Automata dan Teori Bahasa

Pertemuan 4
Non-Deterministic Finite Automata

Definisi NDFA

- Untuk simbol input tertentu, mesin dapat berpindah ke kombinasi status apa pun di mesin.
- Dengan kata lain, keadaan pasti pergerakan mesin tidak dapat ditentukan.

Tuple pada NDFA

- 5-tuple $(Q, \sum, \delta, q_0, F)$

Penjelasan Tuple NDFA

- Q adalah himpunan state yang terbatas.
- $\sum$ adalah kumpulan simbol terbatas yang disebut huruf.
- δ adalah fungsi transisi di mana $\delta: Q \times \sum \rightarrow 2^Q$
  - (Di sini kumpulan daya $Q(2^Q)$ telah diambil karena dalam kasus NDFA, dari suatu keadaan, transisi dapat terjadi ke kombinasi keadaan Q apa pun)
- q_0 adalah status awal dari mana input diproses ($q_0 \in Q$).
- F adalah himpunan keadaan akhir dari Q ($F \subseteq Q$).

Representasi Grafik NDFA

- Simpul mewakili state.
- Busur yang diberi label alfabet masukan menunjukkan transisi.
- Keadaan awal dilambangkan dengan busur masuk tunggal yang kosong.
- Keadaan akhir ditunjukkan oleh lingkaran ganda.

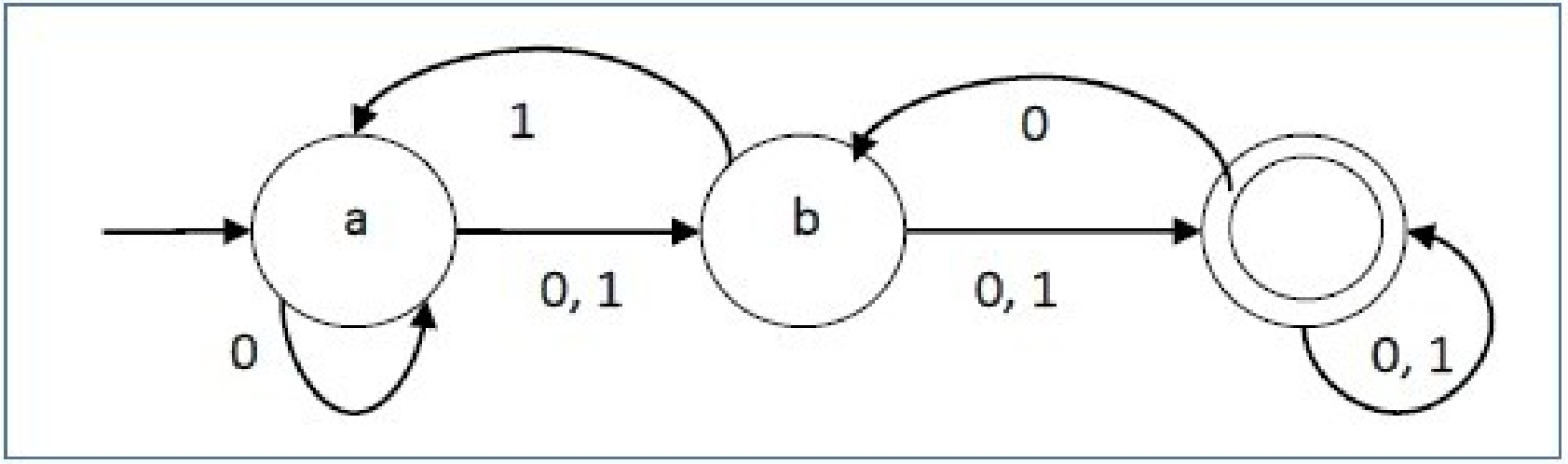
Contoh 1

- Q = {a, b, c}
- ∑ = {0, 1}
- q_0 = {a}
- F = {c}

Contoh 1 : Transisi (δ)

