input))
5. Tentukan state akhir ditambah dengan state yang ε-closure nya menuju state akhir, rumusnya :
6. F' = F ∪ {q | (ε-closure(q) ∩ F ≠ ∅}

Contohnya :



Tabel Transisi

δ	a	b	E
q_o	Ø	Ø	q_1
q_1	q_2	q_3	q_1
q_2	Ø	Ø	q_2
q_3	Ø	Ø	q_3

ε-closure setiap state

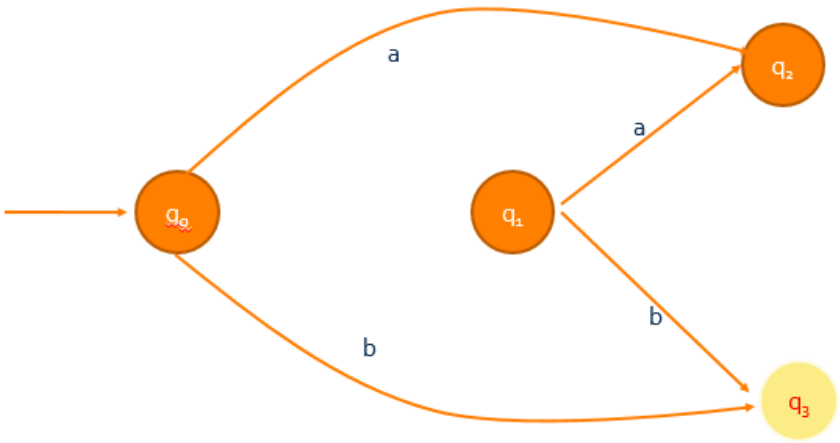
1. Klosure-ε (q_o) = {q_o ,q_1}
2. Klosure-ε (q_1) = {q_1}
3. Klosure-ε (q_2) = {q_2}
4. Klosure-ε (q_3) = {q_3}

Tabel Transisi yang baru (δ')

δ	a	B
q_0	ε-cl(δ(ε-cl(q0),a)) ε-cl(δ({q0,q1},a)) ε-cl(q2) {q2}	ε-cl(δ(ε-cl(q0),b)) ε-cl(δ({q0,q1},b)) ε-cl(q3) {q3}

q_1	ε-cl(δ(ε-cl(q1),a)) ε-cl(δ({q1},a)) ε-cl(q2) {q2}	ε-cl(δ(ε-cl(q1),b)) ε-cl(δ({q1},b)) ε-cl(q3) {q3}
q_2	ε-cl(δ(ε-cl(q2),a)) ε-cl(δ({q3},a)) ε-cl(∅) ∅	ε-cl(δ(ε-cl(q2),b)) ε-cl(δ({q2},b)) ε-cl(∅) ∅
q_3	ε-cl(δ(ε-cl(q3),a)) ε-cl(δ({q3},a)) ε-cl(∅) ∅	ε-cl(δ(ε-cl(q3),b)) ε-cl(δ({q3},b)) ε-cl(∅) ∅

Hasil ekuivalensi



BAB VI
EKSPRESI REGULER/REGULER EXPRESSION (REGEX)

6.1 Tujuan Instruksional

A. Tujuan Instruksional Umum

Mahasiswa memahami konsep ekspresi reguler

B. Tujuan Instruksional Khusus

Mahasiswa dapat menerapkan ekspresi reguler

6.2 Tujuan Ekspresi Reguler

1. Memahami konsep ekspresi reguler dan hubungannya dengan automata
2. Mengetahui penerapan ekspresi reguler
3. Mengetahui bahasa untuk ekspresi reguler
4. Mengetahui proses konversi ekspresi reguler ke automata dan sebaliknya

6.3 FSA dan Regex

1. Ekspresi Reguler bagian dari bahasa reguler
2. Bahasa dinyatakan regular jika terdapat FSA yang dapat menerimanya.
3. Bahasa yang diterima FSA bisa dinyatakan dengan ekspresi reguler.

6.4 Apa itu Ekspresi Reguler?

Suatu cara untuk menspesifikasikan bahasa. Memberikan suatu pola/pattern/template untuk untai/string dari suatu bahasa.

