MODUL
TEORI BAHASA DAN AUTOMATA

DISUSUN OLEH :

Rizqia Cahyaning tyas

1997200314A

0315097901

SEKOLAH TINGGI TEKNIK PLN

TEKNIK INFORMATIKA

JAKARTA

2012

SATUAN ACARA PENGAJARAN (SAP)

MATA KULIAH/ SEMSTER : Otomata dan Teori Bahasa Formal Kode/ SKS : / 3

Pertemuan Ke	Pokok Bahasan dan TIU	Sub_Pokok Bahasan dan Sasaran Belajar	Cara Pengajaran	Media	Referensi
1	Pendahuluan TIU: Mahasiswa mengenal sejarah, definisi otomata dan contoh terapan teori otomata	- Sejarah Otomata Mahasiswa mengetahui sejarah teori otomata - Definisi otomata Mahasiswa mengetahui definisi otomata - Contoh terapan teori otomata Mahasiswa mengetahui contoh mesin yang termasuk ke dalam kategori Automata - Sifat-sifat Otomata Mahasiswa mengetahui sifat mesin automata	Ceramah	Papantulis & OHP	1,2,3
2	Bahasa dan Tata Bahasa Formal TIU: Mahasiswa memahami konsep dan istilah umum dalam teori bahasa .	- Konsep dasar bahasa formal Mahasiswa mengetahui konsep bahasa formal - Elemen Bahasa Formal Mahasiswa memahami empat unsur pembentuk tatabahasa	Ceramah	Papantulis & OHP	1,2,3
3	Bahasa dan Tata Bahasa Formal TIU: Mahasiswa memahami tipe-	- Klasifikasi Tatabahasa Formal menurut Chomsky Mahasiswa mengenal empat kelas tatabahasa dalam hirarki Chomsky, serta	Ceramah	Papantulis & OHP	1,2,3

	tipe bahasa dan menganalisa tipe-tipe bahasa	memahami hubungan antara keempat kelas tatabahasa tersebut			
4	Finite State Automata(FSA) TIU: Mahasiswa memahami mesin abstrak berupa model matematika dengan masukan dan keluaran diskrit yang dapat mengenali bahasa yang paling sederhana	- Model FSA Mahasiswa mengenal model matematika dari system FSA - Pendefinisian FSA Mahasiswa memahami elemen-elemen yang dimiliki oleh setiap FSA - FSA Deterministic Mahasiswa memahami bentuk formal DFA - FSA non_deterministik Mahasiswa memahami bentuk formal NFA - Fungsi Transisi yang diperluas Mahasiswa dapat menuliskan bentuk transisi yang diperluas	Ceramah	Papantulis & OHP	1,2,3
5	Finite State Automata(FSA) Lanjutan TIU: Mahasiswa memahami mesin abstrak berupa model matematika dengan masukan dan keluaran diskrit yang dapat mengenali bahasa yang paling sederhana	- Ekivalen DFA dan FSA Mahasiswa dapat membuat ekivalensi NFA dengan DFA	Ceramah	Papantulis & OHP	1,2,3
6	Finite State Automata(FSA) TIU: Mahasiswa memahami bentuk finite Automata yang	- Finite Automata dengan keluaran Mahasiswa mengenal finite state transducer berupa mesin Moore dan mesin Mealy, serta dapat membuat ekivalensi	Ceramah	Papantulis & OHP	1,2,3

	memiliki keluaran	antara kedua mesin tersebut			
7	Ekspresi Regular TIU: Mahasiswa memahami bentuk ekspresi himpunan string yang termasuk dalam bahasa reguler	- Definisi Ekspresi Reguler Mahasiswa memahami cara pendefinisian ekspresi regular dari sebuah bahasa - Aljabar Ekspresi Reguler Mahasiswa memahami hokum-hukum aljabar untuk ekspresi reguler	Ceramah	Papantulis & OHP	1,2,3
8	Ekspresi Reguler (lanjutan) TIU: Mahasiswa memahami hubungan ekspresi regular dengan finite Automata dan dapat membuat ekivalensinya	- Kaitan Ekspresi Reguler dan FSA Mahasiswa memahami keterkaitan ekspresi regular dan finite Automata - Dari Tatabahasa Reguler ke Finite Automata Mahasiswa dapat mengubah tatabahasa regular ke bentuk finite automata	Ceramah	Papantulis & OHP	1,2,3
9		UJIAN TENGAH SEMESTER			
10	Bahasa Bebas Konteks TIU: Mahasiswa memahami tatabahasa yang digunakan untuk mengenali bahasa bebas konteks (CFG)	- Definisi Mahasiswa memahami kelas bahasa berikutnya dari hirarki Chomsky Mahasiswa memahami aturan produksi dalam kelas CFG ini - Pengubahan Tata Bahasa Bebas Konteks Mahasiswa dapat mengubah CFG kedalam beberapa bentuk tanpa mengubah himpunan kalimat yang dihasilkan oleh tatabahasa tersebut	Ceramah	Papantulis & OHP	1,2,3
11	Bahasa Bebas Konteks TIU: Mahasiswa dapat mengubah suatu tata bahasa ke dalam	- Syarat pengubahan ke dalam CNF Mahasiswa dapat menentukan apakah suatu tatabahasa dapat diubah ke dalam CNF atau tidak	Ceramah	Papantulis & OHP	1,2,3

	bentuk baku Chomsky	- Langkah normalisasi Mahasiswa dapat mengubah suatu tata bahasa ke dalam CNF			
12	Pushdown Automata(PDA) TIU: Mahasiswa dapat merancang PDA dari suatu bahasa	- Pengenalan masukan oleh PDA Mahasiswa memahami perbedaan PDA status akhir dan PDA stack kosong - Definisi Formal PDA Mahasiswa memahami komponen-komponen PDA	Ceramah	Papantulis & OHP	1,2,3
13	Pushdown Automata TIU: Mahasiswa memahami ekivalensi PDA	- Kaitan antara CFG dengan PDA Mahasiswa dapat mengubah dari CFG ke PDA dan sebaliknya - Ekivalensi PDA Mahasiswa dapat mengubah PDA status akhir ke PDA stack kosong dan sebaliknya	Ceramah	Papantulis & OHP	1,2,3
14	Mesin Turing TIU: Mahasiswa mengenal model mesin turing	- Model Mesin Turing Mahasiswa mengenal model mesin turing dan komponen-komponen pembentuknya - Peranan Mesin Turing Mahasiswa mengetahui pemanfaatan mesin turing untuk mengenali himpunan string, menghitung fungsi integer, memodelkan kelas masalah dalam dunia komputasi. - Variasi-variasi Mesin Turing Mahasiswa mengetahui beberapa vaariasi dari mesin turing	Ceramah	Papantulis & OHP	1,2,3

Pustaka:

1. Munir, Rinaldi, Diktat Kuliah Matematika Informatika(Teori bahasa Formal dan Otomata), ITB
2. Hopcroft, John E.,Jefferey D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison Wesley Publishing Company, Massachusetts, 2001
3. Heriyanto, Bambang, Teori Bahasa, Otomata dan Komputasi serta terapannya, Informatika Bandung,2003

PERTEMUAN I

➢ Sejarah Otomata

Otomata bermula sebelum komputer ada pada teori di bidang sistem logika matematika atau formal, ilmuwan David Hilbert telah mencoba menciptakan algoritma umum untuk pembuktian (seluruh) persoalan matematika secara otomatis yaitu mampu menentukan salah benarnya sembarang prosisi matematika.

Tahun 1931, KurtGdel mempublikasikan teori ketidaklengkapan dimana membuktikan prosedur/algoritma yang dikehendaki David Hilbert tersebut tidak akan pernah ada. KurtGdel membangun rumus di kalkulus predikat yang diterapkan pada bilangan bulat yang memiliki pernyataan-pernyataan definisi yang tidak dapat dibuktikan maupun dibantah di dalam sistem logika yang mungkin dibangun manusia.

Formalisasi argumen teorema ketidaklengkapan KurtGdel ini berikut penjelasan dan formalisasi selanjutnya dari prosedur efektif secara intuisi merupakan salah satu pencapaian intelektual terbesar abad 20, yaitu abad dimana formalisasi berkembang semarak.

Pengembangan teori otomata, komputasi dan teori bahasa berikutnya difasilitasi perkembangan bidang psyco-linguistic. Bidang psyco-linguistic berupaya menjawab pertanyan-pertanyan berikut :

- Apakah bahasa secara umum?
- Bagaimana manusia mengembangkan bahasa?
- Bagaimana manusia memahami bahasa?
- Bagaimana manusia mengajarkan bahasa ke anak-anaknya?
- Apa gagasan-gagasan yang dapat dinyatakan dan bagaimana caranya?
- Bagaimana manusia membangun kalimat-kalimat dari gagasan-gagasan yang berada dipikirannya ?

Sekitar tahun 1950-an, Noam Chomsky menciptakan model matematika sebagai sarana untuk mendeskripsikan bahasa serta menjawab pertanyaan-pertanyaan di atas. Saat ini dimulai pendalaman bidang bahasa computer.

Sekitar tahun 1950-an, Noam Chomsky menciptakan model matematika sebagai sarana untuk mendeskripsikan bahasa serta menjawab pertanyaan-pertanyaan di atas. Saat ini dimulai pendalaman bidang bahasa komputer.

Perbedaan antara bahasa komputer dan bahasa manusia adalah sampai sekarang belum diketahuinya bagaimana cara manusia mengartikan bahasa, sementara dengan pasti dapat mengartikan bahasa pada komputer.

Noam Chomsky mengemukakan perangkat format disebut grammar untuk memodelkan properti-properti bahasa. Tata bahasa (grammer) bisa didefinisikan secara, formal sebagai kumpulan dari himpunan? himpunan variabel, simbol?simbol, terminal, simbol awal, yang dibatasi oleh aturan? aturan produksi.Tingkat bahasa dapat digolongkan menjadi empat yaitu :

1.Bahasa : Regular type 3

Mesin otomata : Finite State Otomata (FSA) meliputi deterministic finite automata dan non

deterministic finite automata
Batasan aturan produksi : adalah sebuah simbol variabel maksimal memiliki sebuah simbol variabel yang bila terletak di posisi paling kanan.

2.Bahasa : Bebas konteks/context free /type 2

Mesin otomata : Push down automata (PDA)
Batasan aturan produksi : Berupa sebuah simbol variabel.

3.Bahasa : Context sensitive/type 1

Mesin otomata : Linier bounded automata
Batasan aturan produksi :

4.Bahasa : Unrestricted /phase /natural language/type 0

Mesin otomata : Mesin turing
Batasan aturan produksi : Tidak ada batasan

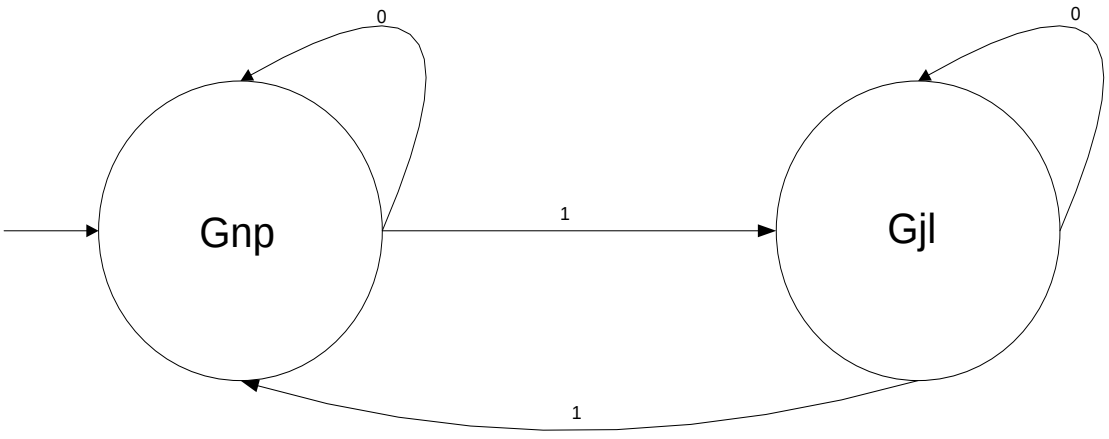
➢ Definisi Otomata

Teori Bahasa

- Teori bahasa membicarakan bahasa formal (*formal language*), terutama untuk kepentingan perancangan kompilator (*compiler*) dan pemroses naskah (*text processor*).
- Bahasa formal adalah kumpulan *kalimat*. Semua kalimat dalam sebuah bahasa dibangkitkan oleh sebuah tata bahasa (*grammar*) yang sama.
- Sebuah bahasa formal bisa dibangkitkan oleh dua atau lebih tata bahasa berbeda.
- Dikatakan bahasa formal karena grammar diciptakan mendahului pembangkitan setiap kalimatnya.
- Bahasa Natural/manusia bersifat sebaliknya; grammar diciptakan untuk meresmikan kata-kata yang hidup di masyarakat. Dalam pembicaraan selanjutnya 'bahasa formal' akan disebut 'bahasa' saja.

Otomata (*Automata*)

- Otomata adalah mesin abstrak yang dapat mengenali (*recognize*), menerima (*accept*), atau membangkitkan (*generate*) sebuah kalimat dalam bahasa tertentu.



