MODUL IX

TEORI BAHASA DAN AUTOMATA

Tujuan :

1. Mahasiswa memahami aturan produksi suatu finite state automata dan dapat merekonstruksi kembali FSA dari suatu bahasa reguler.
2. Mahasiswa mengenal pengembangan lebih jauh dari suatu mesin otomata dan penggunaannya dalam pembuatan keputusan

Materi :

A.

- Aturan produksi bahasa Regular
- Mengkonstruksi Aturan Produksi dari Suatu *Finite State Automata*
- Finite State Automata untuk Suatu Tata Bahasa Regular.

B.

- FSA dengan output
  - Mesin Moore
  - Mesin Mealy

ATURAN PRODUKSI
SUATU FINITE STATE AUTOMATA

Aturan Produksi Bahasa Regular

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 aturan-aturan produksi untuk suatu tata bahasa regular. Kita ingat juga batasan aturan produksi untuk bahasa regular:

$$\alpha \rightarrow \beta$$

α adalah sebuah symbol variabel

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

(bisa dibaca : α menghasilkan β), dimana α atau bisa berupa symbol terminal atau symbol non-terminal/variabel. Simbol variabel/non-terminal adalah symbol yang masih bisa diturunkan, sedang symbol terminal sudah tidak bisa diturunkan lagi. Simbol terminal biasanya dinyatakan dengan huruf kecil, missal a,b,c. Simbol non-terminal/variabel biasanya dinyatakan dengan huruf besar, missal A,B,C. Suatu tata bahasa (grammar) didefinisikan dengan 4 tupel ($G=\{V,T,P,S\}$), dimana:

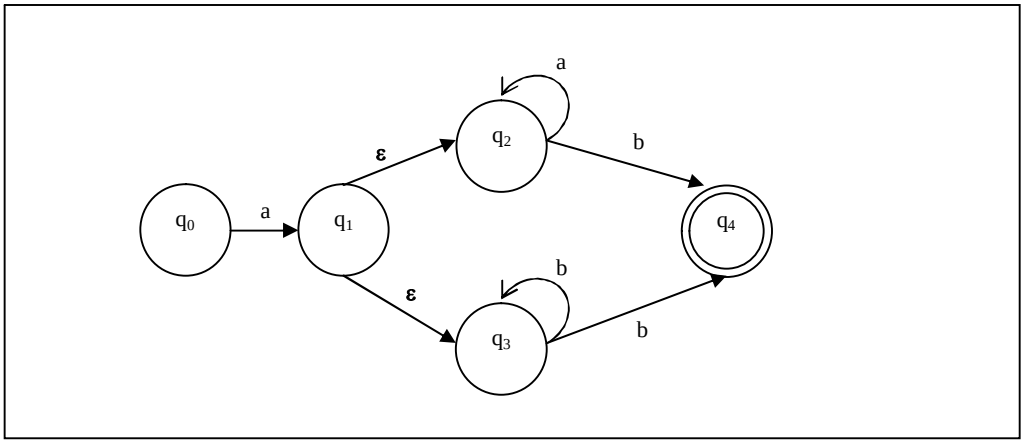
V = himpunan symbol variabel/non terminal

T = himpunan symbol terminal

P = kumpulan aturan produksi

S = symbol awal

Selanjutnya akan kita lihat bagaimana membuat kumpulan aturan produksi untuk suatu *finite state automata*.



