Week 6
Ekspresi Reguler

Ana Muliyana

Del Institute of Technology
Indonesia

Ekspresi Reguler

Ekspresi reguler (regex) adalah notasi formal untuk mendeskripsikan himpunan string dalam suatu bahasa formal

Ekspresi reguler dan finite automata ekivalen secara matematis, artinya:

1. Setiap bahasa yang bisa dinyatakan dengan ekspresi reguler dapat dikenali oleh finite automata.
2. Sebaliknya, setiap bahasa yang dikenali finite automata dapat dituliskan dengan ekspresi reguler.

Notasi Ekspresi Reguler

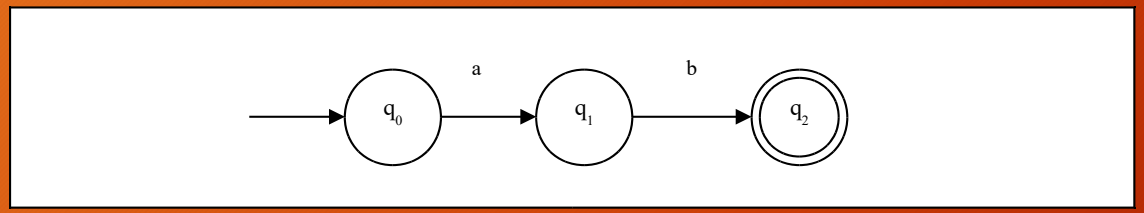
Notasi	Arti	Contoh	Hasil
$*$	Asterisk: bisa tidak muncul/ bisa muncul berhingga kali (0-n)	a^*	$\{\varepsilon, a, aa, aaa, aaaaa, \ldots\}$
$+$	(Pada posisi superscript/diatas) berarti minimal muncul satu kali (1-n)	a^+	$\{a, aa, aaa, aaaaa, \ldots\}$
$+ \; atau \; \cup$	Berarti union (dibaca atau)	$a \cup b$	$\{a, b, aa, bbb, aaaaaa, bbbbbb\}$
$.$	Berarti konkatenasi	$a.b$	$\{ab\}$

Contoh ER

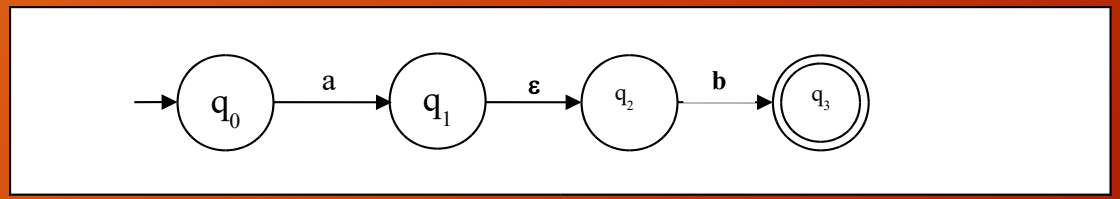
1. ER : ab*cc Contoh string yang bisa dibangkitkan abcc, acc, abbcc, abbbcc, dst. (b bisa tidak muncul atau muncul sejumlah berhingga kali)
2. ER : 010* Contoh string yang bisa dibangkitkan 01,010, 0100,01000, dst. (0 bisa tidak muncul atau muncul sejumlah berhingga kali)
3. ER : $a^{+}d$ Contoh string yang bisa dibangkitkan ad,aad, aaad,aaaad dst. (a minimal muncul satu kali)
4. ER : a* U b* Contoh string yang bisa dibangkitkan a, b, aa, bb, dst.
5. ER : 01*+0 Contoh string yang bisa dibangkitkan 0, 01,011, dst.