➢ BEBERAPA PENGERTIAN DASAR

· Simbol adalah sebuah entitas abstrak (seperti halnya pengertian *titik* dalam geometri). Sebuah huruf atau sebuah angka adalah contoh simbol.
· String adalah deretan terbatas (*finite*) simbol-simbol. Sebagai contoh, jika *a*, *b*, dan *c* adalah tiga buah simbol maka *abcb* adalah sebuah string yang dibangun dari ketiga simbol tersebut.
· Jika *w* adalah sebuah string maka panjang string dinyatakan sebagai |*w*| dan didefinisikan sebagai cacahan (banyaknya) simbol yang menyusun string tersebut. Sebagai contoh, jika *w* = *abcb* maka |*w*|= 4.
· String hampa adalah sebuah string dengan nol buah simbol. String hampa dinyatakan dengan simbol ε (atau ^) sehingga |ε|= 0. String hampa dapat dipandang sebagai simbol hampa karena keduanya tersusun dari nol buah simbol.
· Alfabet adalah hinpunan hingga (*finite set*) simbol-simbol

➢ OPERASI DASAR STRING

Diberikan dua string : *x* = *abc*, dan *y* = *123*

· **Prefik** string *w* adalah string yang dihasilkan dari string *w* dengan menghilangkan *nol* atau lebih simbol-simbol paling belakang dari string *w* tersebut.
Contoh : *abc*, *ab*, *a*, dan ε adalah semua Prefix(*x*)
· **ProperPrefix** string *w* adalah string yang dihasilkan dari string *w* dengan menghilangkan *satu* atau lebih simbol-simbol paling belakang dari string *w* tersebut.
Contoh : *ab*, *a*, dan ε adalah semua ProperPrefix(*x*)
· **Postfix** (atau Sufix) string *w* adalah string yang dihasilkan dari string *w* dengan menghilangkan *nol* atau lebih simbol-simbol paling depan dari string *w* tersebut.
Contoh : *abc*, *bc*, *c*, dan ε adalah semua Postfix(*x*)
· **ProperPostfix** (atau PoperSufix) string *w* adalah string yang dihasilkan dari string *w* dengan menghilangkan *satu* atau lebih simbol-simbol paling depan dari string *w* tersebut.
Contoh : *bc*, *c*, dan ε adalah semua ProperPostfix(*x*)
· **Head string** *w* adalah simbol paling depan dari string *w*.
Contoh : *a* adalah Head(*x*)
· **Tail** string *w* adalah string yang dihasilkan dari string *w* dengan menghilangkan simbol paling depan dari string *w* tersebut.
Contoh : *bc* adalah Tail(*x*)
w adalah string yang dihasilkan dari string *w* dengan menghilangkan *nol* atau lebih simbol-simbol paling depan dan/atau simbol-simbol paling belakang dari string *w* tersebut.
Contoh : *abc*, *ab*, *bc*, *a*, *b*, *c*, dan ε adalah semua Substring(*x*)
· **ProperSubstring** string *w* adalah string yang dihasilkan dari string *w* dengan menghilangkan *satu* atau lebih simbol-simbol paling depan dan/atau simbolsimbol paling belakang dari string *w* tersebut.
Contoh : *ab*, *bc*, *a*, *b*, *c*, dan ε adalah semua Substring(*x*)
· **Subsequence** string *w* adalah string yang dihasilkan dari string *w* dengan menghilangkan *nol* atau lebih simbol-simbol dari string *w* tersebut.

Contoh : *abc*, *ab*, *bc*, *ac*, *a*, *b*, *c*, dan ε adalah semua Subsequence(*x*)

· **ProperSubsequence** string *w* adalah string yang dihasilkan dari string *w* dengan menghilangkan *satu* atau lebih simbol-simbol dari string *w* tersebut.

Contoh : *ab*, *bc*, *ac*, *a*, *b*, *c*, dan ε adalah semua Subsequence(*x*)

· **Concatenation** adalah penyambungan dua buah string. Operator concatenation adalah *concate* atau tanpa lambang apapun.

Contoh : concate(*xy*) = *xy* = *abc123*

· **Alternation** adalah pilihan satu di antara dua buah string. Operator alternation adalah *alternate* atau | |.

Contoh : alternate(*xy*) = *x*|*y* = *abc* atau *123*

· **Kleene** Closure : *x** = ε|*x*|*xx*|*xxx*|… = ε|*x*|*x* 2 |*x* 3 |…

Positive Closure : *x* + = *x*|*xx*|*xxx*|… = *x*|*x* 2 |*x* 3 |…

➢ SIFAT OPERASI DASAR STRING

· Tidak selalu berlaku : *x* = Prefix(*x*)Postfix(*x*)

· Selalu berlaku : *x* = Head(*x*)Tail(*x*)

· Tidak selalu berlaku : Prefix(*x*) = Postfix(*x*) atau Prefix(*x*) ≠ Postfix(*x*)

· Selalu berlaku : ProperPrefix(*x*) ≠ ProperPostfix(*x*)

· Selalu berlaku : Head(*x*) ≠ Tail(*x*)

· Setiap Prefix(*x*), ProperPrefix(*x*), Postfix(*x*), ProperPostfix(*x*), Head(*x*), dan Tail(*x*) adalah Substring(*x*), tetapi tidak sebaliknya

· Setiap Substring(*x*) adalah Subsequence(*x*), tetapi tidak sebaliknya

· Dua sifat aljabar concatenation :

♦ Operasi concatenation bersifat asosiatif : *x*(*yz*) = (*xy*)*z*

♦ Elemen identitas operasi concatenation adalah ε : ε*x* = *x*ε = *x*

·Tiga sifat aljabar alternation :

♦ Operasi alternation bersifat komutatif : *x*|*y* = *y*|*x*

♦ Operasi alternation bersifat asosiatif : *x*|(*y*|*z*) = (*x*|*y*)|*z*

♦ Elemen identitas operasi alternation adalah dirinya sendiri : *x*|*x* = *x*

· Sifat distributif concatenation terhadap alternation : *x* (*y*|*z*) = *xy*|*xz*

· Beberapa kesamaan :

♦ Kesamaan ke-1 : $(x^*)^* = (x^*)$

♦ Kesamaan ke-2 : $\varepsilon | x+ = x+ | \varepsilon = x^*$

♦ Kesamaan ke-3 : $(x|y)^* = \varepsilon|x|y|xx|yy|xy|yx|\ldots$ = semua string yang

merupakan concatenation dari nol atau lebih *x*, *y*, atau keduanya.

➢ Contoh Terapan Teori Otomata

Contoh Penerapan Teori Bahasa Otomata

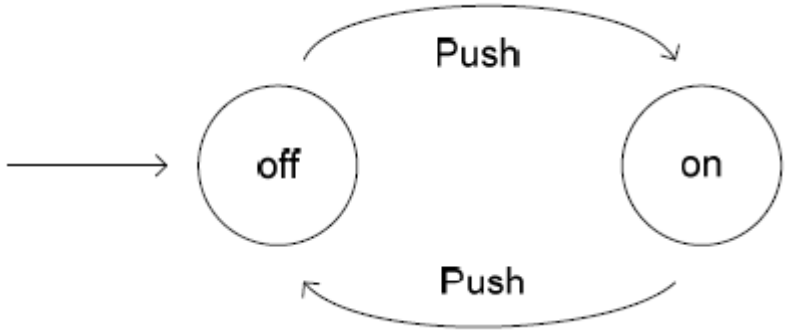
Model switch on/off digambarkan sebagai berikut:

Contoh 1:

Model tersebut mengingat apakah *switch* berada dalam state "*on*" atau state "*off*". Model memungkinkan user untuk menekan tombol yang memiliki pengaruh berbeda tergantung pada keadaan *switch*:

- *switch* berada dalam *state* "off" maka setelah tombol ditekan *state* berubah menjadi "on".
- Jika *switch* berada dalam *state* "on" maka setelah tombol ditekan *state* berubah menjadi "off".

Model pada Gambar 1 dapat dipandang sebagai model *finite automato* sederhana.



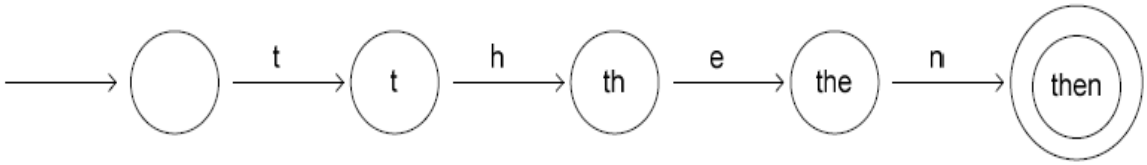
Dalam *finite automata*, ***state*** dinyatakan oleh lingkaran, dan dalam Contoh 1 *state* diberi nama "on" dan "off". ***Arc*** diantara *state* diberi label "input " yang menyatakan pengaruh eksternal pada sistem. Dalam Contoh 1 kedua *arc* diberilabel 'push" yang menyatakan *user* menekan tombol tertentu.

Salah satu *state* dinyatakan sebagai ***start state*** atau ***initial state*** yang merupakan *state* dimana sistem berada dalam keadaan awal. Dalam Contoh *start state* adalah off. Dalam pembahasan selanjutnya, *start state* ditunjukan oleh kata *start* dan panah menuju *start state* tersebut. Dalam Gambar 1 *state* on dinyatakan sebagai ***final*** atau ***accepting*** state.

Dalam *state* tersebut, peralatan yang sedang dikontrol oleh *switch* akan beroperasi. Dalam pembahasan selanjutnya, *final State* dinyatakan dalam lingkaran ganda.

Contoh 2:

Finite automaton berikut dapat dinyatakan sebagai bagian dari *lexical analyzer*.



Tugas dari *automaton* tersebut adalah mengenali *keyword* "then" sehingga diperlukan lima *state* masing-masing menyatakan posisi yang berbeda dalam kata "then" yang telah dicapai sejauh ini. Posisi ini berhubungan dengan prefix dari kata yang berkisar dari kata string kosong (tidak ada kata yang dikenali sejauh ini) sampai dengan kata lengkap. Dalam Gambar 2, input dinyatakan oleh huruf. *Start state* merupakan string kosong, dan setiap *state* memiliki transisi pada huruf selanjutnya dari kata then ke *state* yang menyatakan prefix selanjutnya yang lebih besar. *State* yang diberi nama "then" dimasuki ketika input mengeja kata "then". Karena fungsi dari model dalam Gambar 2 adalah mengenali kata then, maka *state* "then"dinyatakan sebagai *accepting state*.

➢ **Sifat-sifat Otomata**

Automata adalah suatu mesin sekuensial (otomatis), yang menerima input
(dari pita masukan) dan mengeluarkan output, keduanya dalam bentuk diskrit. Automata mempunyai sifat-sifat

• Kelakuan mesin bergantung pada rangkaian masukan yang diterima mesin tersebut.
• Setiap saat, mesin dapat berada pada satu status tertentu dan dapat berpindah ke status baru karena adanya perubahan input.
• Rangkaian input (diskrit) pada mesin automata dapat dianggap sebagai bahasa
yang harus "dikenali" oleh sebuah automata. Setelah pembacaan input selesai, mesin automata kemudian membuat "keputusan".
Jenis- jenis automata :

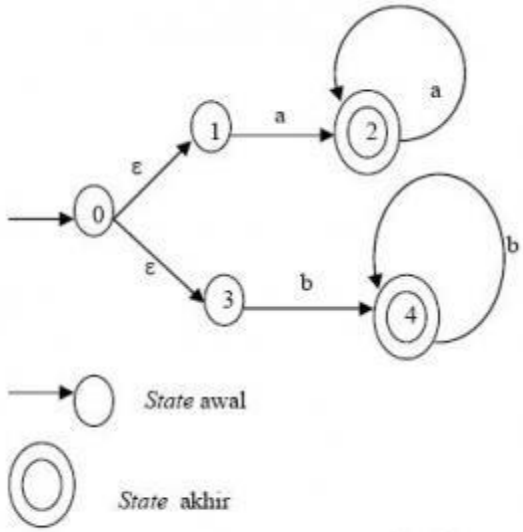
Jenis	Pita masukan	Arah Head	Memori
Finite State	Read Only	1 arah	-
Push Down	Read Only	1 arah	stack
Linear-Bounded	R/W	2 arah	(bounded)
Turing Machine	R/W	2 arah	(unbounded)

Pada bahasan ini jenis automata yang akan dipakai adalah Finite State Automata (FSA). FSA adalah mesin yang dapat mengenali kelas bahasa reguler dan memiliki sifat-sifat :

1. Pita masukan (input tape) berisi rangkaian simbol (string) yang berasal dari himpunan simbol / alfabet.
2. Setiap kali setelah membaca satu karakter, posisi read head akan berada pada symbol berikutnya.
3. Setiap saat, FSA berada pada status tertentu
4. Banyaknya status yang berlaku bagi FSA adalah berhingga.

