OTOMATA 02

REGULAR EXPRESSION & FINITE AUTOMATA

MATERI PERTEMUAN

- Gagasan
- Regular Expression
- Finite Automata
- Finite State Diagram
- Deterministic Finite Automata
- Transition Graph
- Automata with Output

GAGASAN REGULAR EXPRESSION

Pada tahun 1940, 2 orang *neuro-physiologist*, **Warren McCulloch** dan **Walter Pitts** mengembangkan sebuah model sistem syaraf yang disebut **Finite Automata**.

Matematikawan **Stephen Kleene** memformalkan model tersebut melalui sebuah sistem aljabar yang berbentuk himpunan regular. Selanjutnya himpunan ini disebut sebagai **Regular Expression**.

REGULAR EXPRESSION (1)

Dikarenakan peruntukkannya, regular expression sering disebut sebagai *language-defining expression* atau *language-defining tool*.

Bahasa apapun yang dapat didefinisikan melalui alat ini disebut *regular language*.

Sebuah himpunan regular expression dapat didefinisikan melalui aturan berikut :

1. λ adalah regular expression;
2. Jika R dan S masing^2 adalah regular expression, maka :
   (i). (R) dan (S) masing^2 adalah regular expression
   (ii). Produk concatenate : RS adalah regular expression
   (iii). Produk union : R+S atau R|S adalah regular expression
   (iv). Produk closure : R* atau S^+ masing^2 adalah regular expression
3. Tidak ada aturan lain untuk membentuk regular expression, selain kedua aturan di atas.

REGULAR EXPRESSION (2)

Ekspresi bahasa seperti di bawah ini :

$L_2 = \{\lambda, x, xx, xxx, xxxx, \ldots\}$

Dapat ditulis berbasiskan simbol seperti berikut :

$S = \{x\}$
$L_2 = S^*$

Atau, bahasa L2 dapat pula dinyatakan sebagai :

$L_2 = \text{Language}(x^*)$

REGULAR EXPRESSION (3)

Contoh :
Terdapat himpunan alphabet $\Sigma = \{a, b\}$
Dan didefinisikan sebuah bahasa L = {a, ab, abb, abbb, abbbb, … }
Melalui RE, L dapat dinyatakan sebagai L = Language(ab^*)

Contoh :
Terdapat himpunan alphabet $\Sigma = \{x\}$
Dan didefinisikan sebuah bahasa L = {x, xx, xxx, … }
Melalui RE, L dapat dinyatakan sebagai L = Language(x^+)

INGAT!

$$xx^* = x^+ = xx^*x^* = x^*xx^* = x^+x^* = x^*x^+ = \ldots$$

REGULAR EXPRESSION (4)

Contoh :

Terdapat himpunan alphabet Σ = {a, b}

Dan didefinisikan sebuah bahasa L = {λ, a, b, aa, bb, ab, ba, aaa, aab, aba, baa, … }

Melalui RE, L dapat dinyatakan sebagai **L = Language(a+b)***

INGAT : **a*b* ≠ (ab)***

Contoh :

Jika terdapat 2 himpunan bahasa P = {a, bb, bab} dan Q = {λ, bbb}

Maka PQ = {a, bb, bab, abbb, bbbbb, babbbb}

Dan RE dari bahasa tersebut adalah **(a + bb + bab) (λ + bbb)**

Contoh :

Jika terdapat 2 himpunan bahasa M = {λ, x, xx} dan Q = {λ, y, yy, yyy, … }

Maka PQ = {λ, y, yy, yyy, …, x, xy, xyy, xyyy, …, xx, xxy, xxyy, xxyyy, … }

Dan RE dari bahasa tersebut adalah **(λ + x + xx) (y*)** atau **y* + xy* + xxy***

REGULAR EXPRESSION (5)

Regular Expression banyak diaplikasikan oleh bahasa-bahasa yang mengolah teks secara langsung (bahasa-bahasa untuk web scripting), seperti PHP, Java, Javascript, Perl, dll.

Selain itu, Regular Expression dapat pula kita jumpai pada :

- **Editor**, seperti Emacs, vi, dll
- **Bahasa Pemrograman**, seperti Delphi, Visual C++, dll
- **Tools Bahasa Pemrograman**, seperti grep (UNIX), lex, sed, dll

REGULAR EXPRESSION (6)

Notasi Regular Expression pada Bahasa PHP

1. String-string yang dipisahkan oleh logical character " | " (**OR**)

   contoh :

No	Pola	Deskripsi
1	apel	Menemukan kata "apel"
2	apel | pisang	Menemukan kata "apel" **atau** "pisang"
3	begin | end | break	Menemukan kata "begin" **atau** "end" **atau** "break"

REGULAR EXPRESSION (7)

2. Pencarian sebuah string dapat dilakukan melalui penggabungan beberapa perintah sekaligus. Sehingga dapat dihasilkan string yang sangat spesifik.

   beberapa perintah tersebut adalah :