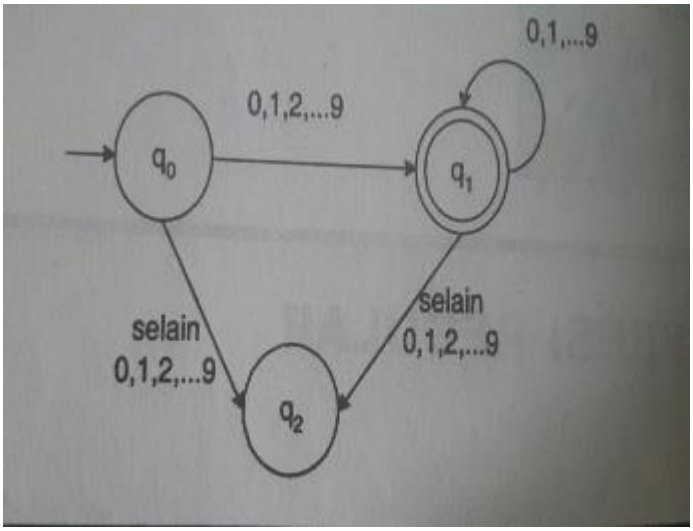
6.5 Implementasi Regex

a) Pencarian (*searching*) untai karakter (string) pada suatu file, biasanya terdapat pada text editor

b) Pembatasan data masukan yang diperkenankan, misal suatu field masukan hanya menerima input bilangan (0..9)

6.6 Contoh Automata dan Regex

7.



Gambar 32. Contoh automata dan regex

Automata hanya menerima simbol input antara 0 – 9

Regulernya : (digit) (digit)*

Digit = 0 – 9

FSA akan diterjemahkan menjadi kode dalam bahasa pemrograman

6.7 Notasi Regex

a) '*'(karakter asterisk, bisa tidak muncul, bisa juga muncul berhingga kali (0-n)

b) '+' (pada posisi superscript, minimal muncul 1 kali (1-n)

c) '+' / 'U' (union)

d) '.' (konkatenasi)

6.8 Contoh Regex

1) ab*cc
2) 010*
3) a*d
4) a+d
5) a* U b*
6) (aUb)

7) (aUb)*

6.9 Contoh Ekspresi Reguler dan String yang dibangkitkan

a) ab*cc

abcc

abbcc

abbbcc

abbbbcc

acc

b) 010*

01

010

0100

01000

c) a*d

d

ad

aad

aaad

d) a^{+}d

ad

aad

aaad

e) a*Ub*

'U' berarti atau

a

b

aa

bb

aaa

bbb

aaaa

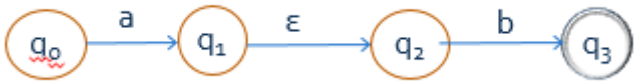
bbbb

6.10 Hubungan Regex dan FSA

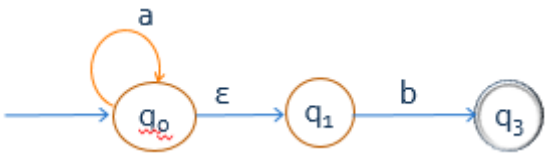
A. Untuk tiap Regex ada satu NFA e-move yang ekuivalen

B. Untuk tiap DFA ada satu regex dari bahasa yang diterima oleh DFA tersebut

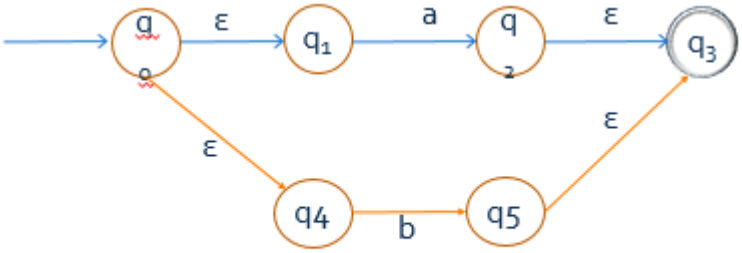
1. Contoh NFA e-move untuk ER : ab



2. Contoh NFA e-move untuk ER : a*b



3. Contoh NFA e-move untuk ER : a U b



NFA e-move → NFA → DFA

C. Deskripsi

1. Dari sebuah FSA (NFA atau DFA) kita bisa menentukan ekspresi reguler yang diterima oleh otomata bersangkutan dan sebaliknya
2. Terdapat langkah-langkah formal untuk menemukan ekspresi reguler dari FSA.
3. Tapi kita juga bisa langsung menentukan ekspresi regulernya dengan mengamati perilaku dari otomata tersebut