Suatu FSA didefenisikan sebagai F = (Q, S, q0, d, F) dengan
Q = himpunan state(keadaan)
∑ = himpunan input
q0 e Q adalah keadaan awal
&= Q x S .. Q adalah tabel transisi
F = keadaan akhir

Suatu NFA dapat direpresentasikan dalam bentuk bagan sebagai suatu graf yang diberi label dan disebut dengan graf transisi. Dalam graf transisi ini nodal adalah state dan label dari sisi menyatakan fungsi transisi, contoh Graf transisi NFA dapat dilihat pada gambar1.



Gambar 1. *NFA* penerima $aa^*|bb^*$

Gambar 1. diatas mempunyai defenisis formal sebagai berikut :
Q = {0, 1, 2, 3, 4}
∑ = {a,b}
q0 = 0
F = {2, 4}
&= diagram transisi dapat dilihat pada tabel 1

Tabel 1. Diagram Transisi

δ	a	b
0	∅	∅
1	2	∅
2	2	∅
3	∅	4
4	∅	4

➢ **Konsep Dasar Bahasa Formal**

- Teori bahasa membicarakan bahasa formal (*formal language*), terutama untuk kepentingan perancangan kompilator (*compiler*) dan pemroses naskah (*text processor*).
- Bahasa formal adalah kumpulan *kalimat*. Semua kalimat dalam sebuah bahasa dibangkitkan oleh sebuah tata bahasa (*grammar*) yang sama.
- Sebuah bahasa formal bisa dibangkitkan oleh dua atau lebih tata bahasa berbeda.
- Dikatakan bahasa formal karena grammar diciptakan mendahului pembangkitan setiap kalimatnya.
- Bahasa Natural/manusia bersifat sebaliknya; grammar diciptakan untuk meresmikan kata-kata yang hidup di masyarakat. Dalam pembicaraan selanjutnya 'bahasa formal' akan disebut 'bahasa' saja.

Konsep Dasar

• Anggota alfabet dinamakan simbol terminal.

• Kalimat adalah deretan hingga simbol-simbol terminal.

• Bahasa adalah himpunan kalimat-kalimat. Anggota bahasa bisa tak hingga kalimat.

• Simbol-simbol berikut adalah simbol terminal :

- huruf kecil, misalnya : a, b, c
- , dan *–simbol operator, misalnya : +,
- simbol tanda baca, misalnya : (,), dan ;
- simbol tanda baca, misalnya : (,), dan ;
- string yang tercetak tebal, misalnya : **if**, **then**, dan **else**.

• Simbol-simbol berikut adalah simbol non terminal /Variabel :

- huruf besar, misalnya : A, B, C
- huruf S sebagai simbol awal
- string yang tercetak miring, misalnya : *expr*

• Huruf yunani melambangkan string yang tersusun atas simbol-simbol terminal atau simbol-simbol non terminal atau campuran keduanya, misalnya : α,β, dan ε

• Sebuah produksi dilambangkan sebagai α --> β, artinya : dalam sebuah derivasi dapat dilakukan penggantian simbol α dengan simbol β.

• Derivasi adalah proses pembentukan sebuah kalimat atau sentensial. Sebuah derivasi dilambangkan sebagai : α ==> β.

• Sentensial adalah string yang tersusun atas simbol-simbol terminal atau simbol-simbol non terminal atau campuran keduanya.

• Kalimat adalah string yang tersusun atas simbol-simbol terminal. Kalimat adalah merupakan sentensial, sebaliknya belum tentu.

➢ **Elemen Bahasa Formal**

Elemen-elemen Bahasa adalah Alphabet,Grammar (Tata Bahasa) dan Semantic.

- **Alphabet** adalah himpunan terhingga dari token-token dimana kalimat dibentuk dalam suatu bahasa. **Alpabet** Adalah himpunan simbol (karakter) tak kosong yang berhingga. Alpabet digunakan untuk membentuk kata-kata (string-string) di bahasa. Bahasa dimulai dengan alpabet. Pada beberapa buku, alpabet dilambangkan dengan Σ Istilah huruf, karakter dan simbol adalah *sinonim* menunjukkan elemen alpabet. Jika simbol berbaris bersebelahan, maka diperoleh "string simbol". Istilah kalimat, kata dan string adalah *sinonim*

  Contoh :
  {a,b} -> Himpunan yang terdiri dari simbol "a" dan "b".

- **Grammar** (Tata Bahsa) adalah himpunan dari aturan-aturan structural yang didefinisikan yang berlaku dalam suatu kalimat pada token-token.

  Grammar G didefinisikan sebagai pasangan 4 tuple : V_t , V_n , S, dan P, dan dituliskan sebagai G(V_t , V_n , S, P), dimana :

  - V_t : himpunan simbol-simbol terminal (alfabet) = kamus
  - V_n : himpunan simbol-simbol non terminal
  - S C V : simbol awal (atau simbol start)
  - P : himpunan produksi

  Contoh :

- G1 : VT = {I, want, need, You}, V = {S,A,B,C}, P = {S --> ABC, A--> I, B--> want | need, C--> You}

S --> ABC

→ IwantYou

→ L(G1)={IwantYou,IneedYou}

2. G2 : VT = {a}, V = {S}, P = {S ® aS | a}

S --> aS

--> aaS

--> aaa L(G2) ={an --> n ≥ 1}

L(G2)={a, aa, aaa, aaaa, …}

- **Semantic** adalah himpunan aturan-aturan yang didefinisikan yang mempunyai efek operasional pada setiap program yang ditulis dalam bahasa apabila ditramslasi dan dieksekusi pada suatu mesin.

Kalimat dalam bahasa inggris dikonstruksi dari himpunan karakter yang terdiri dari huruf, angka, spasi dan tanda-tanda baca. Karakter-karakter ini dibentuk menjadi kata melalui aturan-aturan ejaan dan kamus, dan kemudian kata-kata dibentuk menjadi kalimat bedasarkan aturan-aturan tata bahasa.

Suatu program komputer dapat dikonstruksi dengan cara yang sama dari urutan karakter yang terdapat pada himpunan karakter dari komputer tersebut. Perbedaan yang prinsip antara Bahasa inggris dengan Bahasa pemrogramanan komputer adalah bahwa aturan-aturan ejaan dan tata bahasa dalam bahasa inggris sangat kompleks dan banyak pengecualian dan keragu-raguan, sementara dalam bahasa pemrograman harus mempunyai struktur yang tepat dan pasti.

PERTEMUAN II

➢ **Klasifikasi tata bahasa formal menurut chomsky**

Klasifikasi Chomsky

Menurut Noam Chomsky secara umum tata bahasa dikelompokan menjadi empat kelas. Kelas yang paling umum adalah tata bahasa tak beraturan (*unrestricted grammars*), bukan kelas prasa struktur (non phrase-structured). Tiga kelas yang lain adalah prasa struktur (*phase-structured*) yaitu: *contex-sensitive, contex-free*, dan regular.

Masing-masing tata bahasa mempunyai pengenal yaitu:

1. Regular Grammar $\Rightarrow$ Finite State Automata(FSA)
2. Contex-free Grammar $\Rightarrow$ Push-Down Automata(PDA)
3. Contex-Sensitive Grammar $\Rightarrow$ Linear Bounded Automata (LBA)
4. Unrestricted Grammar $\Rightarrow$ Turing Machine (TM)

Berdasarkan komposisi bentuk ruas kiri dan ruas kanan produksinya ($\alpha \rightarrow \beta$), Noam Chomsky mengklasifikasikan 4 tipe grammar :

1. Grammar tipe ke-0 : Unrestricted Grammar (UG)
   Ciri : $\alpha, \beta \in (V_T \mid V_N)^*$, $\mid \alpha \mid > 0$
2. Grammar tipe ke-1 : Context Sensitive Grammar (CSG)
   Ciri : $\alpha, \beta \in (V_T \mid V_N)^*$, $0 < \mid \alpha \mid \leq \mid \beta \mid$
3. Grammar tipe ke-2 : Context Free Grammar (CFG)
   Ciri : $\alpha \in V_N$, $\beta \in (V_T \mid V_N)^*$
4. Grammar tipe ke-3 : Regular Grammar (RG)
   Ciri : $\alpha \in V_N$, $\beta \in \{V_T, V_T V_N\}$ atau $\alpha \in V_N$, $\beta \in \{V_T, V_N V_T\}$

Tipe sebuah grammar (atau bahasa) ditentukan dengan aturan sebagai berikut :

> A language is said to be type-i (i = 0, 1, 2, 3) language if it can be specified by a type-i grammar but can't be specified any type-(i+1) grammar.

Contoh Analisa Penentuan Type Grammar

1. Grammar G_1 dengan P_1 = {S $\rightarrow$ aB, B $\rightarrow$ bB, B $\rightarrow$ b}.

Ruas kiri semua produksinya terdiri dari sebuah V_N maka G_1 kemungkinan tipe CFG atau RG. Selanjutnya karena semua ruas kanannya terdiri dari sebuah V_T atau string $V_T V_N$ maka G_1 adalah RG(3).

2. Grammar G_2 dengan P_2 = {S $\rightarrow$ Ba, B $\rightarrow$ Bb, B $\rightarrow$ b}.

Ruas kiri semua produksinya terdiri dari sebuah V_N maka G_2 kemungkinan tipe CFG atau RG. Selanjutnya karena semua ruas kanannya terdiri dari sebuah V_T atau string $V_N V_T$ maka G_2 adalah RG(3).

3. Grammar G_3 dengan P_3 = {S $\rightarrow$ Ba, B $\rightarrow$ bB, B $\rightarrow$ b}.

Ruas kiri semua produksinya terdiri dari sebuah V_N maka G_3 kemungkinan tipe CFG atau RG. Selanjutnya karena ruas kanannya mengandung string $V_T V_N$ (yaitu bB) dan juga string $V_N V_T$ (Ba) maka G_3 bukan RG, dengan kata lain G_3 adalah CFG(2).

4. Grammar G_4 dengan P_4 = {S $\rightarrow$ aAb, B $\rightarrow$ aB}.

Ruas kiri semua produksinya terdiri dari sebuah V_N maka G_4 kemungkinan tipe CFG atau RG. Selanjutnya karena ruas kanannya mengandung string yang panjangnya lebih dari 2 (yaitu aAb) maka G_4 bukan RG, dengan kata lain G_4 adalah CFG.

5. Grammar G_5 dengan P_5 = {S $\rightarrow$ aA, S $\rightarrow$ aB, aAb $\rightarrow$ aBCb}.

Ruas kirinya mengandung string yang panjangnya lebih dari 1 (yaitu aAb) maka G_5 kemungkinan tipe CSG atau UG. Selanjutnya karena semua ruas kirinya lebih pendek atau sama dengan ruas kananya maka G_5 adalah CSG.

6. Grammar G_6 dengan P_6 = {aS $\rightarrow$ ab, SAc $\rightarrow$ bc}.

Ruas kirinya mengandung string yang panjangnya lebih dari 1 maka G_6 kemungkinan tipe CSG atau UG. Selanjutnya karena terdapat ruas kirinya yang lebih panjang daripada ruas kananya (yaitu SAc) maka G_6 adalah UG.

Derivasi Kalimat dan Penentuan Bahasa

Tentukan bahasa dari masing-masing gramar berikut :

1. G_1 dengan P_1 = {1. S $\rightarrow$ aAa, 2. A $\rightarrow$ aAa, 3. A $\rightarrow$ b}.

Jawab :

Derivasi kalimat terpendek :

$S \Rightarrow aAa$ (1)

$\Rightarrow aba$ (3)

Derivasi kalimat umum :

$S \Rightarrow aAa$ (1)

$\Rightarrow aaAaa$ (2)

…

$\Rightarrow a^n Aa^n$ (2)

$\Rightarrow a^n ba^n$ (3)

Dari pola kedua kalimat disimpulkan : $L_1(G_1) = \{ a^n ba^n \mid n \geq 1\}$

2. G_2 dengan

$P_2 = \{1.\ S \rightarrow aS,\ 2.\ S \rightarrow aB,\ 3.\ B \rightarrow bC,\ 4.\ C \rightarrow aC,\ 5.\ C \rightarrow a\}$.

Jawab :

Derivasi kalimat terpendek :

$S \Rightarrow aB$ (2)

$\Rightarrow abC$ (3)

$\Rightarrow aba$ (5)

Derivasi kalimat umum :

$S \Rightarrow aS$ (1)

…

$\Rightarrow a^{n-1}S$ (1)

$\Rightarrow a^n B$ (2)

$\Rightarrow a^n bC$ (3)

$\Rightarrow a^n baC$ (4)

…

$\Rightarrow a^n ba^{m-1}C$ (4)

$\Rightarrow a^n ba^m$ (5)

Dari pola kedua kalimat disimpulkan : $L_2(G_2)=\{a^n ba^m \mid n \geq 1, m \geq 1\}$

3. G_3 dengan

$P_3 = \{1.\ S \rightarrow aSBC,\ 2.\ S \rightarrow abC,\ 3.\ bB \rightarrow bb,$

4. $bC \rightarrow bc$, 5. $CB \rightarrow BC$, 6. $cC \rightarrow cc$}.