Jenis Karakter	Makna/Arti
X*	Sebuah string x dapat diulang nol atau lebih (ingat, kleene closure)
X^+	Sebuah string x dapat diulang satu atau lebih (ingat, positive closure)
$x^?$	Sebuah string dapat diulang nol atau satu kali
x{n}	Sebuah string x harus diulang sebanyak n kali
.	Sebarang string yang ditemukan
^x	Mencari sebarang string dengan karakter awal x
x$	Mencari sebarang string dengan karakter akhir x
[xy]	Menentukan sebuah string dengan karakter antara x atau y
/x	Mencari string x
[:<:]	Karakter awal sebuah string adalah alphanumerik atau underscore
[:>:]	Karakter akhir sebuah string adalah alphanumerik atau underscore

REGULAR EXPRESSION (8)

Fungsi-Fungsi Regular Expression pada Bahasa PHP

No	Nama Fungsi	Deskripsi
1	**ereg()**	Memberikan nilai TRUE (jika sebuah pola ditemukan) atau FALSE (jika pola tidak ditemukan)
2	**eregi()**	Sama dengan ereg(), tetapi tidak bersifat case-sensitive
3	**ereg_replace()**	Mengganti sebuah substring dengan substring yang baru
4	**eregi_replace()**	Sama dengan ereg_replace(), tetapi tidak bersifat case-sensitive
5	**split()**	Memecah sebuah string menjadi beberapa substring

FINITE AUTOMATA

Bahasa formal dapat dipandang sebagai himpunan entitas abstrak, berupa sekumpulan string yang berisi simbol[2] alphabet.

Dalam bentuk entitas abstrak, bahasa dapat dikenali atau dibangkitkan oleh mesin komputasi.

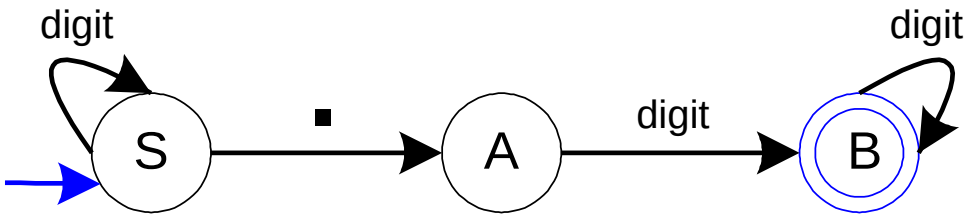
Mesin komputasi yang sesuai untuk kelas bahasa ini adalah Finite (State) Automata.

Finite Automata adalah sebuah model matematika dengan input dan output diskrit.

Untuk menggambarkan perilaku Finite Automata digunakan Finite State Diagram atau State Transition Diagram.

FINITE STATE DIAGRAM (1)

Sebuah FSD untuk memeriksa validitas penulisan bilangan riil dengan setidaknya 1 titik desimal.

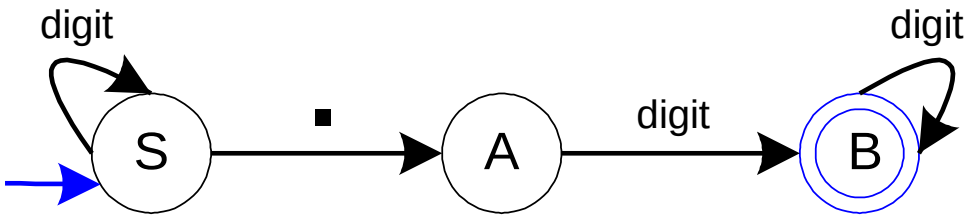


Contoh : **untuk string 9.8765**

(S, 9.8765)	⇒	(S, .8765)	dibaca 9 dan FSD tetap di state S
	⇒	(A, 8765)	dibaca . dan FSD ada di state A
	⇒	(B, 765)	dibaca 8 dan FSD ada di state B
	⇒	(B, 65)	dibaca 7 dan FSD ada di state B
	⇒	(B, 5)	dibaca 6 dan FSD ada di state B
	⇒	(B,)	dibaca 5 dan FSD tetap di state B

(karena B merupakan Final State, maka penulisan 9.8765 dinyatakan benar oleh bahasa tsb)

FINITE STATE DIAGRAM (2)



Input string a → Tidak dikenali oleh FSD

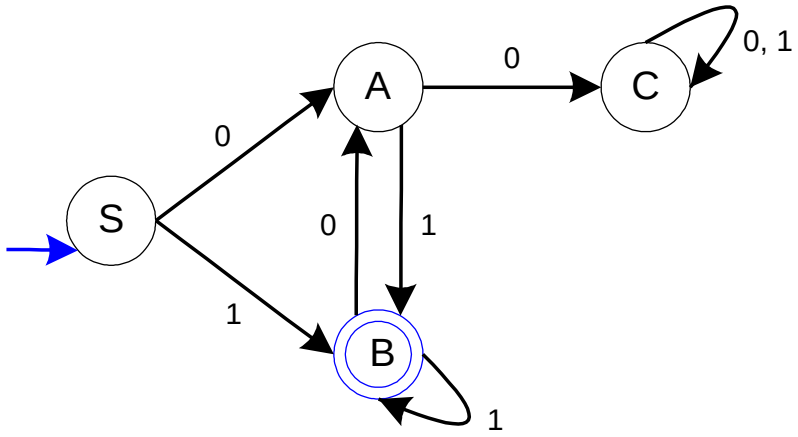
Input string 9 → Penelusuran berhenti di S (bukan final state)