Gambar 1. Mesin FSA

Mengkonstruksi Aturan Produksi dari Suatu *Finite State Automata*

Dalam mengkonstruksi aturan produksi tata bahasa regular dari suatu *finite state automata*, perlu kita ingat yang menjadi perhatian kita adalah *state-state* yang bisa menuju ke *state* akhir. Mesin *finite state automata* di gambar 1 memiliki *input* 'a' dan 'b'. Simbol 'a' dan 'b'akan menjadi symbol terminal pada aturan produksi yang akan kita bentuk. Misalnya kita tentukan symbol awal adalah S. Kita identikkan S dengan state awal q_0. Dari q_0 mendapat input a menjadi q_1. Bisa kita tuliskan sebagai aturan produksi:

$$S \rightarrow aE$$

dimana E kita identikkan dengan q_1 (sebenarnya lebih tepatnya adalah bagian yang belum terbangkitkan mulai dari *state* q_1). Kita bisa menambahkan symbol variabel baru setiap kali kita perlukan. Dari q_1 mendapat transisi ε (tanpa menerima *input*) ke q_2 dan q_3. Kita tuliskan:

$$E \rightarrow A$$
$$E \rightarrow B$$

bila kita identikkan q_2 sebagai A, dan q_3 sebagai B. Dari q_2 mendapat *input* a tetap ke q_2, dan dari q_3 mendapat *input* b tetap ke q_3, bisa kita tuliskan:

$$A \rightarrow aA$$
$$B \rightarrow bB$$

Selanjutnya kita lihat dari q_2 mendapat *input* b ke q_4, dan dari q_3 mendapat input b ke q_4, sementara q_4 *state* akhir dan dari q_4 tidak ada lagi busur keluar, maka bisa kita tuliskan:

$$A \rightarrow b$$

$$B \rightarrow b$$

Kumpulan aturan produksi yang kita peroleh bisa kita tuliskan sebagai berikut:

$$S \rightarrow aE$$

$$E \rightarrow A \mid B$$

$$A \rightarrow aA \mid B$$

$$B \rightarrow bB \mid b$$

Ingat '|' berarti atau. Secara formal tata bahasa yang diperoleh dari otomata pada gambar 1:

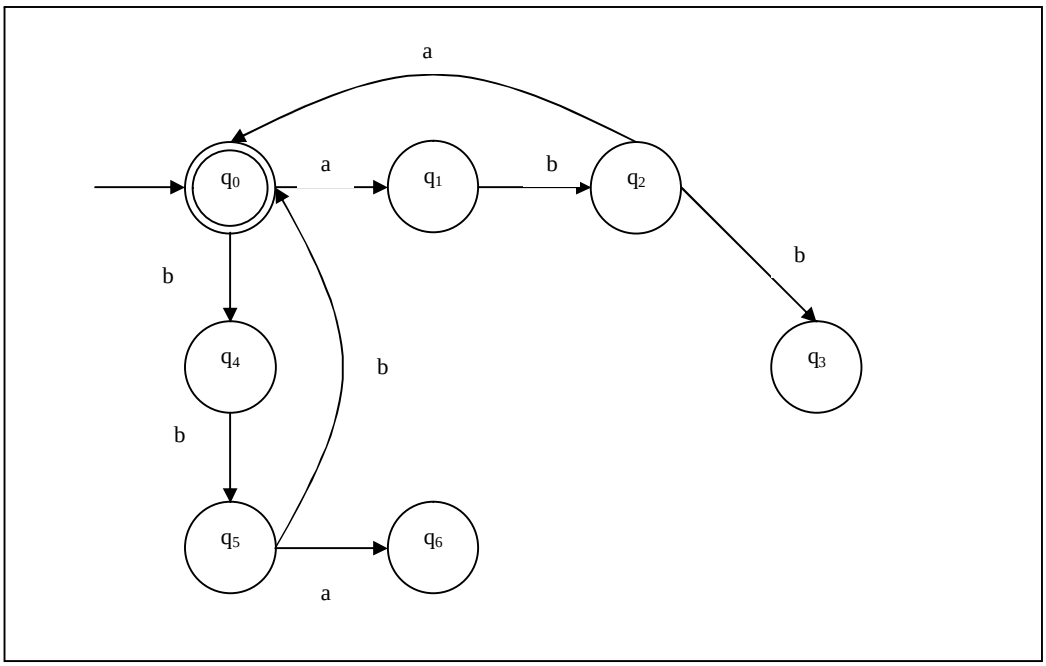
$V = \{ S,E,A,B\}$

$T = \{a,b\}$

$P = \{ S \rightarrow aE, E \rightarrow A \mid B, A \rightarrow aA \mid B, B \rightarrow bB \mid b \}$