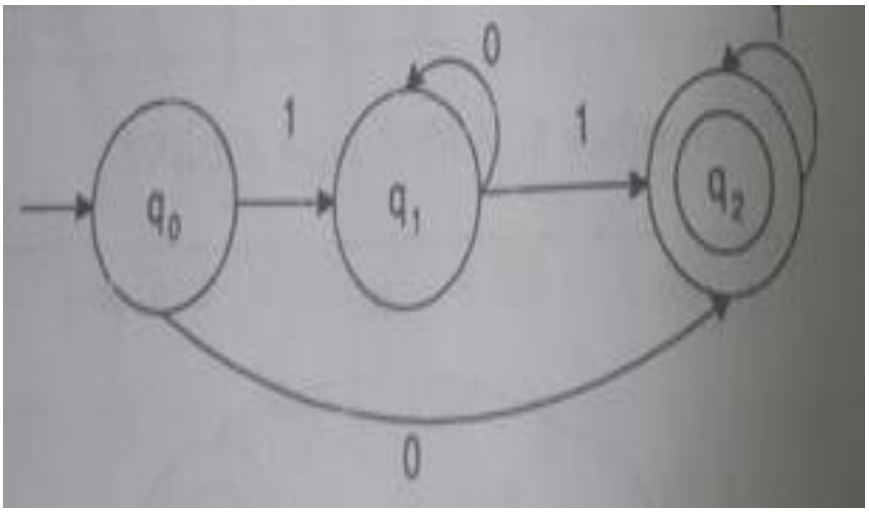
D. Contoh Lain Regex

$$r = (aa)^*(bb)^*b$$

Dapat ditulis menjadi :

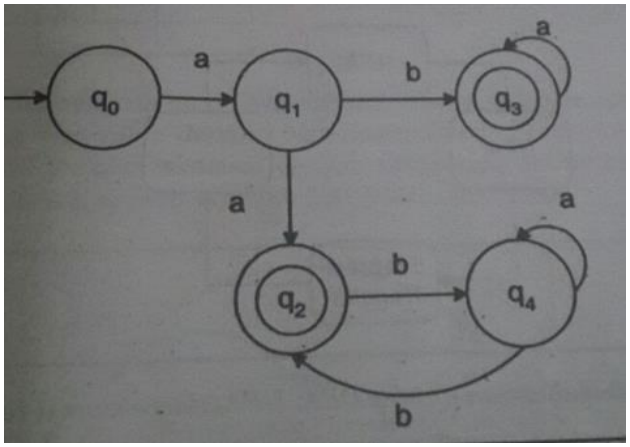
$$L(r) = \{a^{2n} b^{2m} b : n,m \geq 0\}$$

E. Menentukan Regex dari suatu Automata



Gambar 33. Menentukan regex dari suatu automata

1. Mesin menerima 01* atau
2. Mesin menerima 10*11*
3. Dalam Regex
4. 01* U 10*11*



Gambar 34. Menentukan regex dari suatu automata

1. Final State q2 dan q3
2. Input menuju q3 yaitu ab
3. Q3 menerima a*

4. Sehingga q3 menerima aba*
5. Input menuju q2 yaitu aa
6. Q2 menerima ba*b
7. Sehingga q2 menerima aa(ba*b)*
8. Regex = aba* U aa(ba*b)*