Contoh ER dengan FSA

NFA untuk ER: ab



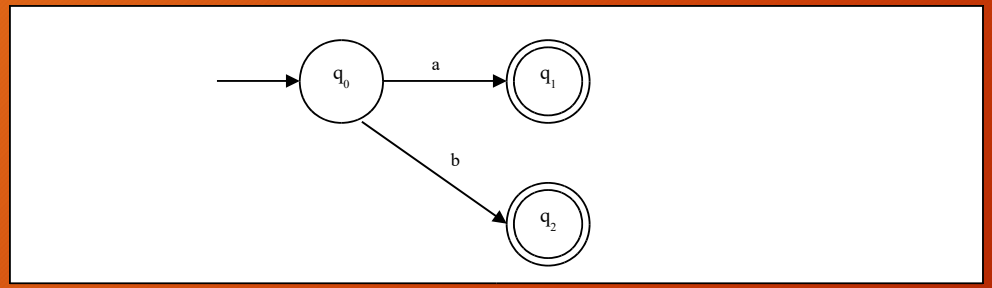
NFA



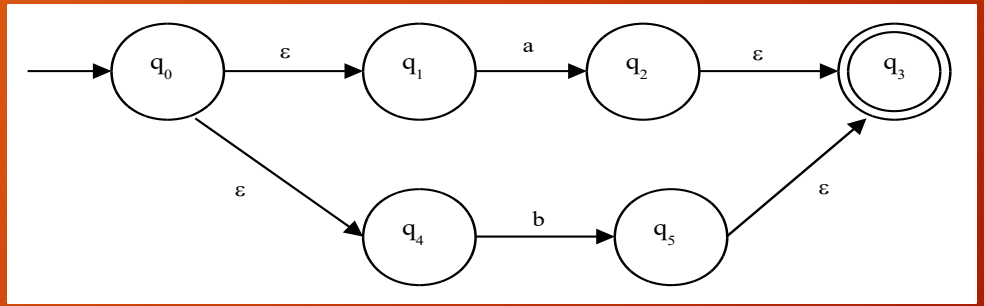
NFA $\varepsilon - move$

Contoh ER dengan FSA

NFA untuk ER: aUb

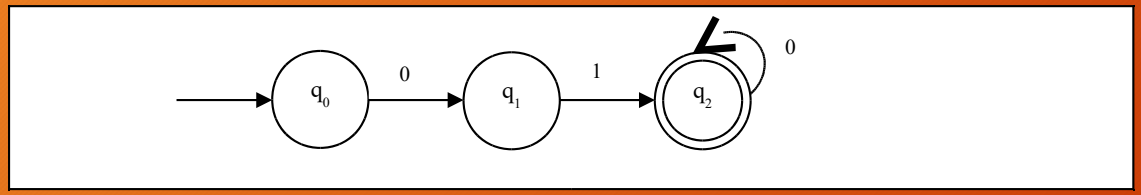


NFA

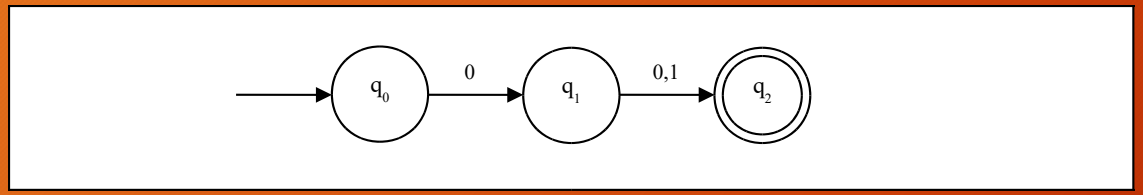


NFA $\varepsilon - move$

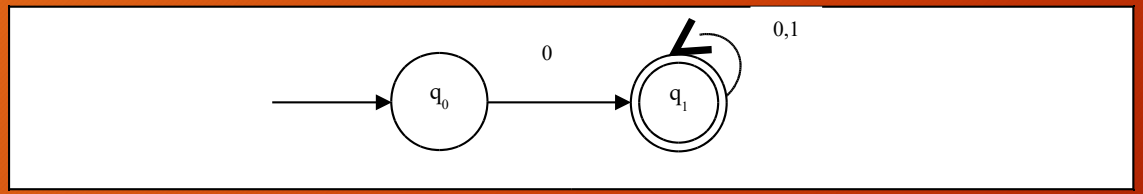
NFA untuk ER: 010*



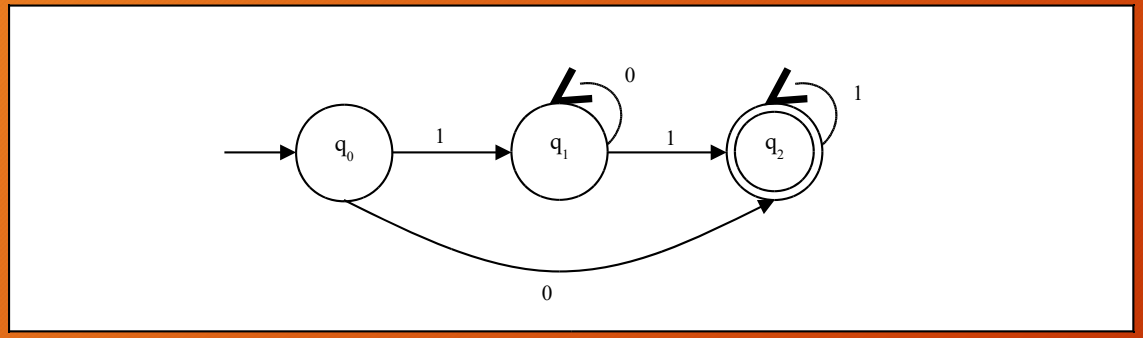