Input string 9. → Penelusuran berhenti di A (bukan final state)

Input string 98765 → Penelusuran berhenti di S (bukan final state)

FINITE STATE DIAGRAM (3)

Sebuah FSD untuk memeriksa validitas bahasa L = { x ∈ (0 + 1)⁺ | dengan karakter terakhir pada string x adalah 1 dan x tidak memiliki substring 00 }



Contoh :

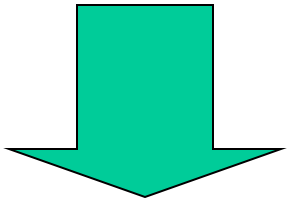
Untuk input string 111 → Dikenali oleh FSD (*accepted*)

Untuk input string 101 → Dikenali oleh FSD (*accepted*)

Untuk input string 1001 → Penelusuran berakhir tidak di B (tidak berakhir di final state)

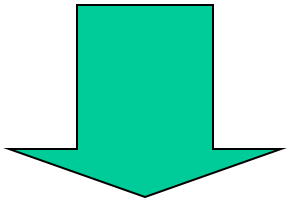
DETERMINISTIC FINITE AUTOMATA (1)

Pernah bermain ular dan tangga ?!



Komponen-komponen permainannya apa saja ?

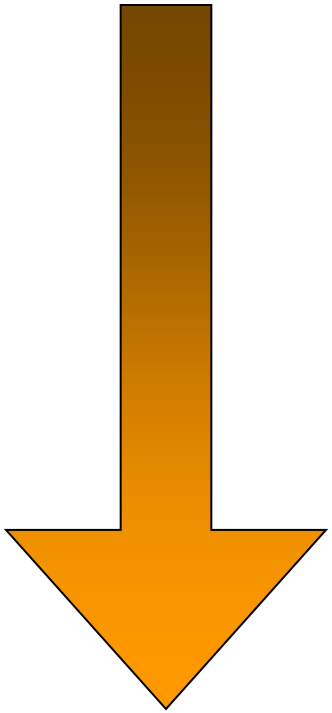
n x n kotak, **Dadu**, **Penanda**, **Pemain**



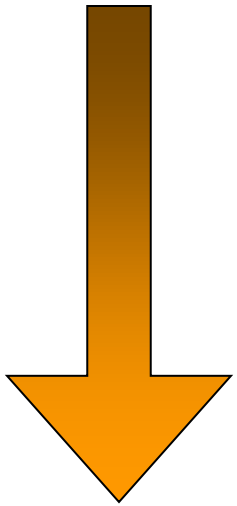
Bagaimana karakteristik permainannya ?

DETERMINISTIC FINITE AUTOMATA (2)

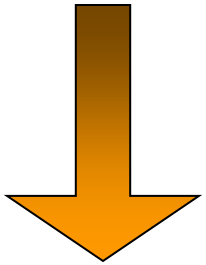
Pola/arah perpindahan kotak/state dalam permainan tersebut bersifat tertentu/mutlak



Jumlah kotak, jumlah dan jenis karakter input terbatas/berhingga



Pergerakan pemain (seolah^2) bersifat otomatis (ditentukan oleh hasil lemparan dadu)



Deterministic Finite Automata (DFA)

DETERMINISTIC FINITE AUTOMATA (3)

Secara definitif, DFA memiliki komponen-komponen :

1. S sebagai himpunan berhingga **state** untuk media perpindahan kendali mesin
2. Σ sebagai himpunan berhingga alphabet untuk **input karakter**
3. s_0 adalah salah satu state dari himpunan S yang diperlakukan sebagai **start state**
4. s_n adalah salah satu state dari himpunan S yang diperlakukan sebagai **final state**

   (DFA dapat memiliki lebih dari satu final state)
5. δ sebagai himpunan berhingga **fungsi transisi** untuk memindahkan kendali mesin

DETERMINISTIC FINITE AUTOMATA (3)

Contoh sebuah DFA :

Himpunan State S = {X, Y, Z}

- Himpunan alphabet Σ = {a, b}
- X $\in$ S sebagai start state
- Z $\in$ S sebagai final state

- Himpunan fungsi transisi δ didefinisikan sebagai :
  1. Dari X diberi input a ke Y
  2. Dari X diberi input b ke Z
  3. Dari Y diberi input a ke X
  4. Dari Y diberi input b ke Z
  5. Dari Z diberi input a atau b ke Z

DETERMINISTIC FINITE AUTOMATA (4)

Contoh sebuah DFA :