q_0	q_1	q_2
a	a,b	b
b	c	a,c
c	b,c	c

Contoh 1 : Graph



Perbedaan DFA dan NDFA

DFA	NDFA
Transisi dari suatu keadaan adalah ke keadaan berikutnya hanya satu tujuan.	Transisi dari suatu keadaan adalah ke keadaan berikutnya bisa lebih dari satu tujuan.
Transisi string kosong tidak dizinkan di DFA.	NDFA mengizinkan transisi string kosong.
Backtracking memungkinkan di DFA	Di NDFA, backtracking tidak selalu memungkinkan.
Membutuhkan lebih banyak ruang.	Membutuhkan lebih sedikit ruang.
Sebuah string diterima oleh DFA, jika ditransfer ke status akhir.	Sebuah string diterima oleh NDFA, jika setidaknya satu dari semua kemungkinan transisi berakhir dalam keadaan akhir.

Acceptor (Recognizer)

- Automation yang menghitung fungsi Boolean disebut Acceptor.
- Semua status Acceptor menerima atau menolak masukan yang diberikan kepadanya.

Classifier

- Classifier memiliki lebih dari dua status akhir dan memberikan satu keluaran saat dihentikan.

Transducer

- Sebuah automation yang menghasilkan keluaran berdasarkan masukan saat ini dan / atau keadaan sebelumnya disebut transducer.

Mealy Machine

- Outputnya tergantung pada keadaan saat ini dan input saat ini.

Moore Machine

- Outputnya hanya bergantung pada keadaan saat ini.

String Acceptance

- String S diterima oleh DFA/NDFA $(Q, \sum, \delta, q_0, F)$, iff
  - $\delta^*(q_0, S) \in F$