NFA untuk ER: 0 (1∪0)



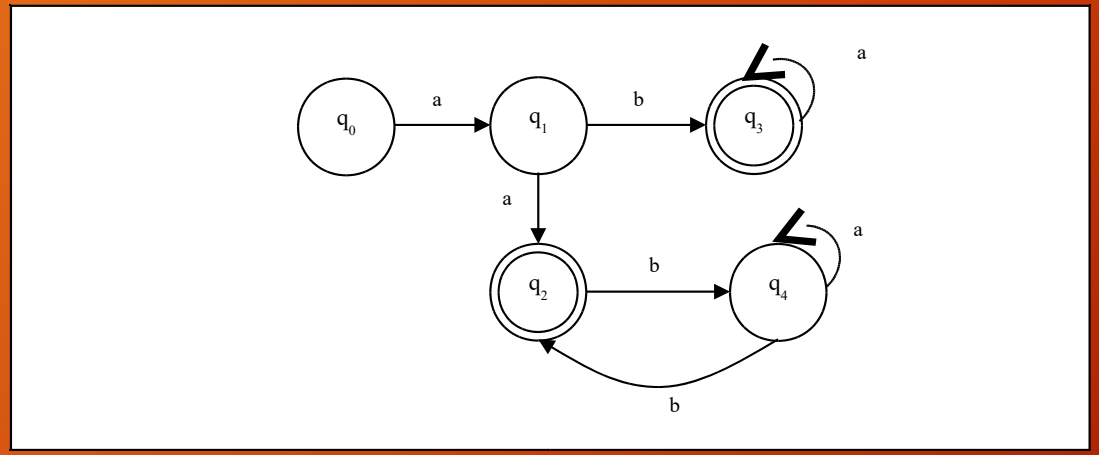
NFA untuk ER: 0 (1∪0)*



NFA untuk ER: 01*∪10*11*



NFA untuk ER: a(ba*∪a(ba*b)*)



Aturan Produksi untuk FSA

Tata Bahasa (grammar) didefinisikan dengan empat (4) tupel G = ({V, T, P, S}) dimana :
V = Himpunan simbol variabel / non terminal
T = Himpunan simbol terminal
P = Kumpulan aturan produksi
S = Simbol awal