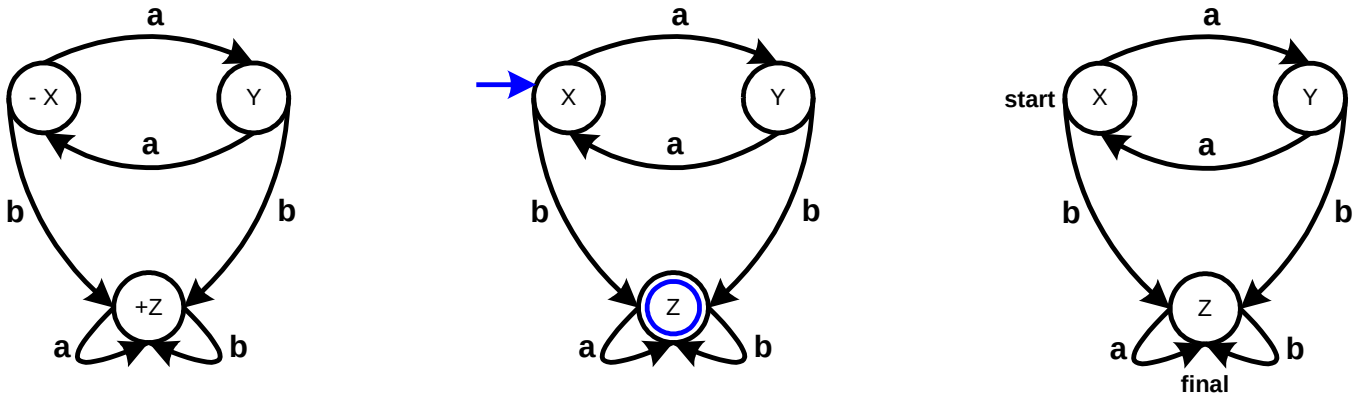
Himpunan State S = {X, Y, Z}

- Himpunan alphabet Σ = {a, b}
- X $\in$ S sebagai start state
- Z $\in$ S sebagai final state
- Himpunan fungsi transisi δ didefinisikan sebagai :
  1. Dari X diberi input a ke Y
  2. Dari X diberi input b ke Z
  3. Dari Y diberi input a ke X
  4. Dari Y diberi input b ke Z
  5. Dari Z diberi input a atau b ke Z

Tabel Transisi

	a	b
(start) X	Y	Z
Y	X	Z
(final) Z	Z	Z

Labelled Directed Graph (representasi piktorial)



TRANSITION GRAPH (1)

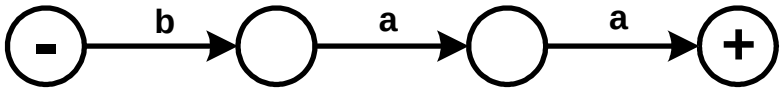
Secara definitif, TG tidak berbeda dengan DFA. Namun dari segi karakteristik, terdapat beberapa perbedaan mendasar antara TG dengan DFA. Perbedaan-perbedaan tersebut antara lain :

- label edge pada TG dapat berisi string (lebih dari satu karakter)
- jumlah outgoing edge dari setiap state dapat lebih dari satu dengan label edge sama (oleh karenanya TG tidak bersifat deterministic (non-deterministic). Sehingga setiap input string belum tentu memiliki path yang unik)
- TG dapat memiliki lebih dari satu Start State

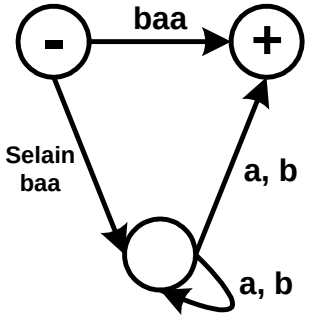
TRANSITION GRAPH (2)

Contoh :

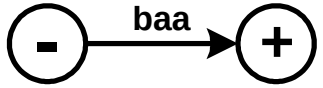
DFA :



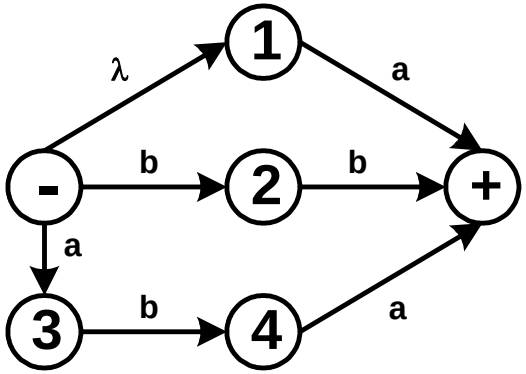
TG :



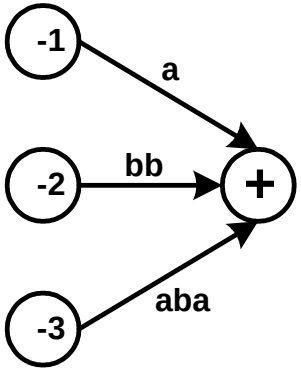
atau,



DFA :



TG :



FINITE AUTOMATA with OUTPUT (1)

Sampai sejauh ini automata yang kita pelajari hanya dapat memberikan hasil/jawaban berupa "ACCEPTED' atau "REJECTED" saja. Untuk sementara, mesin kita ini hanya berfungsi sebagai Acceptor saja.

Tapi sekarang, kita akan mencoba membuat sebuah Transducer. Yaitu sebuah mesin yang dapat menghasilkan berbagai macam bentuk output (*automata with output*).