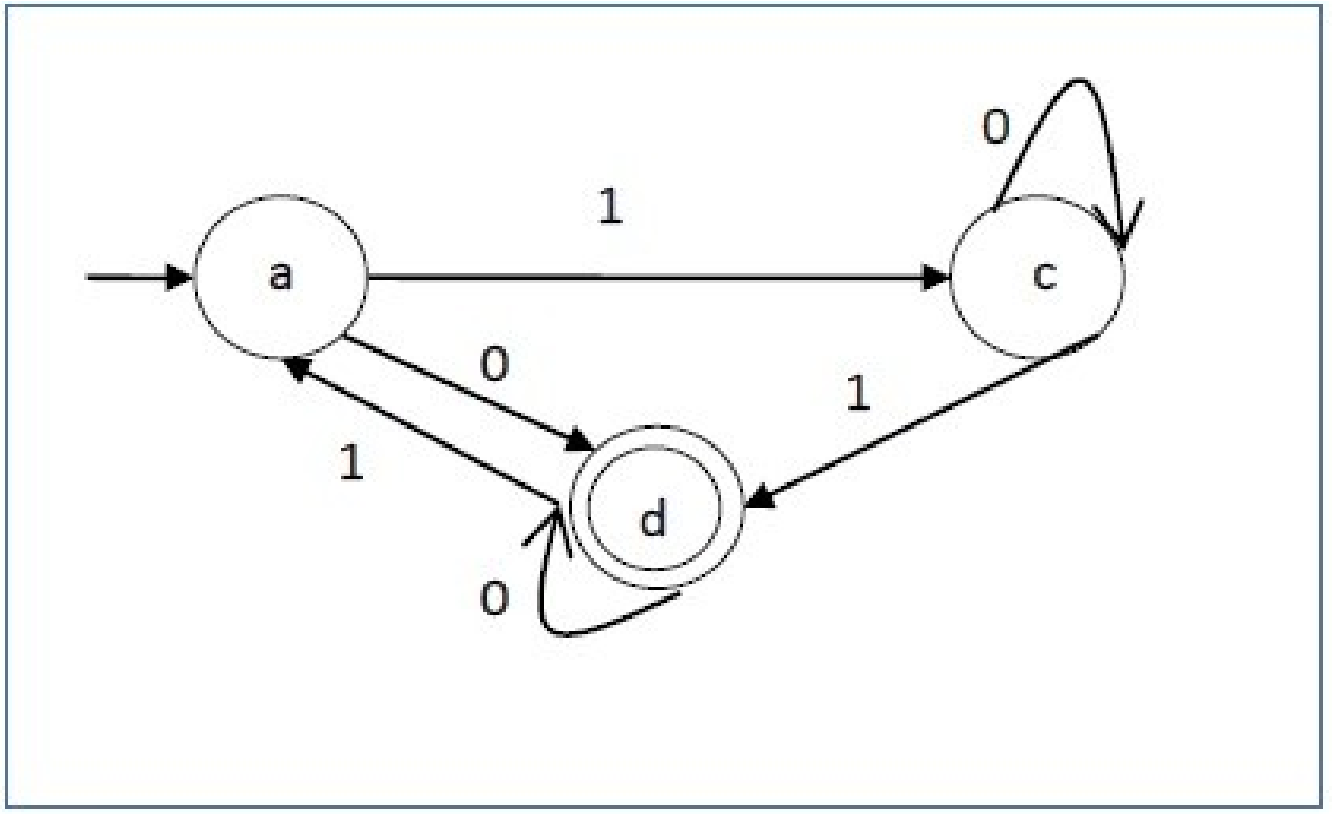
Language Acceptance

- Language L diterima oleh DFA/NDFA jika :
  - $\{S \mid S \in \sum^* \text{ and } \delta^*(q_0, S) \in F\}$

Ditolak jika

- String :
  - $\delta^*(q_0, S) \notin F$
- Language
  - $\{S \mid S \in \sum^* \text{ dan } \delta^*(q_0, S) \notin F\}$

Contoh 2