Aturan Produksi untuk FSA

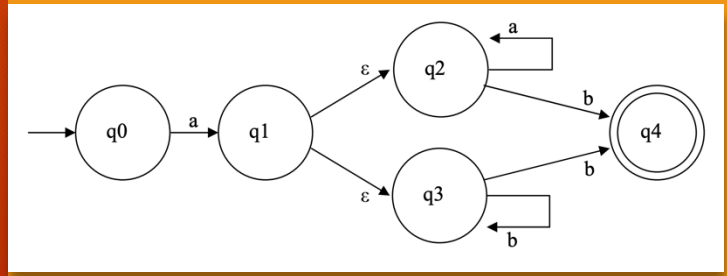
Sebuah otomata berhingga menspesifikasikan sebuah bahasa sebagai himpunan semua untai yang menggerakkannya dari state awal ke salah satu dari state yang diterimanya (himpunan state akhir). Otomata berhingga pada gambar 1 menerima ekspresi regular:

$$a(a^* \cup b^*)b$$

Selain dengan ekspresi regular, kita dapat mengkonstruksi aturanaturan produksi untuk suatu tata bahasa regular. Kita ingat juga batasan aturan produksi untuk bahasa regular:

$$\propto \rightarrow \beta$$

adalah sebuah symbol variabel



Mengkonstruksi Aturan Produksi dari Suatu FSA

Untuk mengkonstruksi aturan produksi dari suatu *finite state automata* (FSA), setiap *state* dalam FSA diubah menjadi sebuah simbol non-terminal, dan setiap transisi (perubahan *state* karena sebuah simbol masukan) diubah menjadi aturan produksi.
Jika suatu *state* adalah *state* akhir, maka simbol non-terminal yang diwakilinya dapat menghasilkan simbol kosong (ε), dan aturan produksinya adalah $X \rightarrow \varepsilon$, di mana X adalah simbol non-terminal tersebut

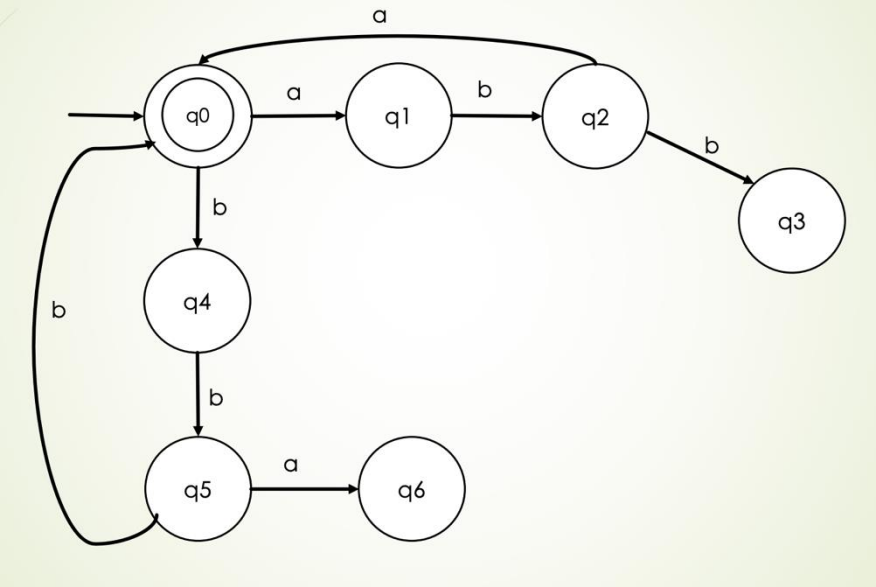
Langkah-langkah Konstruksi Aturan Produksi dari FSA:

Prinsip Dasar

1. Setiap **state** pada FSA direpresentasikan sebagai **non-terminal** dalam grammar.
2. Setiap **transisi** pada FSA akan menjadi **aturan produksi**.
3. **State awal** → menjadi **start symbol** grammar.
4. **State akhir** → menghasilkan aturan produksi ke string kosong (ε)