2 model *automata with output* yang tersedia, yaitu :

- Moore Machine
- Mealy Machine

FINITE AUTOMATA with OUTPUT (2)

MOORE MACHINE

Secara definitif, Moore Machine memiliki komponen-komponen :

1. Q sebagai himpunan berhingga **state** untuk media perpindahan kendali mesin.

   Label pada setiap state ditulis q_i/o, dengan q_i adalah nama state dan o adalah output yang berkorespondensi dengan state tsb;
2. Σ sebagai himpunan berhingga alphabet untuk **input karakter**
3. Γ sebagai himpunan berhingga karakter untuk **output karakter**
4. q_0 adalah salah satu state dari himpunan S yang diperlakukan sebagai **start state**
5. δ sebagai himpunan berhingga **fungsi transisi** untuk memindahkan kendali mesin

FINITE AUTOMATA with OUTPUT (3)

Contoh :

Sebuah Moore Machine didefinisikan sbb :

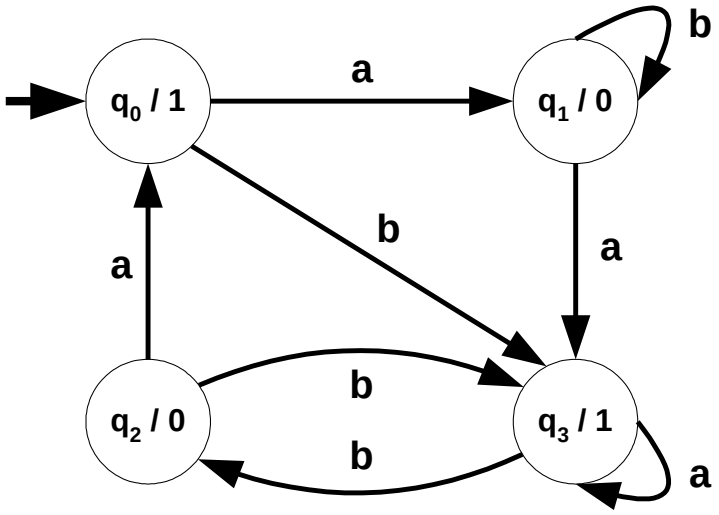
Himpunan state Q = {q_0, q_1, q_2, q_3} dengan q_0 sebagai start state

Himpunan input karakter Σ = {a, b}

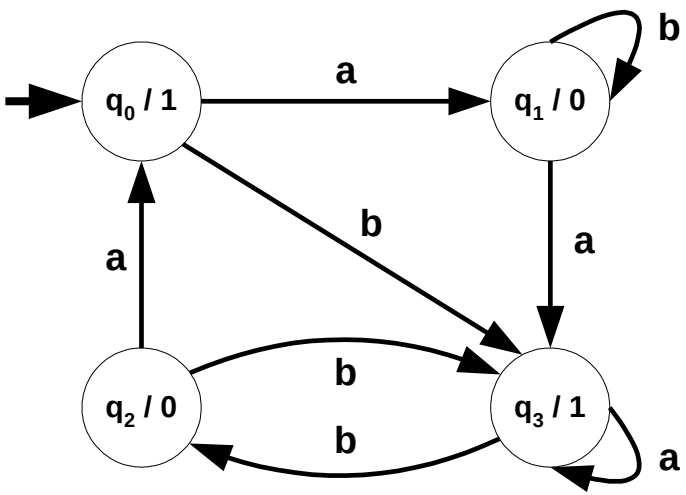
Himpunan output karakter Γ = {0, 1}

Himpunan fungsi transisi didefinisikan pada tabel berikut :

State Awal	Input a	Input b	Output
- q_0	q_1	q_3	1
q_1	q_3	q_1	0
q_2	q_0	q_3	0
q_3	q_3	q_2	1



FINITE AUTOMATA with OUTPUT (4)



Sebuah input string abab
akan diproses seperti berikut :

State Awal	Input	State Tujuan	Output
- q_0			1
q_0	a	q_1	1 0
q_1	b	q_1	1 0 0
q_1	a	q_3	1 0 0 1
q_3	b	q_2	1 0 0 1 0

Atau proses penelusurannya dapat pula digambarkan seperti berikut :

Input		a	b	a	b
State	q_0	q_1	q_1	q_3	q_2
Output	1	0	0	1	0

FINITE AUTOMATA with OUTPUT (5)

MEALY MACHINE

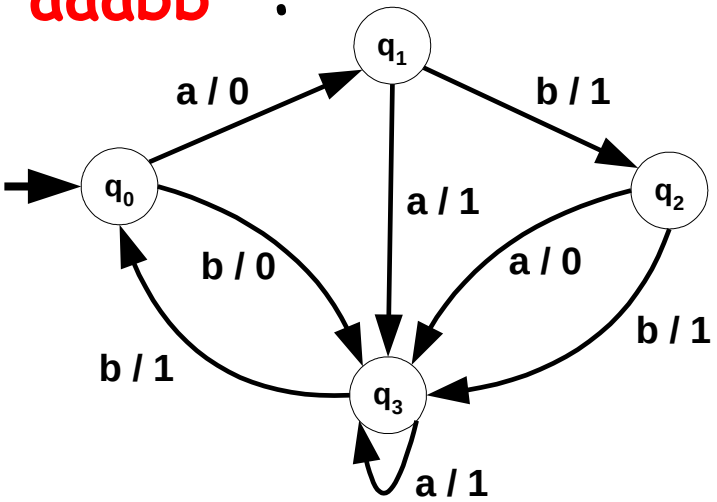
Secara definitif, Mealy Machine memiliki komponen-komponen :

1. Q sebagai himpunan berhingga **state** untuk media perpindahan kendali mesin.
2. Σ sebagai himpunan berhingga alphabet untuk **input karakter**
3. O sebagai himpunan berhingga karakter untuk **output karakter**
4. q_0 adalah salah satu state dari himpunan S yang diperlakukan sebagai **start state**
5. δ sebagai himpunan berhingga **fungsi transisi** untuk memindahkan kendali mesin.