Accepted :

DFA: {0, 00, 11, 010, 101, ……}

Rejected :

DFA: {1, 011, 111, ...}

Konversi NDFA ke DFA

- NDFA : $X = (Q_x, \sum, \delta_x, q_0, F_x)$; $L(X)$
- DFA : $Y = (Q_y, \sum, \delta_y, q_0, F_y)$; $L(X)$
- $L(X) = L(Y)$
  - Input - NDFA
  - Output - DFA yang setara

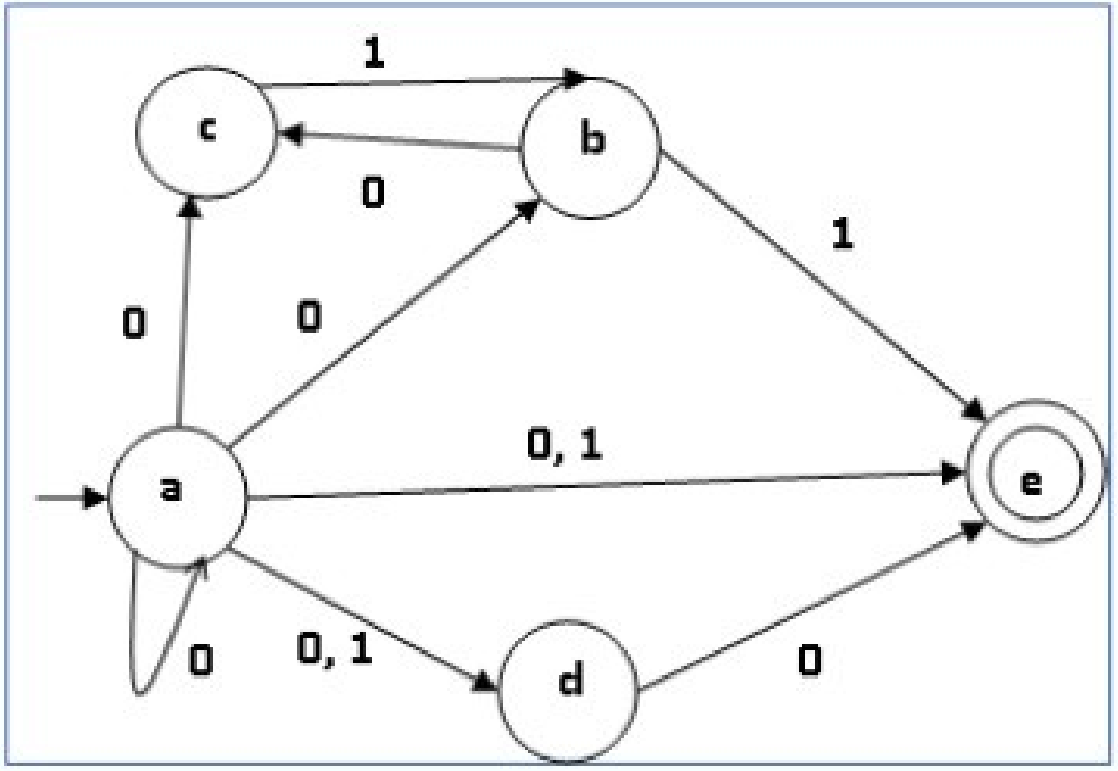
Langkah (1)