Jawab :

Derivasi kalimat terpendek 1:

$S \Rightarrow abC$ (2)

$\Rightarrow abc$ (4)

Derivasi kalimat terpendek 2 :

$S \Rightarrow aSBC$ (1)

$\Rightarrow aabCBC$ (2)

$\Rightarrow aabBCC$ (5) aabcBC (4)

$\Rightarrow aabbCC$ (3)

$\Rightarrow aabbcC$ (4)

$\Rightarrow aabbcc$ (6)

Derivasi kalimat terpendek 3 :

$S \Rightarrow aSBC$ (1)

$\Rightarrow aaSBCBC$ (1)

$\Rightarrow aaabCBCBC$ (2)

$\Rightarrow aaabBCCBC$ (5)

$\Rightarrow aaabBCBCC$ (5)

$\Rightarrow aaabBBCCC$ (5)

$\Rightarrow aaabbBCCC$ (3)

$\Rightarrow aaabbbCCC$ (3)

$\Rightarrow aaabbbcCC$ (4)

$\Rightarrow aaabbbccC$ (6)

$\Rightarrow aaabbbccc$ (6)

Dari pola ketiga kalimat disimpulkan : $L_3\ (G_3) = \{ a^n b^n c^n \mid n \geq 1\}$

Menentukan Grammar Sebuah Bahasa

1. Tentukan sebuah gramar regular untuk bahasa $L_1 = \{ a^n \mid n \geq 1\}$

   Jawab :

   $P_1 (L_1) = \{S \rightarrow aS \mid a\}$

2. Tentukan sebuah gramar bebas konteks untuk bahasa :

L_2 : himpunan bilangan bulat non negatif ganjil

Jawab :

Langkah kunci : digit terakhir bilangan harus ganjil.

Vt={0,1,2,..9}

Vn ={S, G,J}

P={S→HT|JT|J; T→GT|JT|J; H→2|4|6|8; G→0|2|4|6|8;J→1|3|5|7|9}

P={S→GS|JS|J; G→0|2|4|6|8;J→1|3|5|7|9}

Buat dua buah himpunan bilangan terpisah : genap (G) dan ganjil (J)

$P_2(L_2) = \{S \rightarrow J \mid GS \mid JS,\ G \rightarrow 0 \mid 2 \mid 4 \mid 6 \mid 8,\ J \rightarrow 1 \mid 3 \mid 5 \mid 7 \mid 9\}$

3. Tentukan sebuah gramar bebas konteks untuk bahasa :

   L_3 = himpunan semua identifier yang sah menurut bahasa pemrograman Pascal dengan batasan : terdiri dari simbol huruf kecil dan angka, panjang identifier boleh lebih dari 8 karakter

   Jawab :

   Langkah kunci : karakter pertama identifier harus huruf.

   Buat dua himpunan bilangan terpisah : huruf (H) dan angka (A)

   S→HT|H;T→HT|AT|H|A; H→a|..|z; A→0|..|9

   $P_3(L_3) = \{S \rightarrow H \mid HT,\ T \rightarrow AT \mid HT \mid H \mid A,$

   $H \rightarrow a \mid b \mid c \mid \ldots,\ A \rightarrow 0 \mid 1 \mid 2 \mid \ldots\}$

4. Tentukan gramar bebas konteks untuk bahasa

   $L_4(G_4) = \{a^n b^m \mid n,m \geq 1,\ n \neq m\}$

   Jawab :

   Langkah kunci : sulit untuk mendefinisikan $L_4(G_4)$ secara langsung. Jalan keluarnya adalah dengan mengingat bahwa $x \neq y$ berarti $x > y$ atau $x < y$.

   $L_4 = L_A \cup L_B$, $L_A = \{a^n b^m \mid n > m \geq 1\}$, $L_B = \{a^n b^m \mid 1 \leq n < m\}$.

   $P_A(L_A) = \{A \rightarrow aA \mid aC,\ C \rightarrow aCb \mid ab\}$, $Q(L_B) = \{B \rightarrow Bb \mid Db,\ D \rightarrow aDb \mid ab\}$

   $P_4(L_4) = \{S \rightarrow A \mid B,\ A \rightarrow aA \mid aC,\ C \rightarrow aCb \mid ab,\ B \rightarrow Bb \mid Db,\ D \rightarrow aDb \mid ab\}$

5. Tentukan sebuah gramar bebas konteks untuk bahasa :
   L_5 = bilangan bulat non negatif genap. Jika bilangan tersebut terdiri dari dua digit atau lebih maka nol tidak boleh muncul sebagai digit pertama.

   Jawab :

Langkah kunci : Digit terakhir bilangan harus genap. Digit pertama tidak boleh nol. Buat tiga himpunan terpisah : bilangan genap tanpa nol (G), bilangan genap dengan nol (N), serta bilangan ganjil (J).

$P_5(L_5) = \{S \rightarrow N \mid GA \mid JA,\ A \rightarrow N \mid NA \mid JA,\ G \rightarrow 2 \mid 4 \mid 6 \mid 8,$

$N \rightarrow 0 \mid 2 \mid 4 \mid 6 \mid 8,\ J \rightarrow 1 \mid 3 \mid 5 \mid 7 \mid 9\}$

PERTEMUAN III

➢ Model FSA

Model matematika dari sebuah sistem dengan input dan output, yang terdiri dari sejumlah berhingga state & fungsi-fungsi transisi yang menyajikan perubahan state di definisikan juga sebagai pasangan 5 tupel (Q, ∑, δ, S, F) mekanisme kerja dapat di aplikasikan pada : lift, text editor, analisa leksikal (pada proses compile) dan parity.

Keterangan :

Q = Himpunan hingga state

∑ = Himpunan hingga simbol input

δ = Fungsi transisi, menggambarkan transisi state FSA akibat pembacaan input

S = State Awal

➢ Definisi FSA

FSA adalah mesin yang dapat mengenali kelas bahasa reguler dan memiliki sifat-sifat :

1. Pita masukan (input tape) berisi rangkaian simbol (string) yang berasal dari himpunan
   simbol / alfabet.
2. Setiap kali setelah membaca satu karakter, posisi read head akan berada pada simbol berikutnya.
3. Setiap saat, FSA berada pada status tertentu
4. Banyaknya status yang berlaku bagi FSA adalah berhingga.

Ada dua jenis FSA :

- Deterministic finite automata (DFA)
  transisi state FSA akibat pembacaan sebuah simbol bersifat tertentu.
- Non deterministik finite automata.(NFA)
  transisi state FSA akibat pembacaan sebuah simbol bersifat tak tentu.

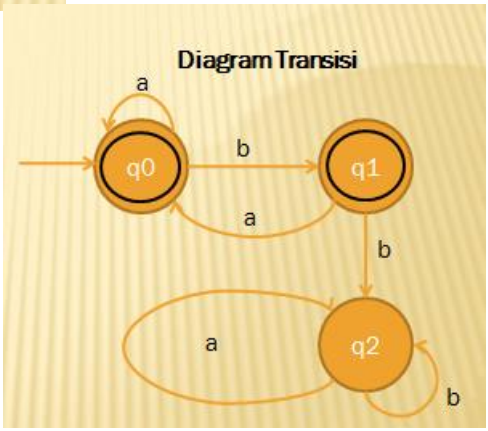
Q = {q0, q1, q2}
∑ = {a,b}
S = q0
F = {q0,q1}
δ diberikan dalam tabel berikut :

δ	a	b
q0	q0	q1
q1	q0	q2
q2	q2	q2

Contoh DFA

Q = {q0, q1, q2}
δ diberikan dalam tabel berikut :
∑= {a, b}
S = q0

F = {q0, q1}



diterima oleh DFA : a, b, aa, abab, baba
oleh DFA : bb, abb, abba

DFA ini menerima semua tersusun dari simbol a dan b mengandung substring bb.

Kalimat yang
ab, ba, aba, bab,
Kalimat yang dittolak

kalimat yang
yang tidak

Contoh NFA

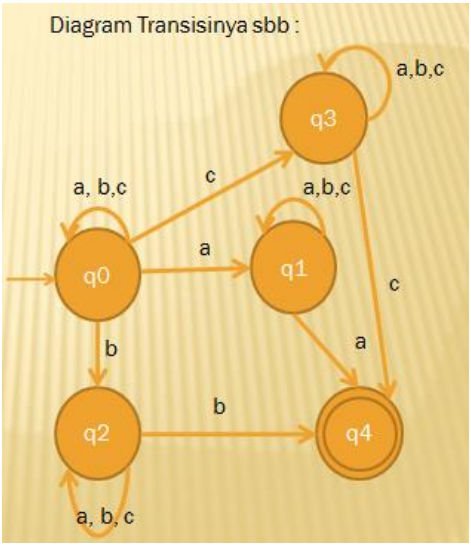
Berikut ini sebuah contoh NFA (Q, ∑, δ, S, F). dimana :
Q = {q0 , q1 , 21 ,q3 , q4 }
∑= {a, b,c}
S = q0

F = {q4}

Diketahu NFA berikut dimana :
Q = {q0, q1, q2, q3,q4}
∑ = {a,b,c}
S = q0
F = {q4}
δ diberikan dalam tabel berikut :

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

Diagram Transisinya sbb :



kalimat yang diterima NFA di atas : aa, bb, cc, aaa, abb, bcc, cbb
kalimat yang tidak diterima NFA di atas : a, b, c, ab, ba, ac, bc

Sebuah kalimat di terima NFA jika :

- salah satu tracing-nya berakhir di state AKHIR, atau
- himpunan state setelah membaca string tersebut mengandung state AKHIR

Fungsi transisi yang diperluas

Fungsi Transisi yang diperluas dimana Mesin M = (Q, Σ, q0, δ, A)

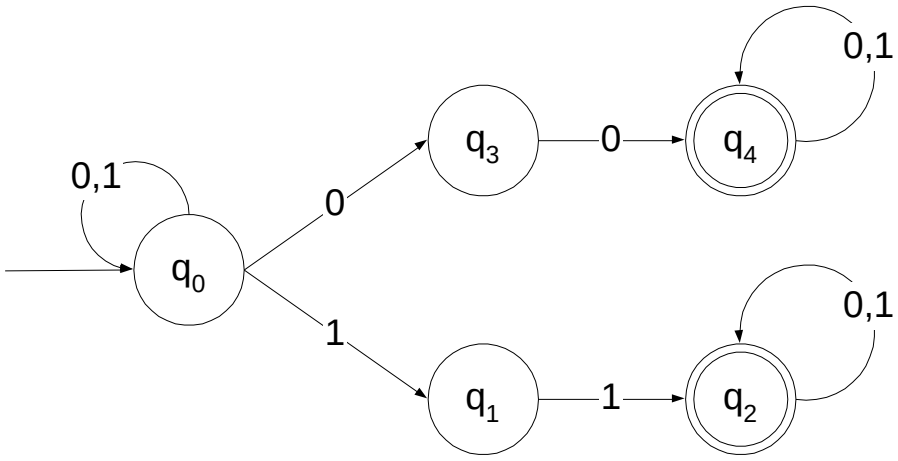
- Fungsi Transisi δ(q, a) menyatakan state mesin M ketika pada state q menerima simbol input a
- Fungsi Transisi δ*(q, x) menyatakan state akhir dari suatu mesin M, ketika menerima input string x dari state q
- Jika δ didefinisikan pada Q x Σ, maka δ* didefinisikan pada Q x Σ*
- Pada mesin M, definisi fungsi transisi yang diperluas :
  δ*: Q x Σ* → Q
  Secara rekursif adalah sebagai berikut:
  – Untuk setiap q ∈ Q, δ*(q, Λ) = q
  – Untuk setiap y ∈ Σ*, a ∈ Σ, and q ∈ Q, δ*(q, ya) = δ(δ*(q, y), a)
- Contoh
  Suatu mesin M: (q) – a →(q1) – b →(q2) – c →(q3)
  δ*(q, abc) = δ(δ*(q, ab), c) = δ(δ(δ*(q, a), b, c)
  = δ(δ(δ*(q, Λa), b, c)) = δ(δ(δ(δ *(q, Λ), a), b), c)
  = δ(δ(δ(q, a), b, c)) = δ(δ(q1, b), c)
  = δ(q2, c) = q3

Penggunaan Fungsi Transisi yang diperluas :

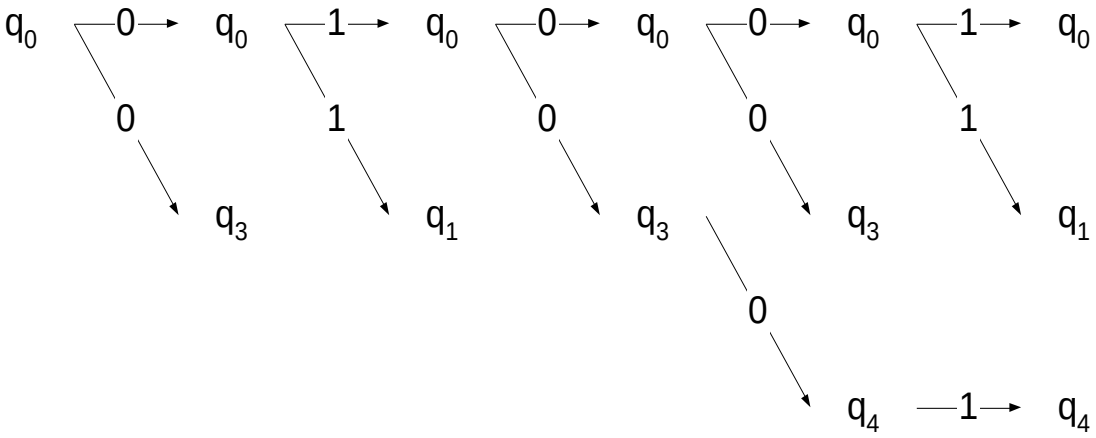
Fungsi ini dapat digunakan untuk menyatakan dengan tepat apa artinya bagi sebuah FA menerima sebuah string: M = (Q, Σ, q0, δ, A) adalah sebuah FA. Sebuah string x ∈ Σ* diterima oleh M jika δ*(q0, x) ∈A. x ditolak oleh M jika tidak diterima. Bahasa yang diterima atau dikenali oleh M adalah bahasa L(M) = { x | x diterima M}. Jika L adalah suatu bahasa dalam Σ*, L diterima atau dikenali oleh M jika L = L(M)