$S = S$

Kita lihat contoh lain pada gambar 2



Gambar 2. Mesin FSA

Kita bisa mengkonstruksi aturan produksi untuk otomata tersebut:

$T = \{a,b\}$

$S = S$

Kumpulan aturan produksinya kita buat sebagai berikut:

$S \rightarrow aA \mid bB$

(identikkan S untuk q_0, A untuk q_1, B untuk q_4)

$A \rightarrow bC$

(identikkan C untuk q_2)

$C \rightarrow aS$

(dari q_2 mendapat input a ke q_0)

(dari q_3 tidak ada transisi keluar dan bukan *state* akhir maka transisi ke q_3 kita abaikan)

$B \rightarrow bD$

(identikkan D untk q_5)

$D \rightarrow bS$

(dari q_5 mendapat *input* b ke q_0)

(dari q_6 tidak ada transisi keluar dan bukan *state* akhir maka transisi ke q_6 kita abaikan)

Pada kasus ini kita lihat q_0 sebagai *state* akhir masih memiliki transisi keluar, maka untuk menandakannya sebagai *state* akhir kita buat:

$S \rightarrow \varepsilon$

*Bedakan dengan kasus gambar 1 dimana *state* akhir q_4 tidak memiliki transisi keluar.

Maka diperoleh:

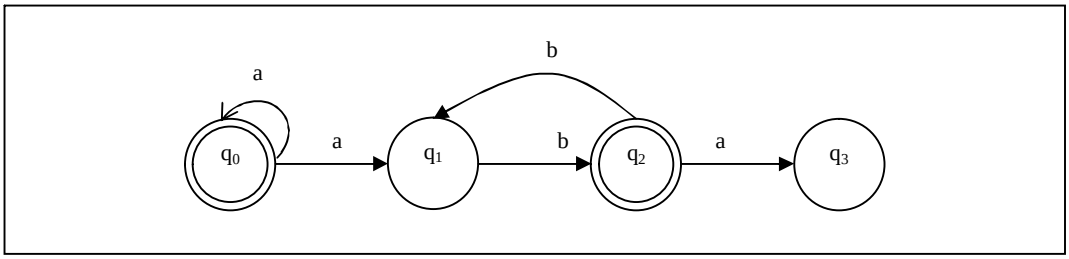
$V = \{ S,A,B,C,D\}$

$P = \{ S \rightarrow aA|bB|\varepsilon,\ A \rightarrow BC,\ B \rightarrow bD,\ C \rightarrow aS,\ D \rightarrow bS \}$

Sebenarnya aturan produksi diatas masih dapat disederhanakan menjadi:

$P = \{ S \rightarrow aA|bB|\varepsilon,\ A \rightarrow BaS,\ B \rightarrow bbS \}$

Sehingga mereduksi jumlah symbol variabel yang diperlukan:



Gambar 3. Mesin FSA

Contoh lain, dari gambar 3 kita bisa membuat aturan produksinya sebagai berikut:

$S \rightarrow aA \mid aS$

$A \rightarrow bB$

$B \rightarrow bA \mid \varepsilon,$

Finite State Automata untuk Suatu Tata Bahasa Regular.