- Langkah 1 - Buat tabel state dari NDFA yang diberikan.
- Langkah 2 - Buat tabel state kosong di bawah abjad input yang mungkin untuk DFA yang setara.
- Langkah 3 - Tandai status awal DFA dengan q_0 (Sama seperti NDFA).

Langkah (2)

- Langkah 4 - Cari tahu kombinasi State $\{Q_0, Q_1, ..., Q_n\}$ untuk setiap alfabet masukan yang memungkinkan.
- Langkah 5 - Setiap kali state DFA baru dibuat di bawah kolom alfabet input, langkah 4 diterapkan lagi, jika tidak lanjutkan ke langkah 6.
- Langkah 6 - Status yang berisi salah satu status akhir NDFA adalah status akhir dari DFA yang setara.

Contoh 3



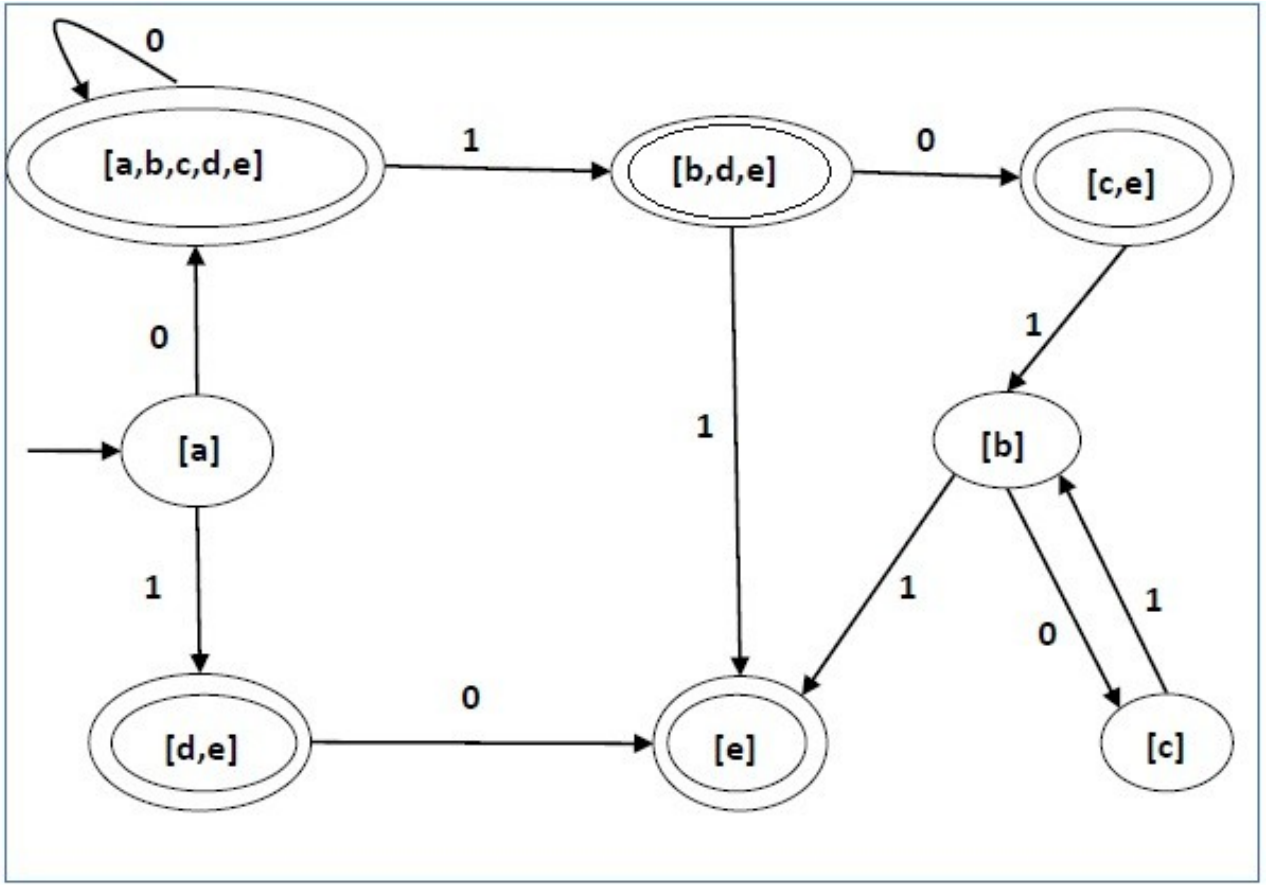
Contoh 3 : Transisi

q	δ(q,0)	δ(q,1)
a	{a,b,c,d,e}	{d,e}
b	{c}	{e}
c	∅	{b}
d	{e}	∅
e	∅	∅

Contoh 3 : Hasil Konversi

q	δ(q,0)	δ(q,1)
[a]	[a,b,c,d,e]	[d,e]
[a,b,c,d,e]	[a,b,c,d,e]	[b,d,e]
[d,e]	[e]	∅
[b,d,e]	[c,e]	[e]
[e]	∅	∅
[c, e]	∅	[b]
[b]	[c]	[e]
[c]	∅	[b]