Konversi FSA ke Grammar

1. Identifikasi state dalam FSA → buat non-terminal.
   Misalnya state = {q0, q1, q2} → Non-terminal = {S, A, B}.
2. Tentukan start state → menjadi start symbol grammar.
   Jika start = q0 → maka S.
3. Buat aturan produksi berdasarkan transisi:
   Jika ada transisi q0 --a--> q1, maka aturannya:
   S → aA (S mewakili q0, A mewakili q1).
4. Tambahkan aturan produksi ke ε untuk final state:
   Jika q1 adalah final state → tambahkan aturan:
   A → ε



Hasil :

$V = \{ A, A, B, D, C\}$

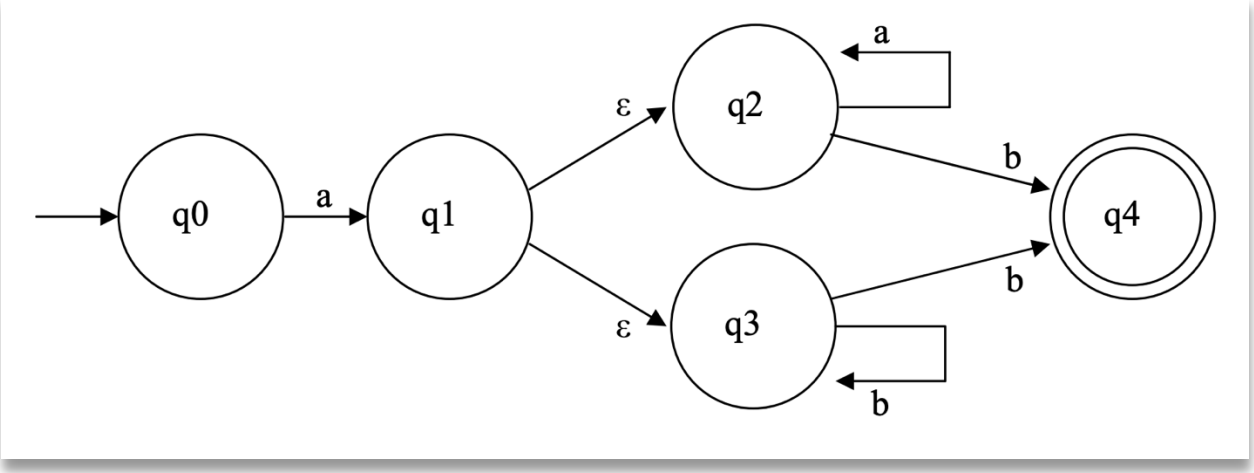
$S \rightarrow aA|bB|e$

$A \rightarrow bC$

B$\rightarrow bD$

$C \rightarrow aS$

$D \rightarrow bS$


q0
a
q1
ε
q2
a
b
q4
ε
q3
b
b

Langkah-langkah Konstruksi FSA untuk suatu Tata Bahasa Reguler

1. Mencermati aturan produksi dengan baik, perhatikan dan perkirakan beberapa variabel yang akan ada hubungannya dengan state yang akan dibangun.
2. Mencermati aturan produksi dengan memperhatikan hubungan antara input dan state yang mungkin akan terbentuk dari langkah 1.
3. Hubungkan state dengan input sesuai dengan aturan produksi
4. Tentukan final state sesuai dengan urutan aturan produksi. Final state dapat dilihat dari akhir alur aturan produksi yang hanya menuju ke terminal.