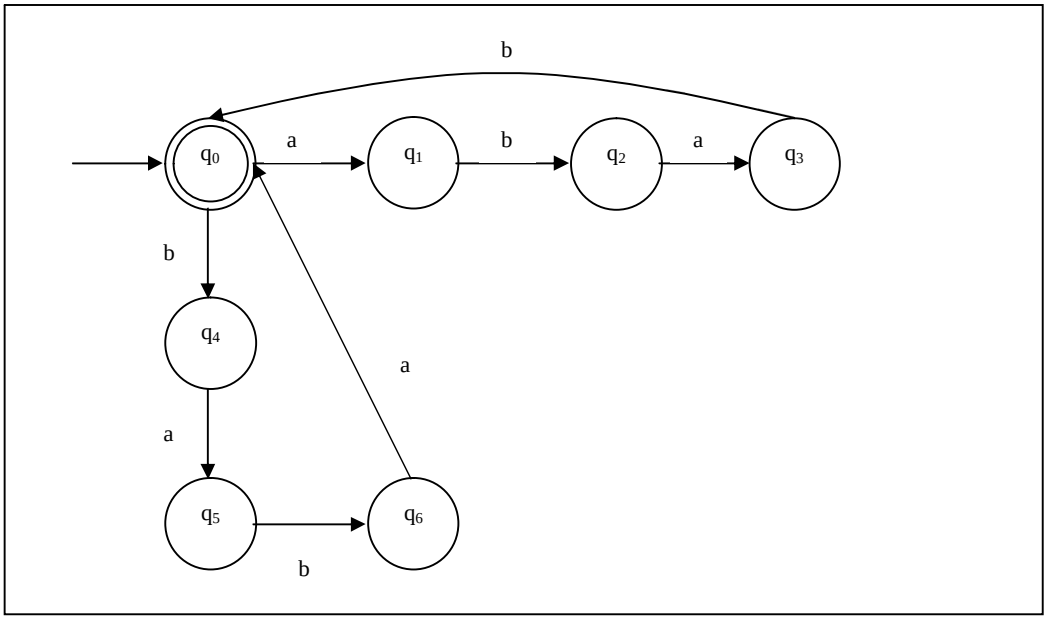
Bila sebelumnya dari suatu diagram transisi *finite state automata* kita bisa membuat aturan-aturan produksinya.

Misalkan terdapat tata bahasa regular dengan aturan produksi:

$$S \rightarrow aB \mid bA \mid \varepsilon$$

$$A \rightarrow abaS$$

$$B \rightarrow babS$$



Gambar 4. Finite Otomata dari suatu regular grammar

Bisa anda lihat hasilnya pada gambar 4. S akan berkorelasi dengan q_0, A dengan q_4, dan B dengan q_1. Sementara S → ε menandakan q_0 termasuk state akhir.

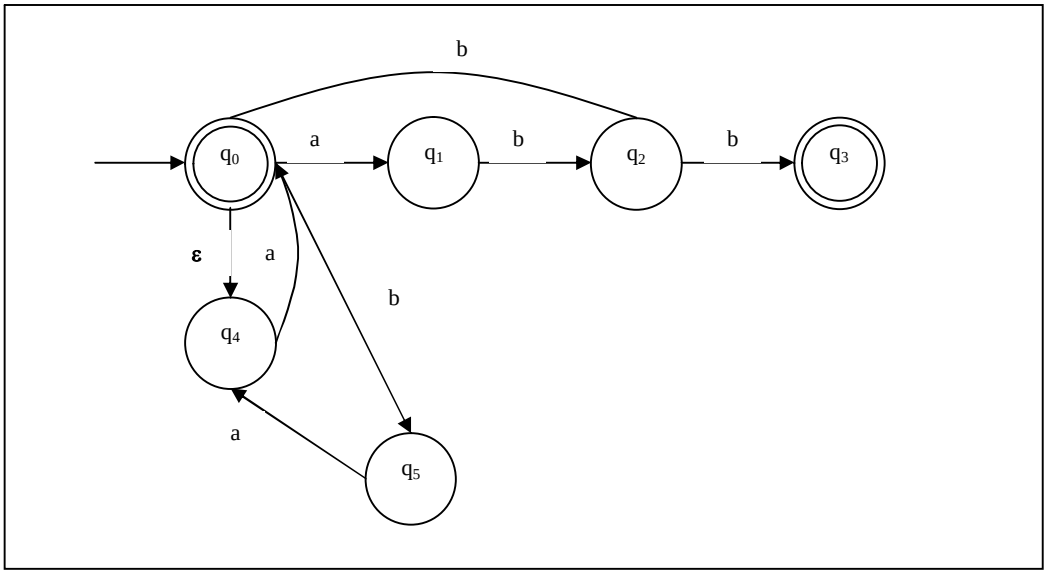
Contoh lain akan dibuat diagram transisi untuk tata bahasa regular:

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

$$A \rightarrow bS \mid b$$

$$B \rightarrow aS$$

Hasilnya bisa kita lihat pada gambar 5. Kita lihat S akan berkorelasi dengan q_0, A dengan q_2, B dengan q_4. Sementara S → ε menandakan q_o *state* akhir. Dari S→ B, maka kita bisa buat transisi ε dari q_0 ke q_4. Dari A→ kita tentukan state q_3 termasuk *state* akhir.



Gambar 5. *Finite otomata dari suatu regular grammar*

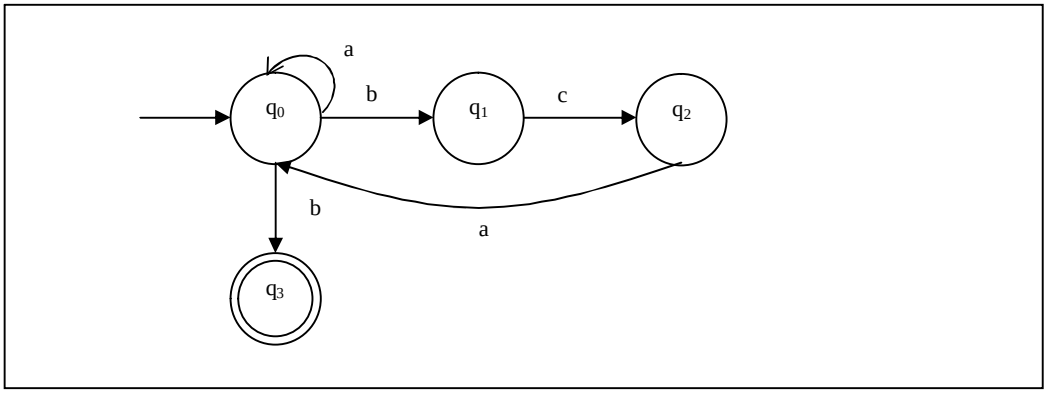
Contoh lain, akan dibuat diagram transisi untuk tata bahasa regular.

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

$$B \rightarrow cC$$

$$C \rightarrow aS$$

Bisa anda lihat hasilnya pada gambar 6 akan berkorelasi dengan q_0, B dengan q_1, C dengan q_2. Kita lihat S → b, maka kita buat state akhir q_3.



Gambar 6. Finite Otomata dari suatu regular grammar

FINITE STATE AUTOMATA
DENGAN OUTPUT

Mesin Moore

Suatu keterbatasan dari *finite state automata* yang sudah kita pelajari selama ini keputusannya terbatas pada diterima atau ditolak. otomata tersebut biasa disebut sebagai *accepter*, dalam hal ini *finite state accepter*. Kita bisa mengkonstruksi sebuah *finite state automata* yang memiliki keputusan beberapa keluaran/*output*, dalam hal ini otomata tersebut akan dikenal sebagai *transducer*. Pada mesin Moore, *output* akan berasosiasi dengan *state*. Mesin Moore didefinisikan dalam 6 (enam) tupel, M = (Q, Σ, δ, S, Δ, λ), dimana:

Q = himpunan state

Σ = himpunan symbol input

δ = fungsi transisi

S = state awal, $S \in Q$

Δ = himpunan output

λ = fungsi output untuk setiap state

*Perhatikan: komponen *state Final* dari *Deterministic Finite Automata* dihilangkan, karena disini keputusan dimunculkan sebagai *output*.