   Label pada setiap panah ditulis ε/o, dengan ε adalah input karakter dan o adalah output karakter yang berkorespondensi dengan kedua adjacent state tsb.

FINITE AUTOMATA with OUTPUT (6)

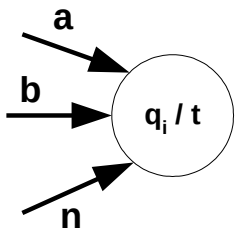
Sebuah contoh Mealy Machine dan penelusurannya untuk input **aaabb** :



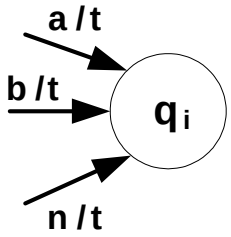
Input		a	a	a	b	b
State	q_0	q_1	q_3	q_3	q_0	q_3
Output		0	1	1	1	0

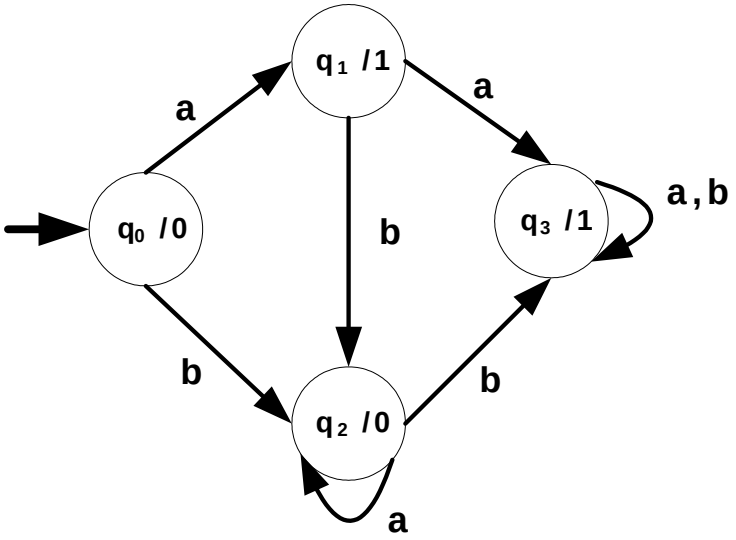
FINITE AUTOMATA with OUTPUT (7)

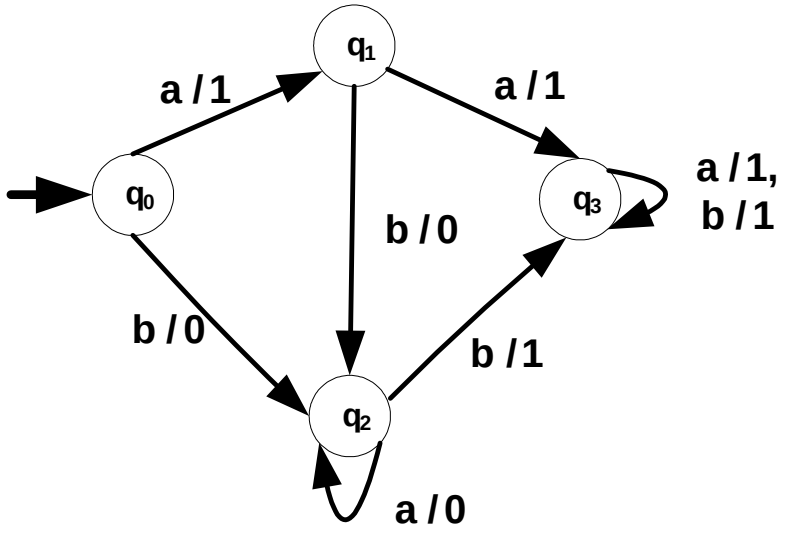
Proses konversi dari Moore Machine ke bentuk Mealy Machine dapat dilakukan dengan sangat mudah.