BAB VII
CONTEXT FREE GRAMMAR DAN PARSING TREE

7.1 Tujuan Instruksional

A. Tujuan Instruksional Umum

Mahasiswa memahami konsep CFG dan Parsing Tree

B. Tujuan Instruksional Khusus

Mahasiswa dapat membuat CFG dan Parsing Tree

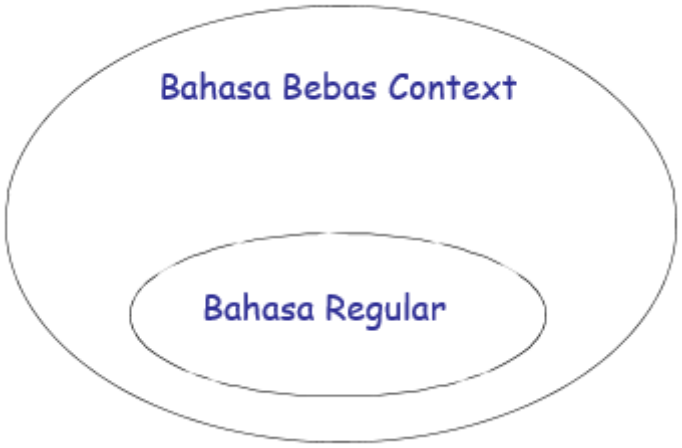
7.2 Bahasa Formal

1. Generator → Grammar → Aturan untuk menghasilkan string dari suatu bahasa
2. Acceptor/Recognizer → Automata → Mesin yang dapat mengenali string input

7.3 Hirarki Chomsky

Penggolongan bahasa menjadi 4 tingkatan :

Bahasa	Mesin Automata
Reguler	FSA (DFA dan NFA)
Bebas Konteks/Context Free	PDA
Context Sensitive	LBA
Unrestricted/Phase Structure/Natural Language	Mesin Turing



7.4 Bahasa Reguler/Reguler Grammar

α → β dimana α dan β bisa berupa terminal atau non terminal/variabel

α adalah simbol variabel

β maksimal memiliki sebuah simbol variabel yang terletak di paling kanan bila ada

Terdapat batasan pada ruas kanan atau hasil produksi

A → e

A → efg

A → efgH

C → D

7.5 Bahasa Bebas Konteks/Context Free Grammar

a → b

Tidak terdapat batasan hasil produksi

Batasannya hanya ruas kiri (a) adalah sebuah simbol variabel

Simbol variabel/non terminal → simbol yang masih bisa diturunkan (huruf besar)

Simbol terminal → simbol yang sudah tidak bisa diturunkan (huruf kecil)

S → Aa | B

A → ab

B → b

7.6 Notasi Grammer

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

V : Himpunan variable/non terminal

T : Himpunan Simbol Terminal

S : Simbol awal

P : Himpunan aturan produksi

Contoh:

Grammer : S → aSb

S→ λ

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

$V = \{S\}$ $T = \{a, b\}$

$P = \{S \rightarrow aSb,\ S \rightarrow \lambda\}$

Contoh :

8.

$\langle sentence \rangle \rightarrow \langle noun\_phrase \rangle\ \langle predicate \rangle$

$\langle noun\_phrase \rangle \rightarrow \langle article \rangle\ \langle noun \rangle$

$\langle predicate \rangle \rightarrow \langle verb \rangle$

$\langle article \rangle \rightarrow a$

$\langle article \rangle \rightarrow the$

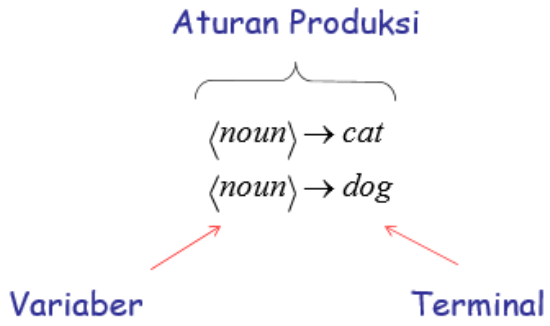
$\langle noun \rangle \rightarrow cat$

$\langle noun \rangle \rightarrow dog$

$\langle verb \rangle \rightarrow runs$

$\langle verb \rangle \rightarrow walks$

9.



Keterangan :

<sentence>, <noun _ phrase> , <predicate>, <article>, <noun>, <verb> adalah

Non Terminal karena masih bisa diturunkan

Sedangkan

{a, the, cat, dog, runs, walks} adalah

Terminal, karena sudah tidak bisa diturunkan

$$\boldsymbol{G = (V, T, S, P)}$$

V : {<sentence>, <noun _ phrase> , <predicate>, <article>, <noun>, <verb> }

T : {a, the, cat, dog, runs, walks}

S : <sentence>

P : { di slide sebelah sebelumnya}

Derivasi dari "*the dog walks"*:

<sentence> → *<noun_phrase> <predicate>*

→ *<noun_phrase> <verb>*

→ *<article> <noun> <verb>*

→ *the <noun> <verb>*

→ *the dog <b verb>*

→ *the dog walks*

Derivasi dari "*a cat runs*"

<sentence> → *<noun_phrase> <predicate>*

→ *<noun_phrase> <verb>*

→ *<article> <noun> <verb>*

→ *a <noun> <verb>*

→ *a cat < verb>*

→ *a cat runs*

Bahasa dari gramer:

L = { "a cat runs",

"a cat walks",

"the cat runs",

"the cat walks",

"a dog runs",

"a dog walks",

"the dog runs",

"the dog walks" }

7.7 PARSING TREE

Menggambarkan bagaimana memperoleh suatu string (untai) dengan cara menurunkan simbol variabel menjadi simbol terminal.

A. PARSING

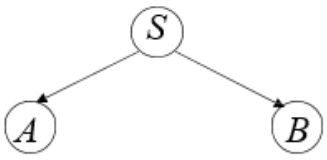
Proses pengenalan sebuah string masukan yang terdiri dari symbol-symbol apakah string tersebut sesuai dengan tata bahasa yang berlaku

B. Pohon (tree)

Graph terhubung tidak sirkuler yang memiliki satu simpul (node/vertex) yang disebut akar (root) dan memiliki lintasan ke tiap simpul.

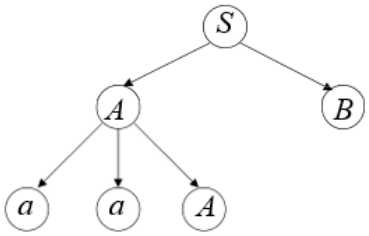
$$S \rightarrow AB \qquad A \rightarrow aaA \mid \lambda \qquad B \rightarrow Bb \mid \lambda$$

$$S \Rightarrow AB$$



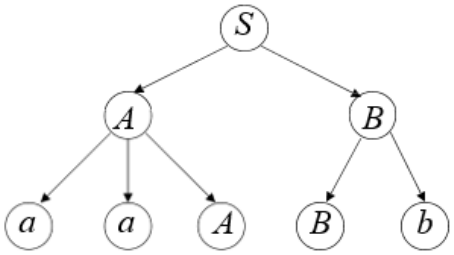
$S \to AB \qquad A \to aaA \,|\, \lambda \qquad B \to Bb \,|\, \lambda$

$S \Rightarrow AB \Rightarrow aaAB$



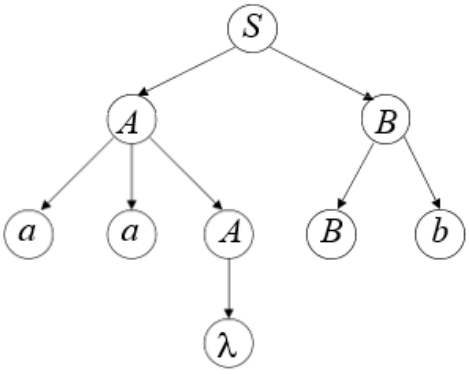
$S \to AB \qquad A \to aaA \,|\, \lambda \qquad B \to Bb \,|\, \lambda$

$S \Rightarrow AB \Rightarrow aaAB \Rightarrow aaABb$



$S \to AB \qquad A \to aaA \,|\, \lambda \qquad B \to Bb \,|\, \lambda$

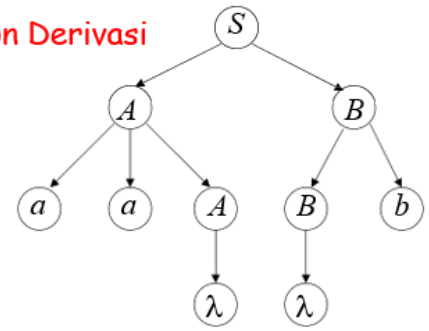
$S \Rightarrow AB \Rightarrow aaAB \Rightarrow aaABb \Rightarrow aaBb$



$S \to AB \qquad A \to aaA \mid \lambda \qquad B \to Bb \mid \lambda$

$S \Rightarrow AB \Rightarrow aaAB \Rightarrow aaABb \Rightarrow aaBb \Rightarrow aab$