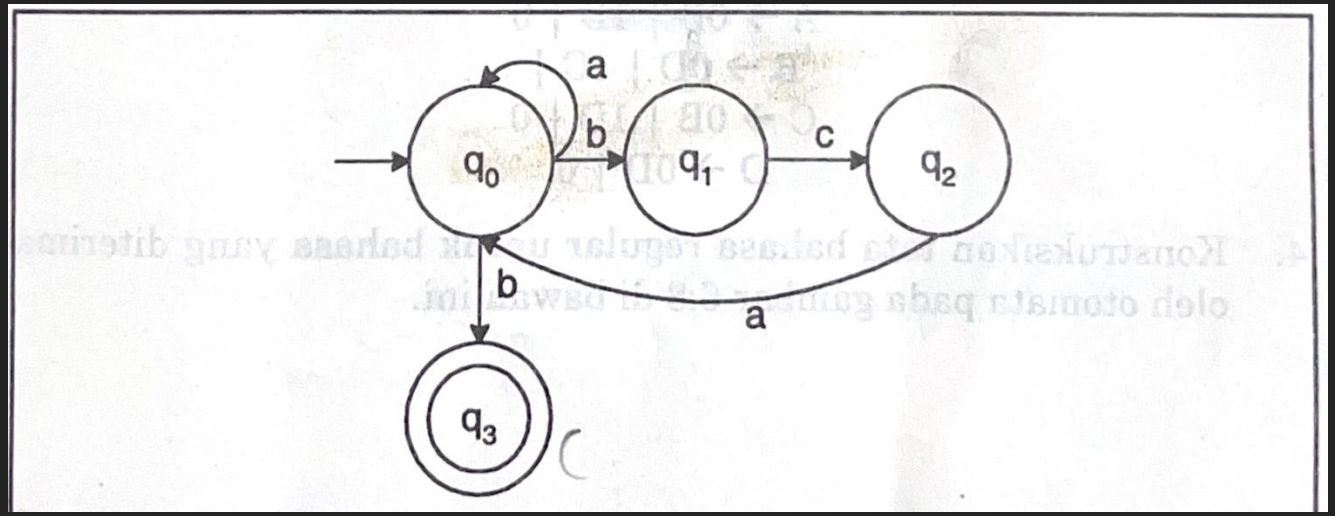
Contoh

Terdapat aturan produksi :