Contoh 3 : Hasil Konversi dalam Graph



Minimalisasi DFA

- Teorema Myhill-Nerode
  - Input – DFA
  - Output – Minimized DFA

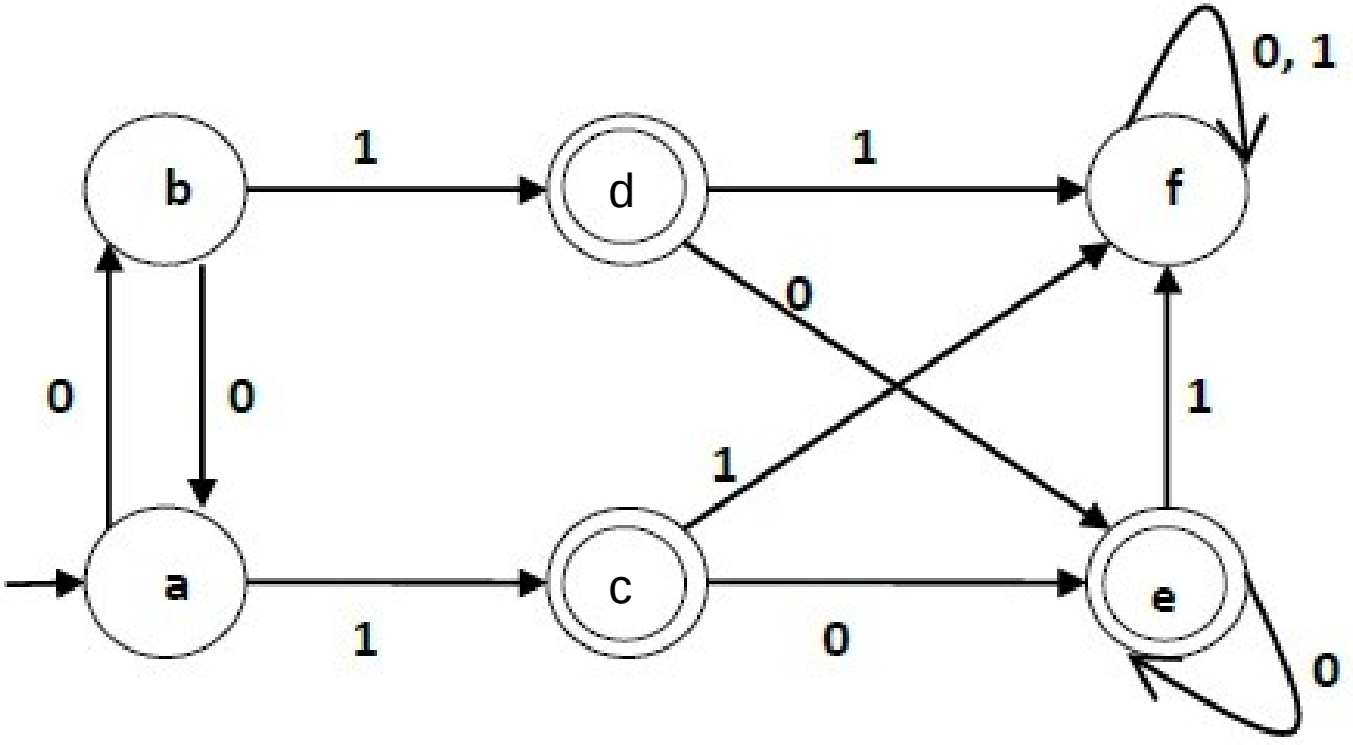
Teorema Myhill-Nerode (1)

- Step 1 − Membuat tabel semua hubungan antar state (Q_i, Q_j) tanpa mempertibangkan koneksi langsung [semua kemungkinan]
- Step 2 − Pertimbangkan semua hubungan state dalam DFA (Q_i, Q_j) dimana $Q_i \in F$ and $Q_j \notin F$ [F : himpunan final states] dan sebaliknya. (semua yang tidak berakhir di final)

Teorema Myhill-Nerode (2)

- Step 3 − Ulangi langkah ini sampai kita tidak dapat menandai lagi status -
  - Jika ada hubungan tak bertanda (Q_i, Q_j), tandai sebagai hubungan {δ (Q_i, A), δ (Q_i, A)} untuk beberapa alfabet masukan.
- Step 4 − Gabungkan semua pasangan tak bertanda (Q_i, Q_j) dan jadikan mereka satu keadaan di DFA yang dikurangi.

Contoh 4



Step1

Dibuat tabel untuk semua hubungan yang memungkinkan Q {a,b,c,d,f} dengan warna hijau untuk menandakan transitif (diusahakan kosong)

	a	b	c	d	e	f
a						
b						
c						
d						
e						
f						

Step 2

Semua hubungan antar Q {a,b,f} dan F {c,d,e} diberi centang

	a	b	c	d	e	f
a						
b						
c	✔	✔				
d	✔	✔				
e	✔	✔				
f			✔	✔	✔	

Step 3 (1)

- Jika kita memasukkan 1 untuk menyatakan 'a' dan 'f', itu sudah melewati 'c' dan 'f'.
  - (c, f) sudah ditandai, maka (a, f) juga ditandai.
- Selanjutnya 1 untuk menyatakan 'b' dan 'f'; itu akan melewati 'd' dan 'f'.
  - (d, f) sudah ditandai, maka (b, f) juga ditandai.