- Perbedaan dengan NFA: fungsi transisi dapat memiliki 0 atau lebih fungsi transisi
- G = ({q_0 , q_1 , q_2 , q_3, q_4 }, {0,1}, δ , q_0 , { q_2 , q_4}}

δ	0	1
q_0	{ q_0,q_3}	{q_0,q_1}
q_1	ε	{q_2}
q_2	{q_2}	{q_2}
q_3	{q_4}	ε
q_4	{q_4}	{q_4}

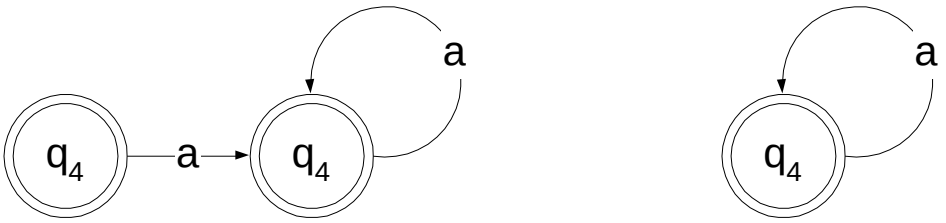


- String diterima NFA bila terdapat suatu urutan transisi berdasar input, dari state awal ke state akhir.
- harus mencoba semua kemungkinan.
- Contoh : string 01001



Def 2. Dua buah FSA disebut ekuivalen apabila kedua FSA tersebut menerima bahasa yang sama

Contoh : FSA yang menerima bahasa $\{a^n \mid n \geq 0\}$



Def 3. Dua buah state dari FSA disebut indistinguishable (tidak dapat dibedakan) apabila :

$\delta(q,w) \in F$ sedangkan $\delta(p,w) \notin F$ dan

$\delta(q,w) \notin F$ sedangkan $\delta(p,w) \in F$ untuk semua $w \in \Sigma^*$

Def 4. Dua buah state dari FSA disebut distinguishable (dapat dibedakan) bila terdapat $w \in \Sigma^*$ sedemikian hingga:

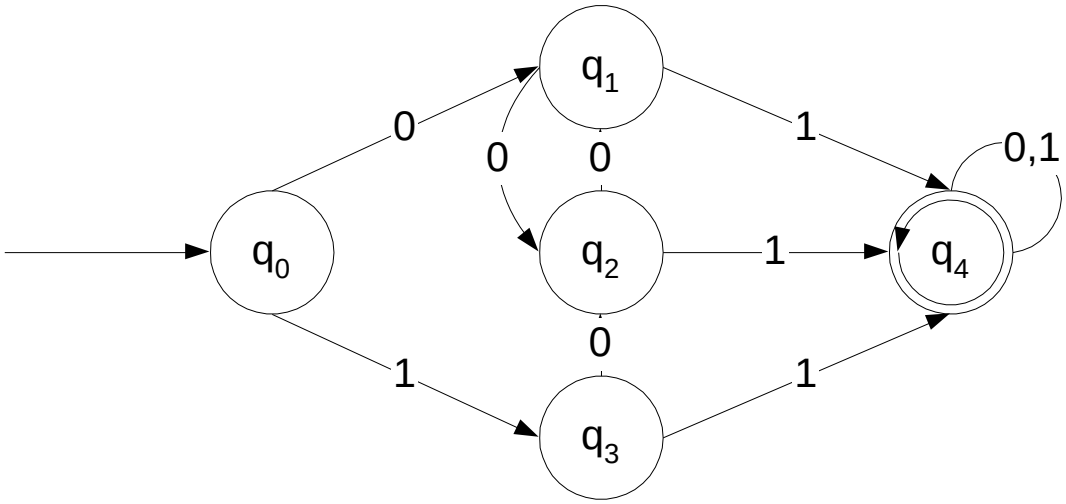
$\delta(q,w) \in F$ sedangkan $\delta(p,w) \notin F$ dan

$\delta(q,w) \notin F$ sedangkan $\delta(p,w) \in F$ untuk semua $w \in \Sigma^*$

Prosedur menentukan pasangan status indistinguishable

1. Hapus semua state yang tak dapat dicapai dari state awal.
2. Catat semua pasangan state (p,q) yang distinguishable, yaitu $\{(p,q) \mid p \in F \wedge q \notin F\}$
3. Untuk setiap pasangan (p,q) sisanya,
   untuk setiap $a \in \Sigma$, tentukan $\delta(p,a)$ dan $\delta(q,a)$

Contoh



1. Hapus state yang tidak tercapai -> tidak ada
2. Pasangan distinguishable (q_0,q_4), (q_1,q_4), (q_2,q_4), (q_3,q_4).
3. Pasangan sisanya (q_0,q_1), (q_0,q_2), (q_0,q_3), (q_1,q_2) (q_1,q_3) (q_2,q_3)

pasangan	state 1		state 2		hasil
	0	1	0	1	
(q_0,q_1)	q1	q3	q2	q4	distinguishable
(q_0,q_2)	q1	q3	q1	q4	distinguishable
(q_1,q_2)	q2	q4	q1	q4	indistinguishable
(q_0,q_3)	q1	q3	q2	q4	distinguishable
(q_1,q_3)	q2	q4	q2	q4	indistinguishable
(q_2,q_3)	q1	q4	q2	q4	indistinguishable

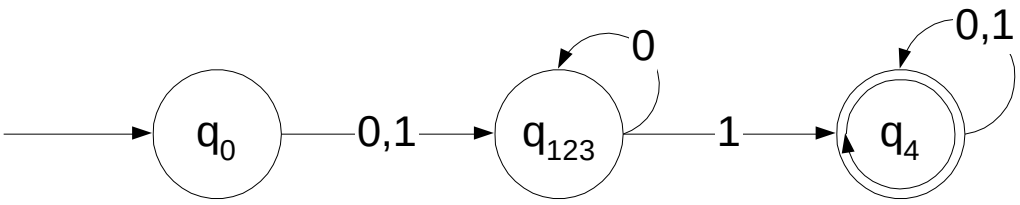
Catatan : jumlah pasangan seluruhnya : $C\binom{5}{2} = \frac{5!}{2!3!} = 10$

Prosedur Reduksi DFA

1. Tentukan pasangan status indistinguishable.
2. Gabungkan setiap group indistinguishable state ke dalam satu state dengan relasi pembentukan group secara berantai : Jika p dan q indistingishable dan jika q dan r indistinguishable maka p dan r indistinguishable, dan p,q serta r indistinguishable semua berada dalam satu group.
3. sesuaikan transisi dari dan ke state-state gabungan.

Contoh

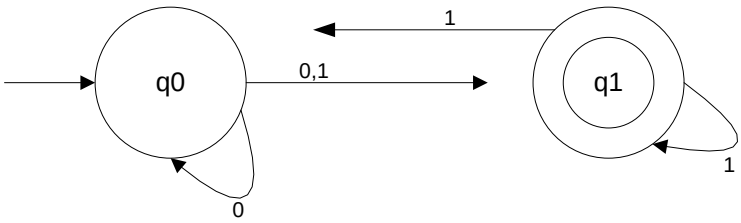
1. pasangan status indistinguishable (q_1,q_2), (q_1,q_3) dan (q_2,q_3).
2. q_1,q_2,q_3 ketiganya dapat digabung dalam satu state q_{123}
3. Menyesuaikan transisi, sehingga DFA menjadi



PERTEMUAN IV

- **<u>Ekivalen DFA dan FSA</u>**

Dari sebuah NFA dapat dibuat bentuk DFA nya yang ekivalen (bersesuaian). Ekivalen disini artinya mampu memproduksi atau menerima bahasa yang sama. Adapun tahap pembuatan DFA yang ekivalen dari suatu NFA adalah sebagai berikut: Contoh Diketahui NFA sebagai berikut

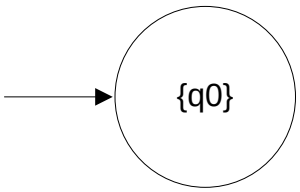


Konfigurasi NFA contoh 4 secara formal adalah sebagai berikut :

Q = {q0, q1 }
Σ = {0, 1}
S = q0
F = {q1}
Tabel transisinya :

δ	0	1
q0	{q0, q1}	{q1}
q1	Ø	{q0, q1}

Kita mulai dengan state {q0}

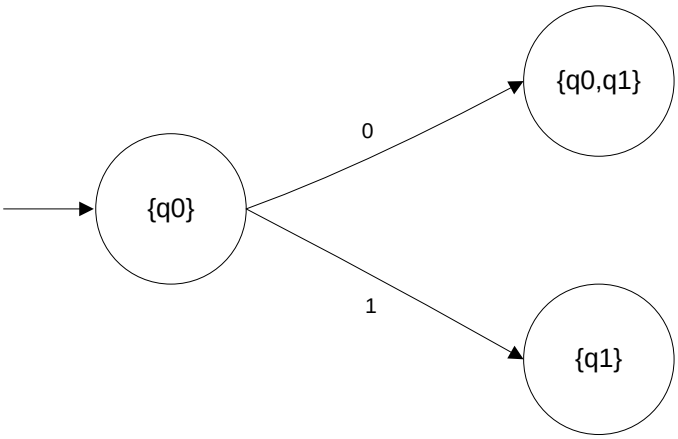


Telusuri state berikutnya :

• δ (q0, 0) = {q0, q1 }

• δ (q0, 1) = {q1}

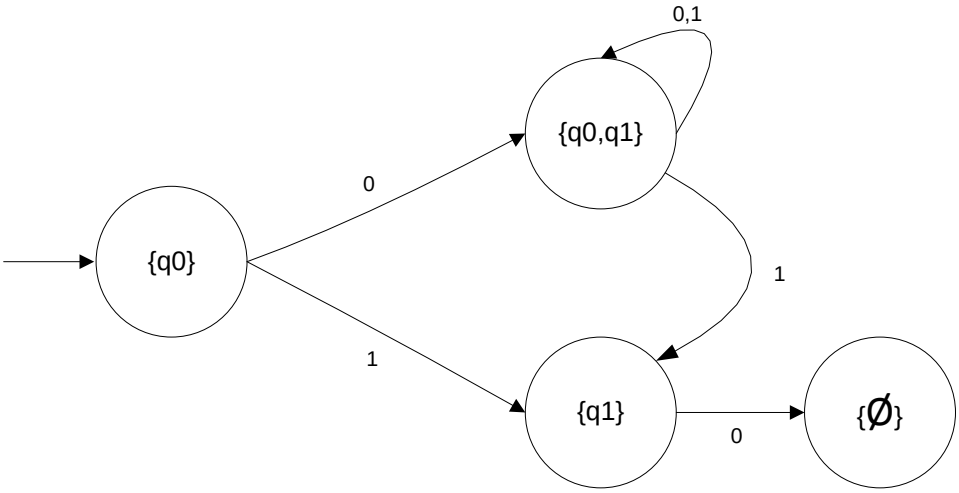
Hasilnya :



Selanjutnya telusuri untuk setiap state baru yang terbentuk :

- δ (q1, 0) = Ø
- δ (q1, 1) = {q0, q1}
- δ ({q0,q1}, 0) = {q0, q1} adalah hasil gabungan dari
  δ (q0, 0) = {q0, q1}dengan δ (q1, 0) = Ø
- δ ({q0,q1}, 1) = {q0, q1} adalah hasil gabungan dari
  δ (q0, 1) = {q1}dengan δ (q1, 0) = {q0, q1}

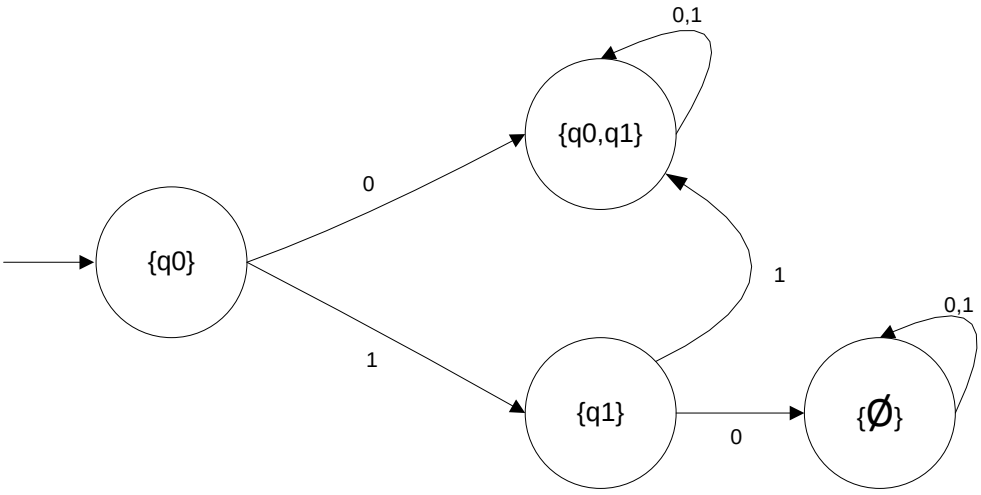
Hasilnya :