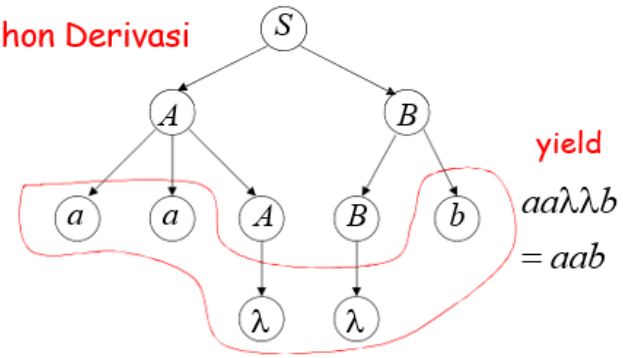
Pohon Derivasi



$S \to AB \qquad A \to aaA \mid \lambda \qquad B \to Bb \mid \lambda$

$S \Rightarrow AB \Rightarrow aaAB \Rightarrow aaABb \Rightarrow aaBb \Rightarrow aab$

Pohon Derivasi



Notasi yang tepat

$A \to aAb$
$A \to \lambda$ ⇒ $A \to aAb \mid \lambda$

$\langle article \rangle \to a$
$\langle article \rangle \to the$ ⇒ $\langle article \rangle \to a \mid the$

7.8 Proses Parsing

Left most derivation → simbol variabel terkiri yang diperluas

Right most derivation → sebaliknya

- Derivasi Order (1)

  1. S → AB
  2. A → aaA
  3. A → λ
  4. B → Bb
  5. B → λ

  Derivasi dari kiri

  1 2 3 4 5

  $S \rightarrow AB \rightarrow aaAB \rightarrow aaB \rightarrow aaBb \rightarrow aab$

  Derivasi dari kanan

  1 4 5 2 3

  $S \rightarrow AB \rightarrow ABb \rightarrow Ab \rightarrow aaAb \rightarrow aab$

- Derivasi Order

  $S \rightarrow aAB$

  $A \rightarrow bBb$

  $B \rightarrow A \mid \lambda$

  Derivasi dari kiri :

  $S \rightarrow aAB \rightarrow abBbB \rightarrow abAbB \rightarrow abbBbbB \rightarrow abbbbB \rightarrow abbbb$

  Derivasi dari kanan :

  $S \rightarrow aAB \rightarrow aA \rightarrow abBb \rightarrow abAb \rightarrow abbBbb \rightarrow abbbb$

7.9 Ambiguitas

tata bahasa dikatakan ambigu bila terdapat lebih dari satu pohon parsing yang berbeda untuk satu untai.

Contoh ambigu :

S → A | B

A → a

B → a

Untuk mendapat untai ‘a’ terdapat 2 cara penurunan, yaitu :

S → A → a atau

S → B → a

- S → SbS | ScS | a,

  Untai ‘abaca’ diturunkan dengan :

  S → SbS → SbScS → SbSca → abaca

  Atau

  S → ScS → SbScS → abScS → abacS → abaca

Ambiguitas dapat menimbulkan masalah pada bahasa tertentu, baik bahasa alami maupun bahasa pemrograman. Bila suatu struktur bahasa memiliki lebih dari suatu dekomposisi (penurunan), dan susunannya akan menentukan arti, maka artinya menjadi ambigu

DAFTAR PUSTAKA

1. Tedy Setiadi, Diktat Teori Bahasa dan Otomata, Teknik Informatika UAD, 2005
2. Hopcroft John E., Rajeev Motwani, Jeffrey D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, 2rd, Addison-Wesley,2000
3. Martin C. John, *Introduction to Languages and Theory of Computation,* McGraw-Hill Internatioanal edition,1991
4. Linz Peter,*Introduction to Formal Languages & Automata,* DC Heath and Company, 1990
5. Dulimarta Hans, Sudiana, *Catatan Kuliah Matematika Informatika, Magister Teknik Informatika ITB, 1998*
6. Hinrich Schütze, IMS, Uni Stuttgart, WS 2006/07, Slides based on RPI CSCI 2400


LABORATORIUM
TEKNIK INFORMATIKA
UNIVERSITAS AHMAD DAHLAN


UNIVERSITAS AHMAD DAHLAN