$S \rightarrow aS \mid bB \mid b$

B → cC

C → aS



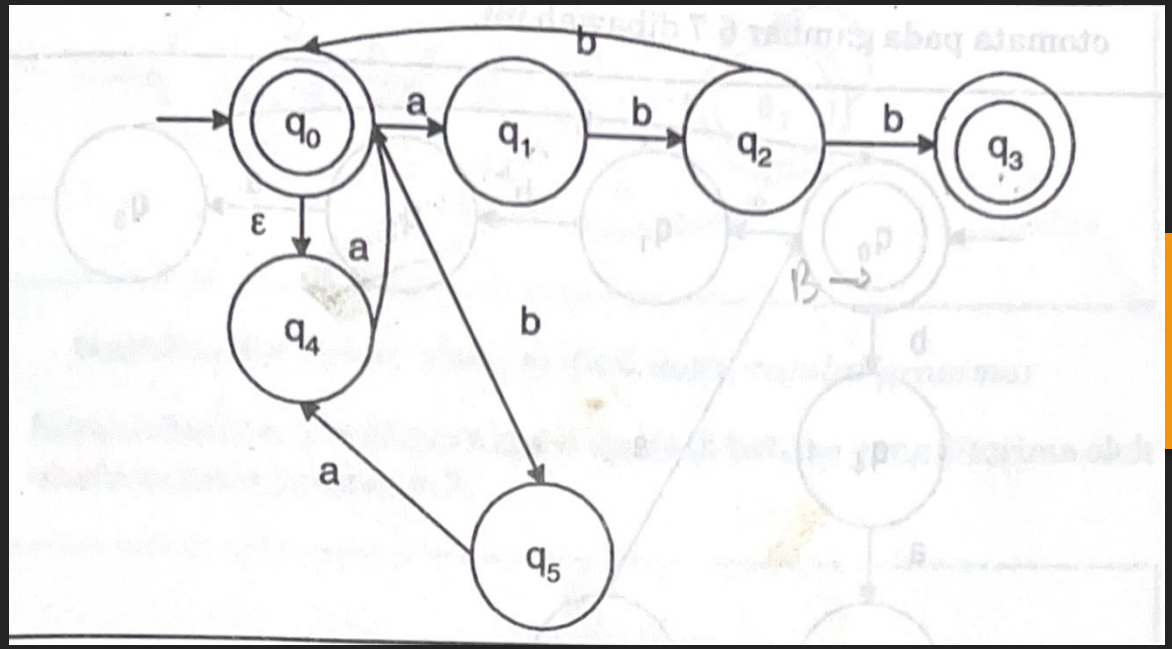
Contoh

Terdapat aturan produksi :

$S \rightarrow abA \mid B \mid baB \mid \varepsilon$

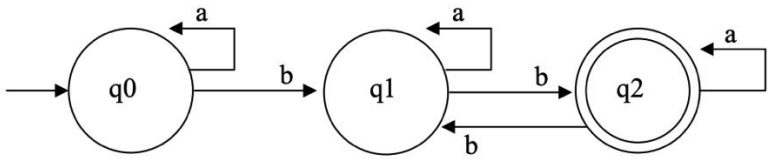
A → bS | b

B → aS



Latihan

1. Diketahui FSA sebagai berikut :



Kontruksi tata bahasa regular nya !

2. Buatlah FSA dari aturan-aturan produksi tata bahasa regular berikut ini :

S → 0A

A → 10A | ε

3. Buatlah FSA dari aturan-aturan produksi tata bahasa regular berikut ini :

A → 00B | 11D | 0

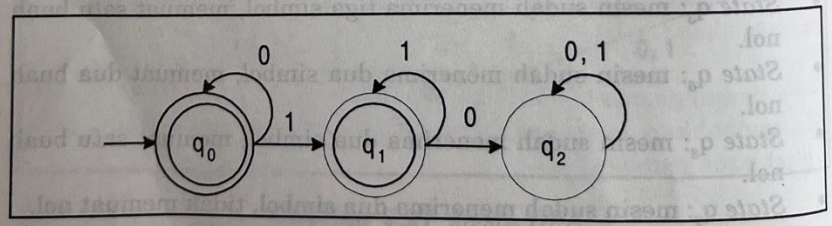
B → 00D | 11C | 1

C → 00B | 11D | 0

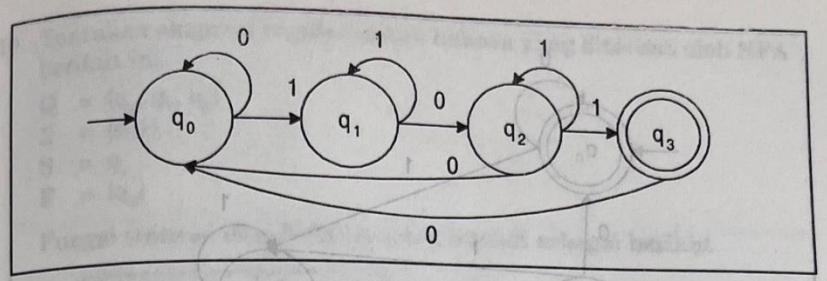
D → 00D | 1

LATIHAN

1. Deskripsikan dalam bahasa Indonesia himpunan *string* yang diterima oleh *finite state automata* seperti dalam:
   a. Gambar 5.31
   b. Gambar 5.32
   c. Gambar 5.33



Gambar 5.31 *Mesin FSA*



Gambar 5.32 *Mesin FSA*

3. Deskripsikan dalam bahasa Indonesia himpunan *string* yang dinyatakan dalam ekspresi regular berikut.

   $(11 + 0)^*(00 + 1)^*$

4. Bentuklah *finite state automata* dari ekspresi regular berikut.

   $10+(0+11)0^*1$