Selanjutnya telusuri state baru yang terbentuk :

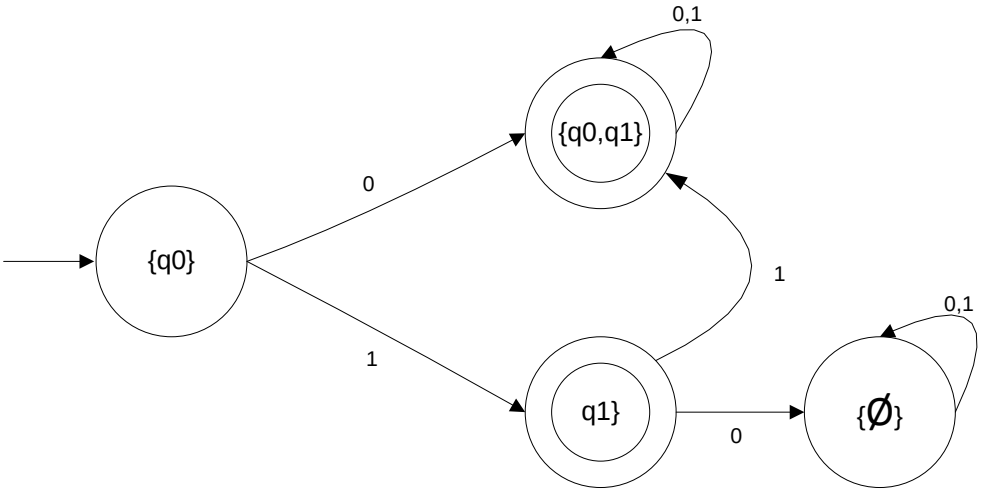
- δ (Ø, 0) = Ø
- δ (Ø, 0) = Ø

Hasilnya :



Selanjutnya kita ingat bahwa F = {q1} maka himpunan state akhir (F) sekarang adalah semua yang mengandung state q1.
F = {{q1}, {q0, q1}}

Hasilnya :



Ekivalen DFA dan FSA

Ekivalensi DFA dan NFA :

- Suatu DFA dapat dipandang sebagai kasus khusus (subset) dari NFA.
- Jelas bahwa kelas bahasa yang diterima oleh DFA juga akan diterima oleh DFA
- Namun ternyata DFA juga dapat mensimulasikan NFA; yaitu untuk setiap NFA kita dapat membuat DFA yang ekivalen.
- Dapat dibuktikan bahwa DFA dan NFA adalah ekivalen, sehingga dapat disebut FA saja.

Simulasi NFA oleh DFA :

- Cara simulasi NFA oleh DFA adalah dengan membuat state DFA berkorespondensi dengan set state di NFA

- DFA yang dibentuk mencatat semua state yang mungkin pada NFA setelah membaca input tertentu

Pembuktian :

- Teorema: Jika L adalah himpunan yang diterima oleh NFA maka ada sebuah DFA yang menerima L
- Misalnya sebuah NFA M = (Q, Σ, q0, δ, A) dan DFA M' = (Q', Σ', q0', δ', A'). State pada M' adalah semua subhimpunan dari himpunan state M, yaitu Q' = 2Q. M' akan mencatat dalam statenya semua state M yang mungkin pada waktu tertentu.
- F' adalah himpunan semua state di Q' yang mengandung final state dari M.
- Elemen Q' akan dinyatakan sebagai [q1, q2, …, qi] dimana q1, q2, …, qi ada di Q
  – [q1, q2, …, qi] adakah satu state dalam DFA yang berkorespondensi dengan suatu himpunan state di NFA
  – q0' = [q0]
- δ'([q1, q2, …., qi], a) = [p1, p2, …, pj] jika dan hanya jika δ'({q1, q2, …., qi}, a) = {p1, p2, …, pj}
- Aplikasi δ' terhadap elemen [q1, q2, …, qi] dari Q' dihitung dengan mengaplikasikan δ terhadap setiap q1, q2, …, qi dan membuat gabungannya (unionnya), Gabungan tersebut digunakan untuk membuat set state baru p1, p2…pj. Himpunan baru ini memiliki representasi [p1, p2, …, pj] di Q', dan elemen tersebut adalah nilai dari δ'([q1, q2, …., qi], a)
- Dengan menggunakan induksi, dapat dibuktikan bahwa: δ'(q0', x) = [q1, q2, …, qi] jika dan hanya jika δ(q0, x) = {q1, q2, …, qi}. Jadi, dapat dibuat sebuah mesin DFA yang menerima bahasa yang sama dengan yang diterima oleh sebuah mesin NFA.

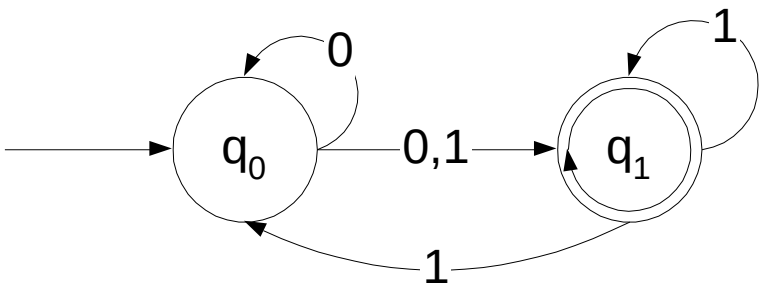
- Contoh : permainan catur, banyak alternatif pada suatu posisi tertentu -> nondeterministic
- Non deterministik dapat menyelesaikan problem tanpa backtrack, namun dapat diekuivalensikan ke DFA.

Algoritma

1. Buat semua state yang merupakan subset dari state semula. jumlah state menjadi 2^Q
2. Telusuri transisi state–state yang baru terbentuk, dari diagram transisi.
3. Tentukan state awal : $\{q_0\}$
4. Tentukan state akhir adalah state yang elemennya mengandung state akhir.
5. Reduksi state yang tak tercapai oleh state awal.

Contoh Ubahlah NFA berikut menjadi DFA
M=$\{\{q_0,q_1\}, \{0,1\}, \delta, q_0, \{q_1\}\}$ dengan tabel transisi

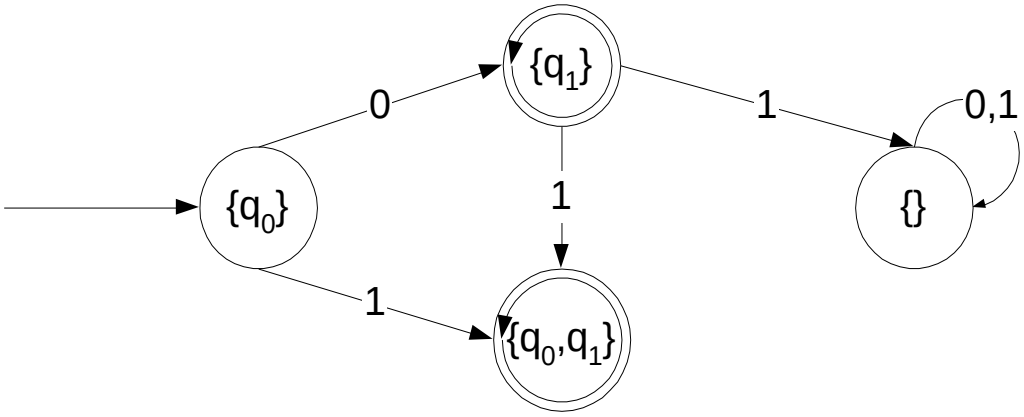
δ	0	1
q_0	$\{q_0,q_1\}$	q_1
q_1	{}	$\{q_0,q_1\}$



1. State yang akan dibentuk : {}, {q_0} {q_1},{q_0,q_1}
2. Telusuri state

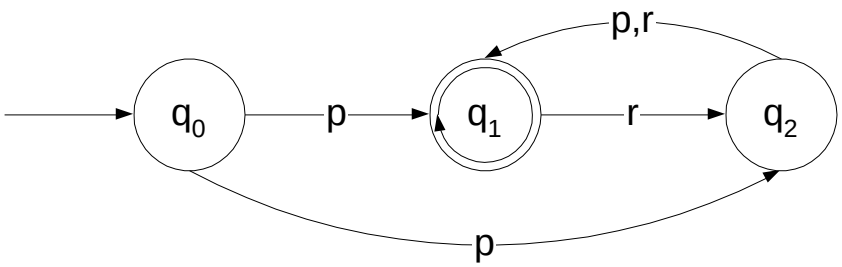
δ	0	1
{}	{}	{}
{q_0}	{q_0,q_1}	{q_1}
{q_1}	{}	{q_0,q_1}
{q_0,q_1}	{q_0,q_1}	{q_0,q_1}

3. State awal : {q0}
4. State akhir yang mengandung q_1, yaitu {q_1},{q_0,q_1}



Contoh : Ubahlah NFA berikut menjadi DFA
M={{q_0,q_1 ,q_2}, {p,r}, δ, q_0,{q_1}} dengan tabel transisi

δ	0	1
q_0	{q_1,q_2}	{}
q_1	{}	{q_0,q_1}
q_2	{q_1}	{q_1}

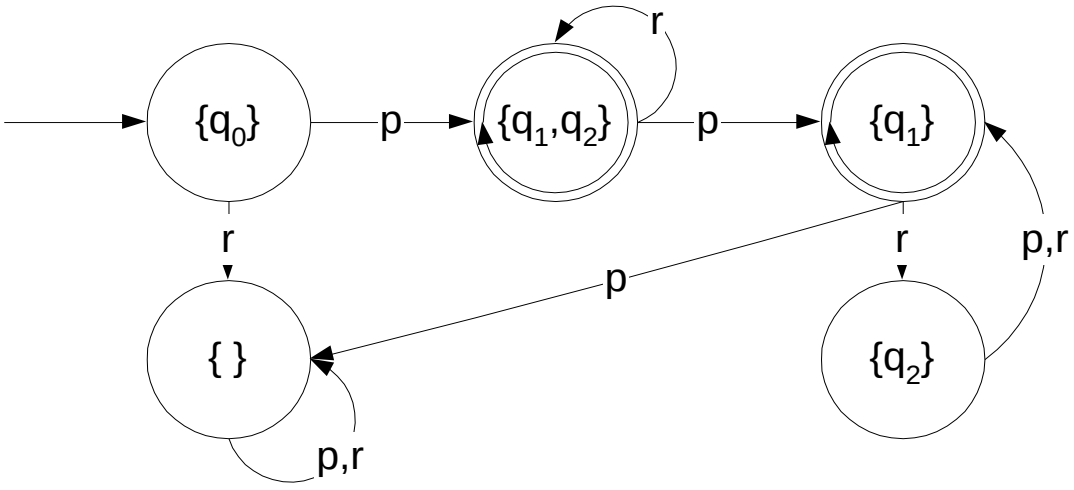


1. State yang akan dibentuk : {}, {q_0} {q_1},{q_2}, {q_0,q_1}, {q_0,q_2}, {q_1,q_2}, {q_0,q_1,q_2}

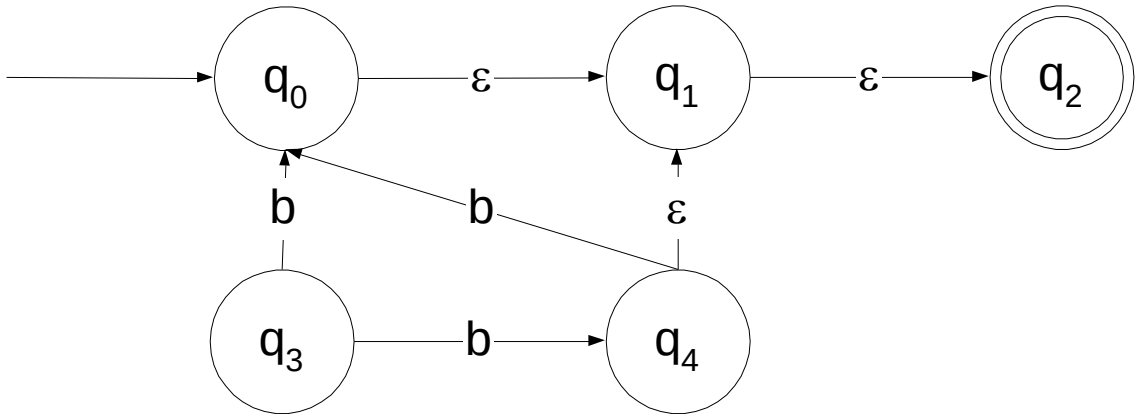
2. Telusuri state:

δ	p	r
{}	{}	{}
{q_0}	{q_1,q_2}	{}
{q_1}	{}	{q_2}
{q_2}	{q_1}	{q_1}
{q_0,q_1}	{q_1,q_2}	{q_2}
{q_0,q_2}	{q_1,q_2}	{q_1}
{q_1,q_2}	{q_1}	{q_1,q_2}
{q_0,q_1,q_2 }	{q_1,q_2}	{q_1,q_2}

3. State awal : {q_0}
4. State akhir yang mengandung q_1, yaitu {q_1},{q_1,q_2}
5. Reduksi {q_0,q_1}{q_0,q_2}{q_0,q_1,q_2 } sehingga FSA menjadi



NFA dengan ε-move



Def 1. ε-move adalah suatu transisi antara 2 status tanpa adanya input. Contoh gambar : transisi antara status q_1 ke q_3

Def 2. ε-closure adalah himpunan state yang dapat dicapai dari suatu state tanpa adanya input. Contoh gambar :

ε-closure(q_0) = [q_0,q_1,q_3]

ε-closure(q_1) = [q_1,q_3]

ε-closure(q_3) = [q_3]

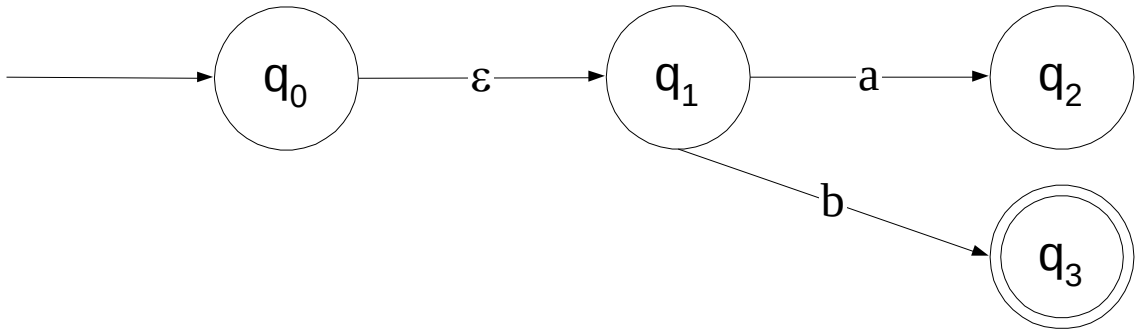
Ekuivalensi NFA dengan ε-move ke NFA tanpa ε-move

1. Buat tabel transisi NFA dengan ε-move
2. Tentukan ε-closure setiap state
3. Carilah fungsi transisi /tabel transisi yang baru, rumus :

   δ'(state,input)=ε-closure(δ(ε-closure(state,input))
4. Tentukan state akhir ditambah dengan state yang ε-closure nya menuju state akhir, rumusnya

   $F' = F \cup \{q \mid (\varepsilon\text{-closure}(q) \cap F \neq \varnothing\}$

Contoh



Tabel transisi-nya

δ	0	1
q_0	$\varnothing$	$\varnothing$
q_1	q_2	q_3
q_2	$\varnothing$	$\varnothing$
q_3	$\varnothing$	$\varnothing$

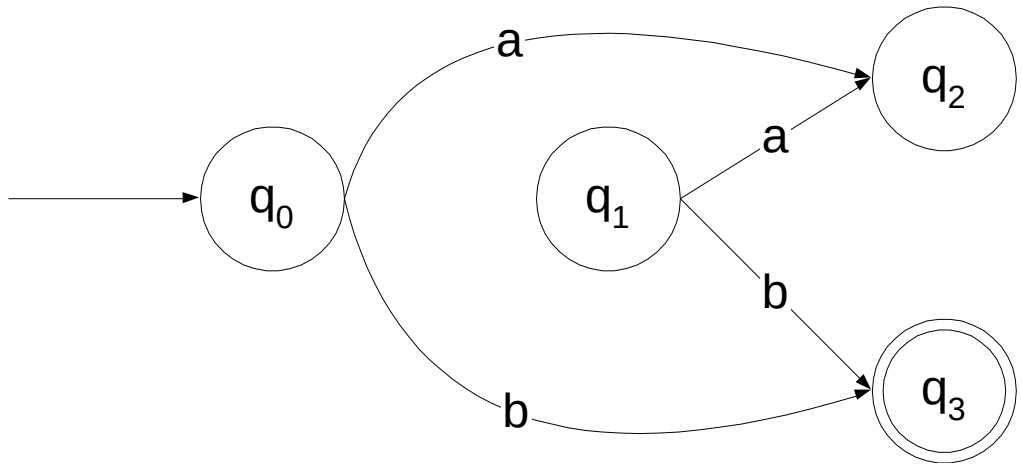
ε-closure dari FSA tersebut

ε-closure(q_0) = [q_0,q_1]
ε-closure(q_1) = [q_1]
ε-closure(q_2) = [q_2]
ε-closure(q_3) = [q_3]

Cari tabel transisi yang baru (δ') :

δ'	A	b
q_0	ε-cl(δ(ε-cl(q_0),a)) ε-cl(δ({q_0,q_1},a)) ε-cl(q_2) {q_2}	ε-cl(δ(ε-cl(q_0),b)) ε-cl(δ({q_0,q_1},b)) ε-cl(q_3) {q_3}
q_1	ε-cl(δ(ε-cl(q_1),a)) ε-cl(δ({q_1},a)) ε-cl(q_2) {q_2}	ε-cl(δ(ε-cl(q_1),b)) ε-cl(δ({q_1},b)) ε-cl(q_3) {q_3}
q_2	ε-cl(δ(ε-cl(q_2),a)) ε-cl(δ({q_3},a)) ε-cl($\varnothing$) $\varnothing$	ε-cl(δ(ε-cl(q_2),b)) ε-cl(δ({q_2},b)) ε-cl($\varnothing$) $\varnothing$
q_3	ε-cl(δ(ε-cl(q_3),a)) ε-cl(δ({q_3},a)) ε-cl($\varnothing$) $\varnothing$	ε-cl(δ(ε-cl(q_3),b)) ε-cl(δ({q_3},b)) ε-cl($\varnothing$) $\varnothing$

Hasilnya menjadi

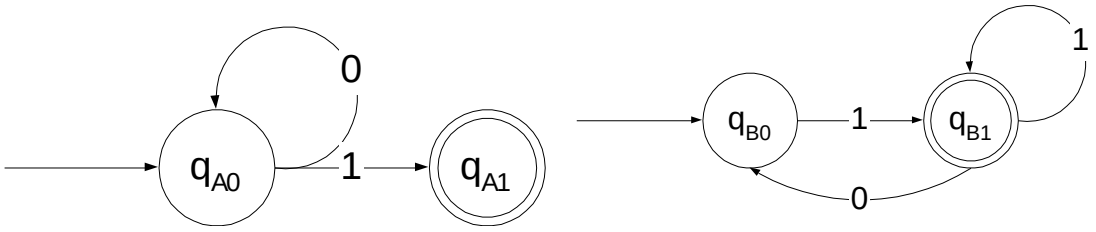


Penggabungan FSA

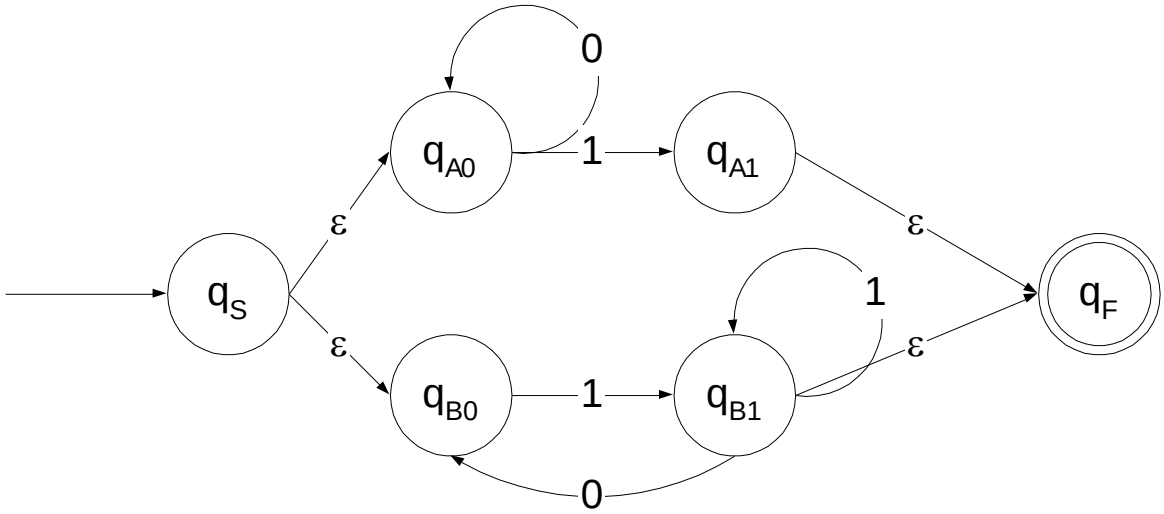
Bila diketahui L_1 adalah bahasa yang diterima oleh M1 dan L2 adalah bahasa yang diterima oleh M2 maka

1. FSA M3 yang dapat menerima L_1+L_2 dibuat dengan cara
   - Tambahkan state awal untuk M3, hubungkan dengan state awal M1 dan state awal M2 menggunakan transisi ε
   - Tambahkan state akhir untuk M3, hubungkan dengan state-state akhir M1 dan state-state akhir M2 menggunakan transisi ε
2. FSA M4 yang dapat menerima L_1L_2 dibuat dengan cara
   - State awal M1 menjadi state awal M4
   - State-state akhir M2 menjadi state-state akhir M4
   - Hubungkan state-state akhir M1 dengan state awal M2 menggunakan transisi ε

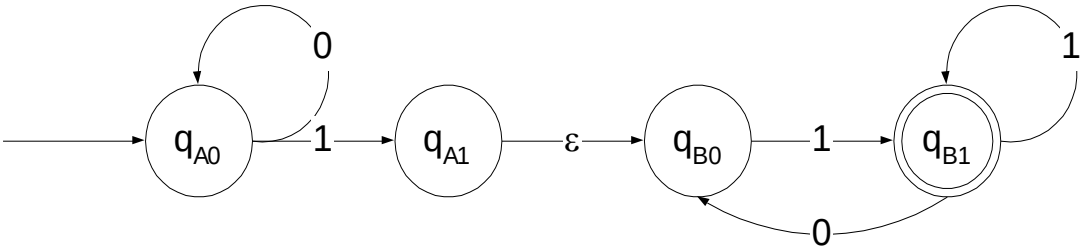
Contoh FSA M1 dan M2



FSA M3



FSA M4



PERTEMUAN V

Finite Automata dengan Keluaran

FSA hanya memberikan status keluaran berupa indikasi biner "diterima" atau "ditolak" terhadap string masukan. Dibutuhkan mesin finite state lain yang menghasilkan keluaran bukan biner tapi suatu simbol alfabet lain. Finite State Transducer (FST) adalah mesin yang menerima string masukan dan menerjemahkannya menjadi string keluaran.
FSA : accepter, dapat menerima atau tidak.
FSA dengan output : transducer
Pendekatan perancangan FST:

- FST yang keluarannya diasosiasikan dengan suatu status, disebut mesin Moore.
- FST yang keluarannya diasosiasikan dengan suatu transisi, disebut mesin Mealy.

1. Mesin Moore

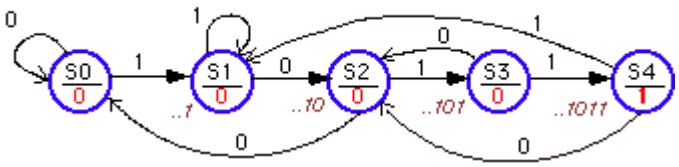
Mesin Moore dinyatakan dengan 6-tuple (Q, Σ, Δ, δ, λ, q0), dimana :
Q: himpunan berhingga status.
Σ: himpunan berhingga simbol alfabet.
Δ: himpunan simbol keluaran (alfabet keluaran).
δ : fungsi transisi yang memetakan Q x Σ ke Q.
λ: fungsi yang memetakan Q ke Δ, memberikan keluaran yang diasosiasikan dengan tiap status.
q0: status awal, anggota Q.

Keluaran mesin Moore terhadap masukan a1 a2…an≥n adalah λ(q0)λ(q1)…λ(qn) dimana q0, q1,…,qn adalah barisan status sedemikian sehingga δ(qi-1,ai) = qi untuk 1 ≤ i≤ n. Jika string masukan ε, mesin Moore memberikan keluaran λ(q0).