Step 3 (2)

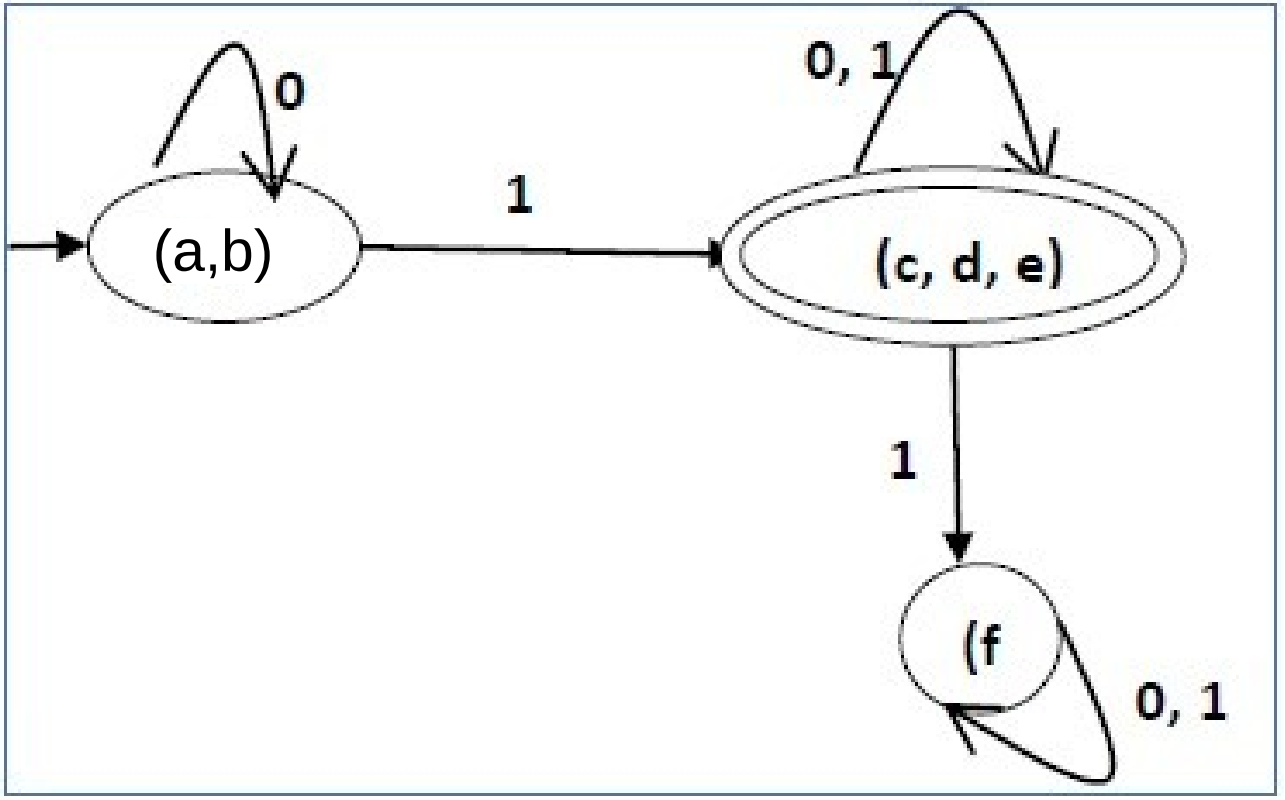
{a, b} {c, d} {c, e} {d, e} tidak ditandai lagi, {c, d} {c, e} {d, e} digabung menjadi {c, d, e}

	a	b	c	d	e	f
a						
b						
c	✔	✔				
d	✔	✔				
e	✔	✔				
f	✔	✔	✔	✔	✔	

Contoh 4 : Hasil Graph

Karena a dan b memiliki tujuan yang sama maka digabung menjadi {a, b}.