menjadi

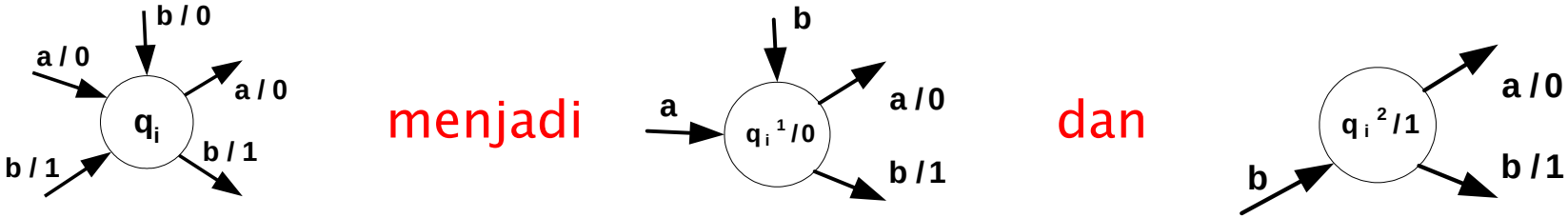






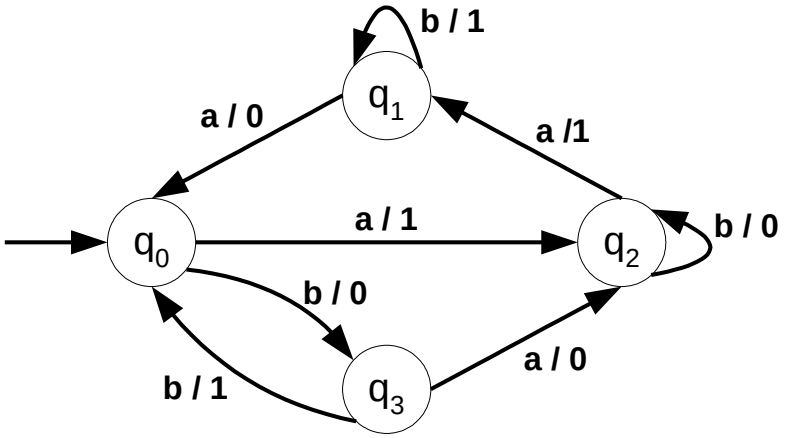
FINITE AUTOMATA with OUTPUT (8)

Tetapi sebaliknya, konversi dari Mealy Machine ke bentuk Moore Machine agak sedikit 'ribet'.



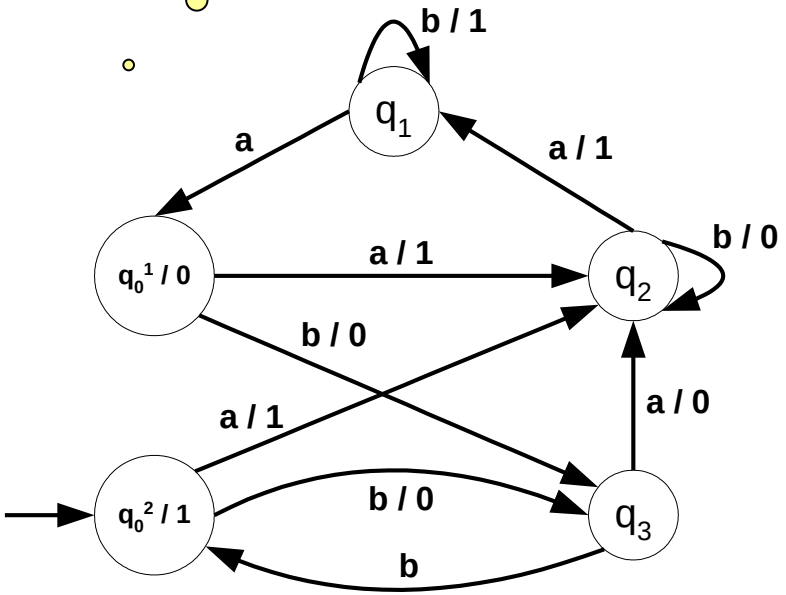
FINITE AUTOMATA with OUTPUT (9)

Mealy Machine berikut dapat dikonversikan menjadi Moore Machine :



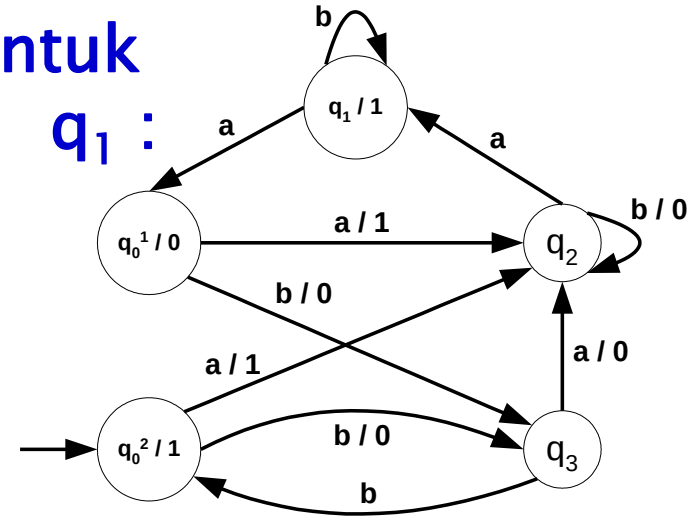
Gambarkan dulu semua state & edge yg tidak terlibat dg proses konversi saat itu

Untuk q_0 :