Contoh :
Mesin Moore yang menghasilkan keluaran modulo 5 dari suatu bilangan bulat positif biner adalah : (Q, Σ, Δ, δ, λ, q0) dimana :
Q = {q0, q1, q2, q3, q4}
Σ = {0,1}
Δ = {0,1,2,3,4}
λ = Q → Δ, yaitu λ (q0) = j untuk j = 0,1,2,3,4

Q x Σ → Q didefinisikan sbb:

Status	Masukan	
	0	1
q0	q0	q1
q1	q2	q3
q2	q4	q0
q3	q1	q2
q4	q3	q4



2. Mesin Mealy

Mesin Mealy dinyatakan dengan 6-tuple (Q, Σ, Δ, δ, λ, q0), dimana:
Q: himpunan berhingga status.
Σ: himpunan berhingga simbol alfabet.
Δ: himpunan simbol keluaran (alfabet keluaran).
δ : fungsi transisi yang memetakan Q x Σ ke Q.
λ: fungsi yang memetakan Q x Σ ke Δ, λ(q,a) memberikan keluaran yang di asosiasikan dengan transisi dari q terhadap symbol keluaran a.
q0: status awal, anggota Q.

Keluaran mesin Mealy terhadap masukan a1 a2…an ≥ n adalah λ(q0,a1) λ(q0,a1) λ(q1,a2) … λ (qn-1,an) dimana q0, q1,…,qn-1 adalah barisan status sedemikian sehingga δ(qi-1,ai) = qi untuk 1 ≤ I ≤ n. Jika string masukan ε, mesin Mealy memberikan keluaran ε.

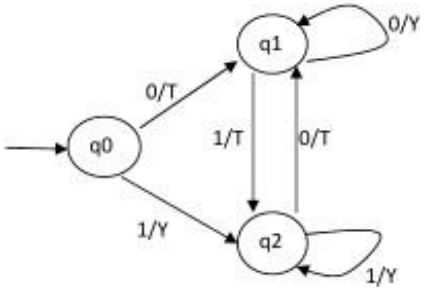
Contoh :
Mesin Mealy yang menerima bahasa himpunan string dari alfabet {0,1} yang dua simbol akhirnya sama adalah: (Q, Σ, Δ, δ, λ, q0) dimana:
Q = {q0, q1, q2}
Σ = {0,1}
Δ = {y,n}

δ = Q x Σ → Q

Status	Masukan	
	0	1
q0	q1	q2
q1	q1	q2
q2	q1	q2

λ = Q x Σ → Δ

Status	Masukan	
	0	1
q0	T	T
q1	Y	T
q2	T	Y



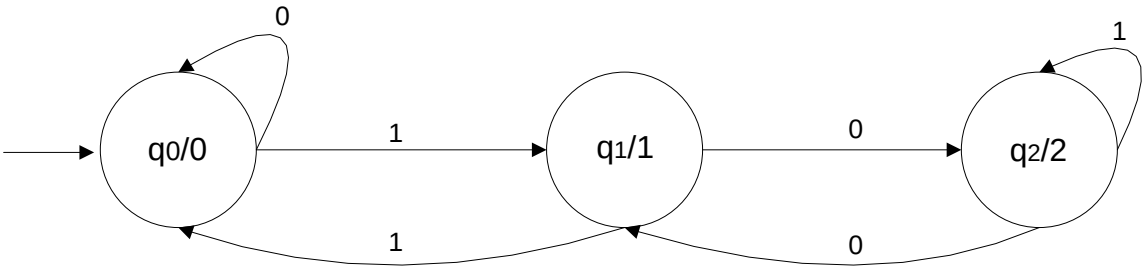
3. Mesin Moore

M = (Q,Σ,δ,S,Δ,λ)
Q : himpunan state
Σ: himpunan simbol input
δ: fungsi transisi
S : state awal S ∈ Q
Δ: himpunan output
λ: fungsi output untuk setiap state

Contoh mesin moore untuk memperoleh modulus 3 pada suatu bilangan biner:
M = (Q,Σ,δ,S,Δ,λ)
Q : q0,q1,q2
Σ: [0,1]
S : q0
Δ: [0,1,2]
λ(q0) =0
λ(q1) =1
λ(q2) =2

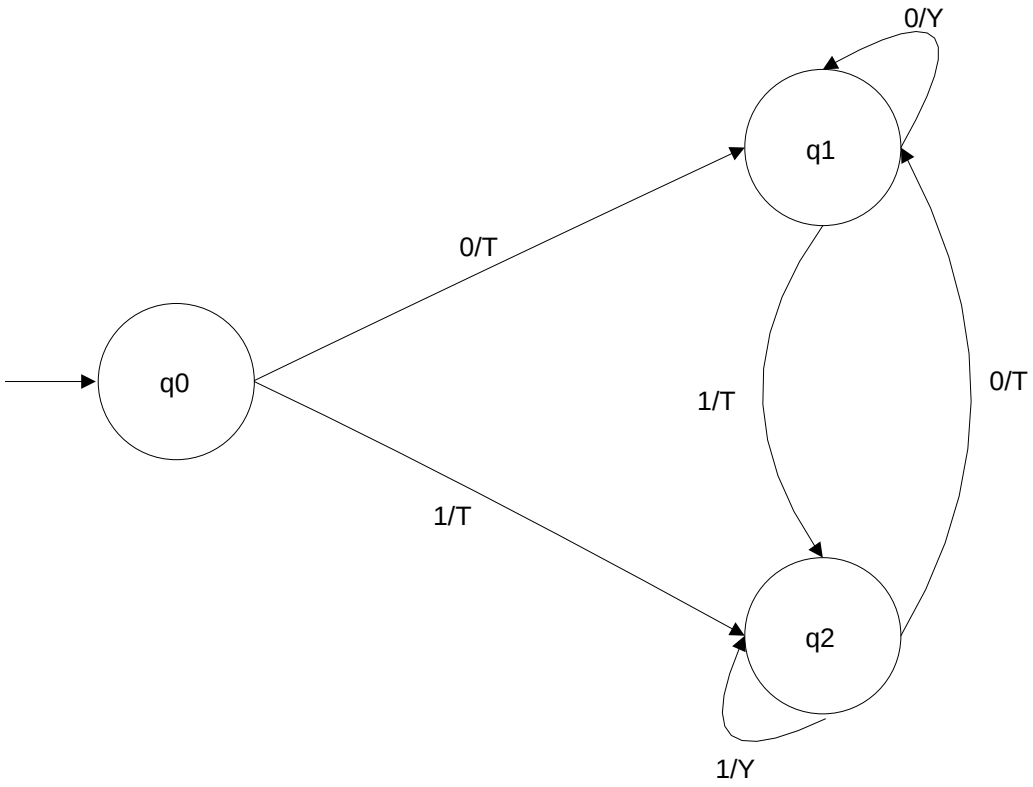
Prinsip:
jika i diikuti dengan 0, maka hasilnya 2i
1012=5 10102= 2*5 =10
jika i diikuti dengan 1, maka hasilnya 2i+1
1012=5 10112= 2*5+1 =11

jika i/3 mempunyai sisa p, maka untuk input berikutnya bernilai 0 maka
2i/3 mempunyai sisa 2p mod 3
untuk p=0 maka 2p mod 3 = 0
untuk p=1 maka 2p mod 3 = 2
untuk p=2 maka 2p mod 3 = 1
jika i/3 mempunyai sisa p, maka untuk input berikutnya bernilai 1 maka
(2i+1)/3 mempunyai sisa (2p+1) mod 3
untuk p=0 maka (2p+1) mod 3 = 1
untuk p=1 maka (2p+1) mod 3 = 0
untuk p=2 maka (2p+1) mod 3 = 2
Sehingga didapat mesin FSA sbb :



Contoh :
input 5 (1012) , state terakhir q2/2 , 5 mod 3 = 2
input 10 (10102) , state terakhir q1/1 , 10 mod 3 = 1
Mesin Mealy
M = (Q,Σ,δ,S,Δ,λ)
Q : himpunan state
Σ : himpunan simbol input
δ : fungsi transisi
S : state awal S ∈Q
Δ : himpunan output
λ : fungsi output untuk setiap **transisi**

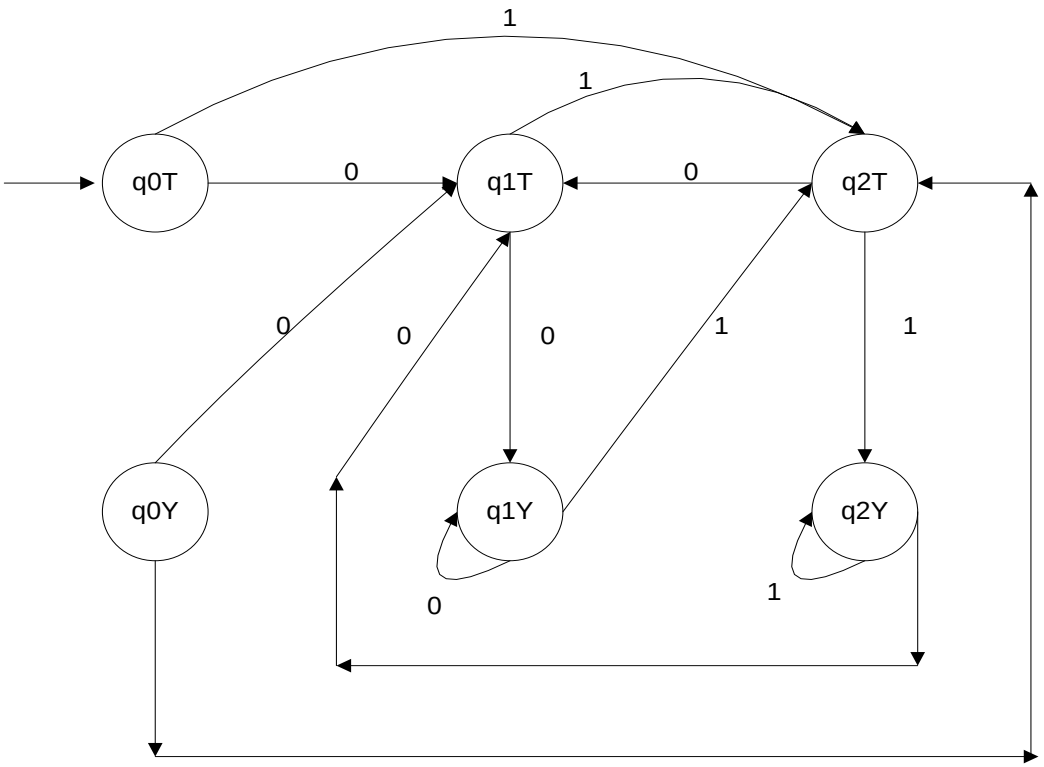
Contoh mesin Mealy untuk mendeteksi ekspresi reguler
(0+1)*(00+11)
Jawab
M = (Q,Σ,δ,S,Δ,λ)
Q : q0,q1,q2
Σ : [0,1]
S : q0
Δ : [0,1,2]
λ(q0,0) =T
λ(q0,1) =T
λ(q1,0) =Y
λ(q1,1) =T
λ(q2,0) =T
λ(q2,1) =Y



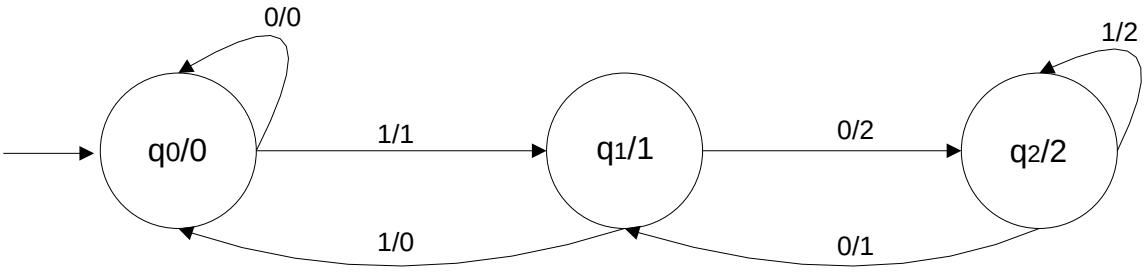
Ekuivalensi mesin Moore dengan mesin Mealy

♦ Mesin Moore ke mesin Mealy

Jml state = jml state sebelum * jml output



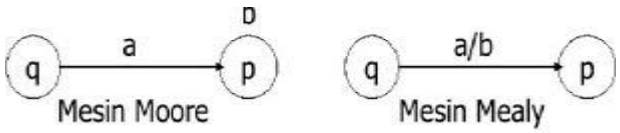
♦ Mesin Mealy ke mesin Moore
Menambah label output pada transisi
Menghapus label output pada state



3. Ekivalensi Mesin Moore dan Mesin Melay
Jika diberikan mesin Moore maka kita dapat membuat mesin Mealy dan sebaliknya.
Diberikan :

- mesin Moore M1 = (Q, Σ, Δ, δ, λ, q0)
- mesin Mealy M2 = (Q, Σ, Δ, δ, λ', q0)

Maka didefinisikan λ'(q,a) = λ(δ (q,a)) untuk semua q di dalam Q, a di dalam Σ, dan b di dalam Δ.



jika diberikan :

- mesin Mealy M1 = (Q, Σ, Δ, δ, λ, q0)
- mesin Moore M2 = (Q, Σ, Δ, δ', λ', [q0 ,b0])

Mesin Moore M2 yg ekivalen dengan M1 dibuat dengan memecah setiap status dari M1 menjadi sejumlah |Q| x |Δ| status yg berbeda pada M2.
Maka didefinisikan :

- δ'([q,b],a)=[δ(q,a), λ(q,a)]
- λ'(q,b) = b