Kita lihat contoh penerapan dari Mesin Moore. Misal kita ingin memperoleh sisa pembagian (modulus) suatu bilangan dengan 3. Dimana *input* dinyatakan dalam biner. Mesin Moore yang bersesuaian bisa dilihat pada gambar 1. Konfigurasi mesin sebagai berikut:

$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{0,1\}$ (input dalam biner)

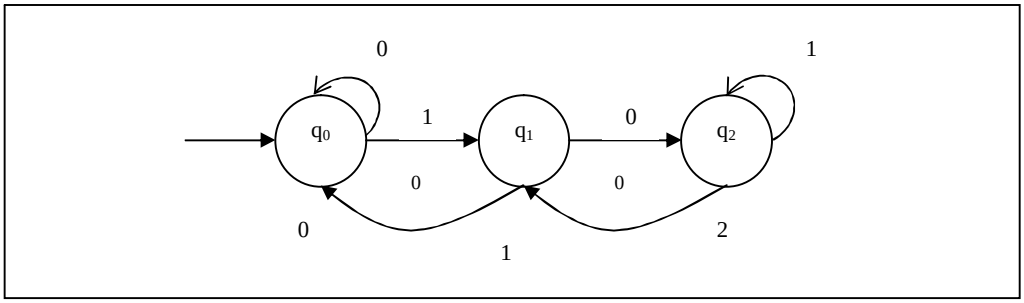
$\Delta = \{0,1,2\}$ (untuk *output*-nya pada kasus mod dengan 3 maka sisanya kemungkinan adalah (0,1,2)

$S = q_0$

$\lambda (q_0) = 0$

$\lambda (q_1) = 1$

$\lambda (q_2) = 2$



Gambar 1. Mesin Moore untuk modulus 3

Misalkan saja

- *5 mod 3* = ?

  input 5 dalam biner 101

  bila kita masukkan 101 ke dalam mesin, urutan *state* yang dicapai: q_0,q_1,q_2,q_3

  Perhatikan *state* terakhir yang dicapai adalah q_2, λ (q_2) =2, maka *5 mod 3* = 2
- *10 mod 3* =?

  input 10 dalam biner 1010

  bila kita masukkan 1010 ke dalam mesin, urutan *state* yang dicapai: q_0,q_1,q_2,q_2, q_1

  λ (q_1) =1, maka *10 mod 3* = 1

Mesin Mealy

Bila *output* pada mesin Moore berasosiasi dengan *state*, maka *output* pada Mesin Mealy akan berasosiasi dengan transisi. Mesin Mealy sendiri didefinisikan dalam 6 tupel, M = ($Q, \Sigma, \delta, S, \Delta, \lambda$), dimana:

Q = himpunan state

Σ = himpunan symbol input

δ = fungsi transisi

S = state awal, $S \in Q$

Δ = himpunan output

λ = fungsi output untuk setiap transisi

Contoh penerapan Mesin Mealy kita lihat pada gambar2.

Mesin itu akan mengeluarkan *output* apakah menerima (Y) atau menolak (T), suatu masukan. Dimana mesin akan mengeluarkan *output* ‘Y’ bila menerima untai yang memiliki akhiran 2 simbol berturutan yang sama, atau secara formal dalam ekspresi regular:

(0+1)*(00+11)

Contoh input yang diterima :

01011, 01100, 1010100, 10110100, 00, 11, 100, 011, 000, 111

Konfigurasi dari Mesin Mealy tersebut:

$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{0,1\}$

$\Delta = \{Y,T\}$

$S = q_0$

$\lambda (q_0,0) = T$

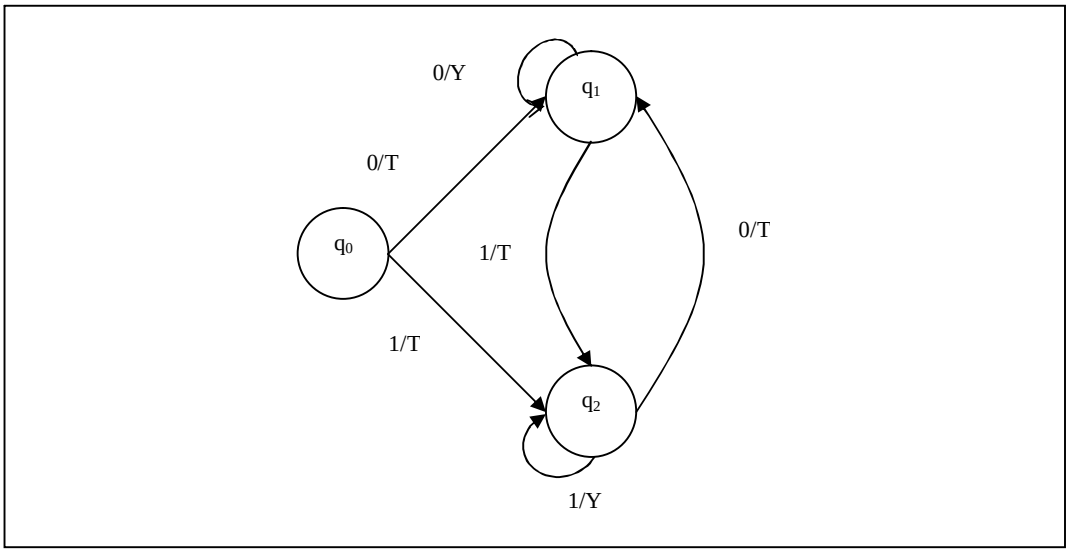
$\lambda (q_0,1) = T$

$\lambda (q_1,0) = Y$

$\lambda (q_1,1) = T$

$\lambda (q_2,0) = T$

$\lambda (q_2,1) = Y$



Gambar 2. Mesin Mealy

Ekivalensi Mesin Moore dan Mesin Mealy

Dari suatu Mesin Moore dapat dibuat Mesin Mealy yang ekivalen, begitu juga sebaliknya. Untuk mesin Mealy pada gambar 2 dapat kita buat Mesin Moore yang ekivalen yaitu gambar 3. Bisa kita lihat *state* pada mesin Moore dibentuk dari

kombinasi *state* pada Mealy dan banyaknya *output*. Karena jumlah *state* Mealy = 3, dan jumlah *output* = 2, maka jumlah *state* pada Moore yang ekivalen = 6.

Bisa dilihat konfigurasi Mesin Moore yang dibentuk:

$Q = \{q_0Y, q_0T, q_1Y, q_1T, q_2Y, q_2T\}$

$\Sigma = \{0,1\}$

$\Delta = \{Y,T\}$

$S = q_0$

$\lambda (q_0Y) = Y$

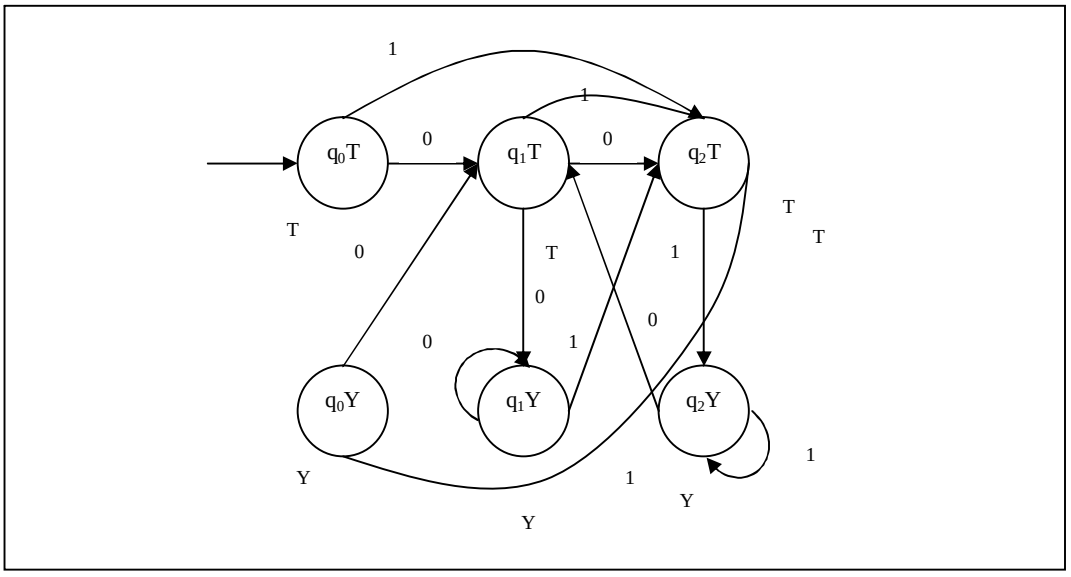
$\lambda (q_0T) = T$

$\lambda (q_1Y) = Y$

$\lambda (q_1T) = T$

$\lambda (q_2Y) = Y$

$\lambda (q_2T) = T$



Gambar 3. *Mesin Moore yang ekivalen dengan gambar 2*

Bila kita perhatikan dari gambar diatas, *state* q_0Y dapat dihapus karena tidak ada busur yang mengarah ke *state* tersebut.

Untuk memperoleh ekivalensi mesin Mealy dari suatu mesin Moore caranya lebih mudah, cukup dengan menambahkan label *output* ke setiap transisi dan menghapus label *output* pada setiap *state*. Kita lihat gambar 4 merupakan mesin Mealy yang ekivalen dengan mesin Moore pada gambar 1.

Konfigurasi Mesin Mealy tersebut:

$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{0,1\}$

$\Delta = \{0,1,2\}$

$S = q_0$

$\lambda\ (q_0,0) = 0$

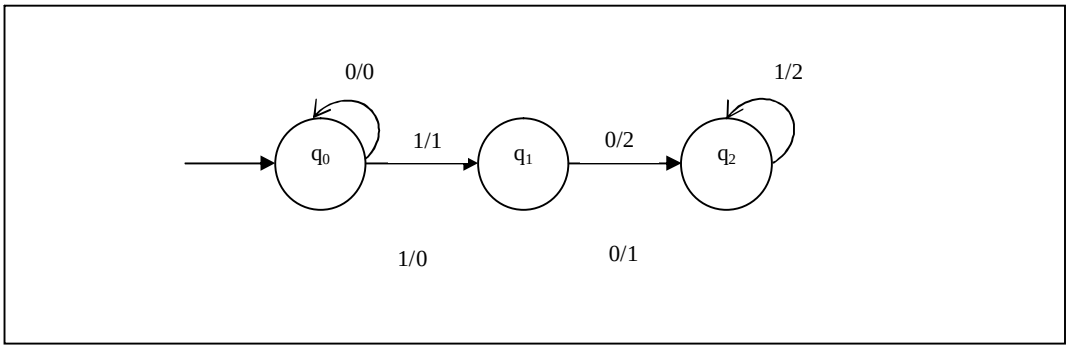
$\lambda\ (q_0,1) = 1$

$\lambda\ (q_1,0) = 2$

$\lambda\ (q_1,1) = 0$

$\lambda\ (q_2,0) = 1$

$\lambda\ (q_2,1) = 2$



Gambar 4. *Mesin Mealy yang ekivalen dengan gambar1.*