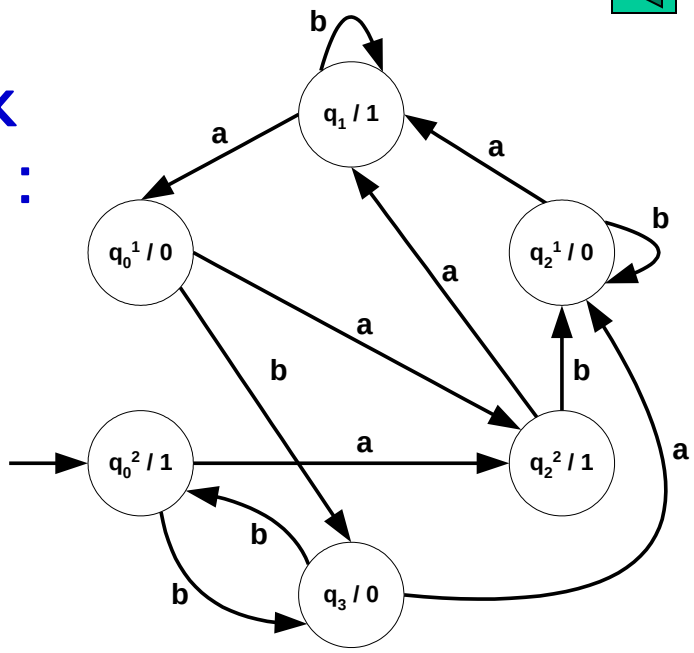


FINITE AUTOMATA with OUTPUT (10)

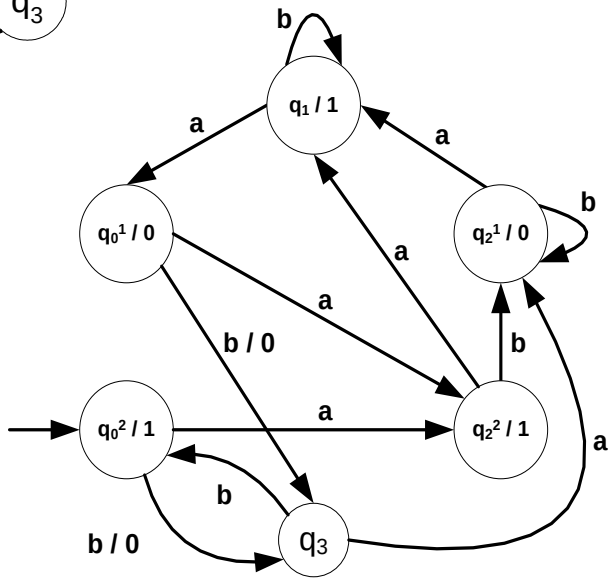
Untuk q_1 :



Untuk q_3 :



Untuk q_2 :



LATIHAN

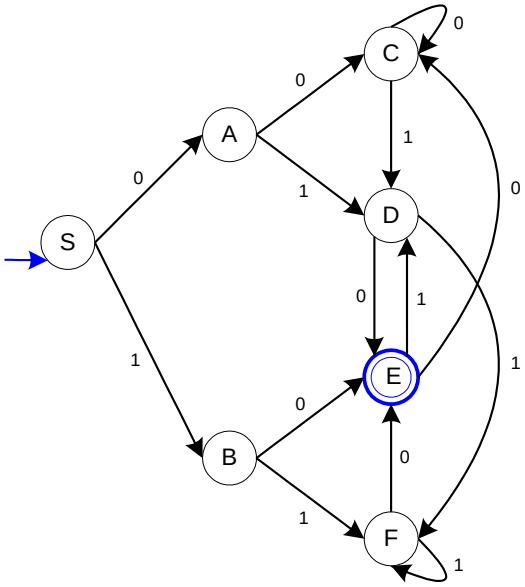
1. Misal terdapat sebuah himpuan alphabet S = {aa, ab, ba, bb}
   a. Deskripsikan RE untuk bahasa S*
   b. Berikan contoh himpunan string pada S* dimana string-string tersebut mempunyai jumlah a dan b habis dibagi 3

2. Buatlah RE untuk bahasa yang didefinisikan dari himpunan alphabet Σ = {a, b} :
   a. Semua string dalam bahasa tersebut tidak memiliki substring ab
   b. Semua string dalam bahasa tersebut memiliki jumlah a genap dan b ganjil
   c. Semua string dalam bahasa tersebut memiliki 3 karakter b

3. Tunjukkan bahwa pasangan RE berikut adalah ekivalen :
   a. (a* + b)* dan (a + b)*
   b. ((a + bb)*aa)* dan λ + (a + bb)*aa
   c. λ + a(a + b)*aa(a + b)* dan ((b*a)*ab*)*

LATIHAN

4. Amatilah lingkungan hidup anda sehari-hari. Tentukan sebuah obyek (misalnya, sistem lift, sistem traffic light, sistem perpanjangan STKB, dll) yang anda dapat gambarkan/modelkan dengan Automata. **(no groups tells the same stories & no groups adopts my examples above)**



5. Lakukan penelusuran input-input string berikut pada FSD di atas :

a. 10	c. 110	e. 0010
b. 010	e. 110010	f. 001

TUGAS INDIVIDU

Buku Introduction to Computer Theory 2nd edition (Cohen)

Hal 71-75: 17, 19

Hal 164-168: 8, 9, 10