DFA menjadi minimalis yaitu hanya {f}, {a, b} and {c, d, e}



Equivalence Theorem

- Jika X dan Y adalah dua kondisi dalam DFA, kita dapat menggabungkan kedua kondisi ini menjadi {X, Y} jika keduanya tidak dapat dibedakan.
- Dua keadaan dapat dibedakan, jika ada setidaknya satu string S, sehingga salah satu dari δ (X, S) dan δ (Y, S) Accepted dan yang lain Rejected.
- Karenanya, DFA minimal jika memiliki state yang tidak dapat dibedakan.

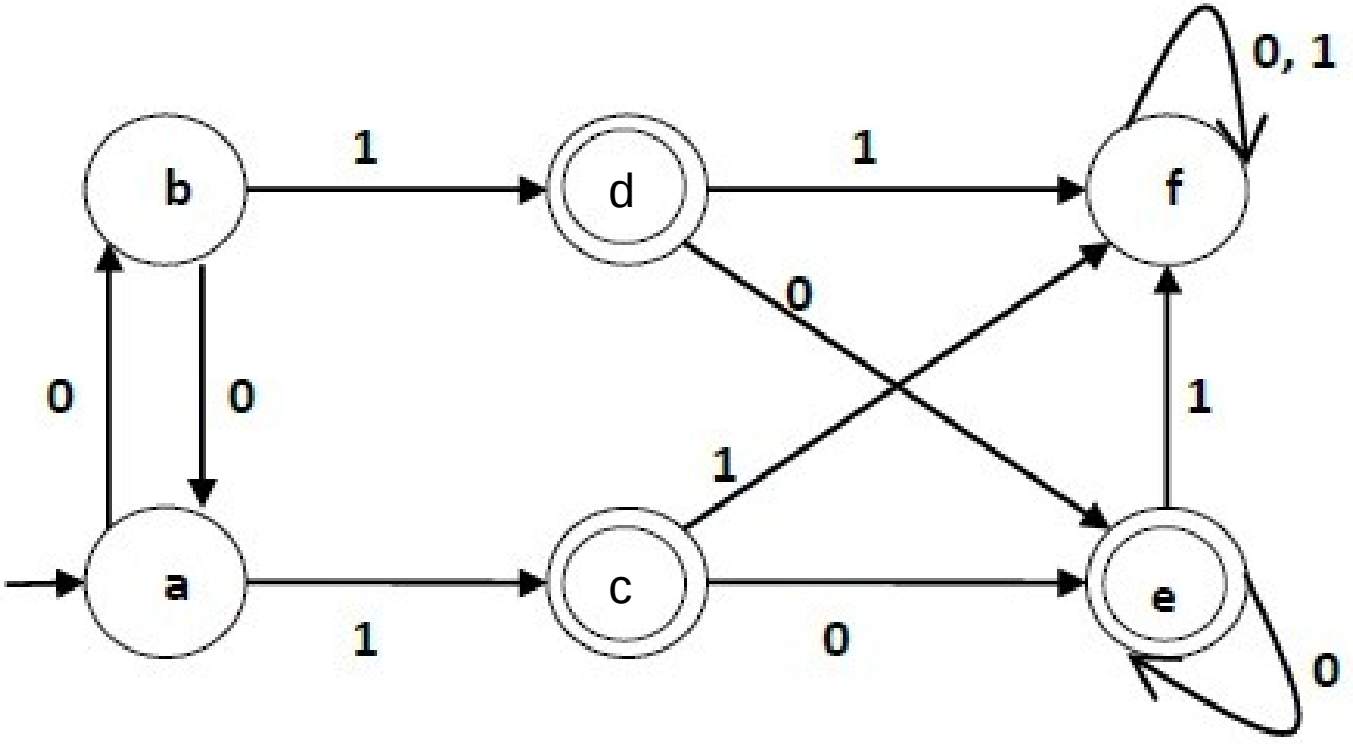
Step (1)

- Step 1 - Semua status Q dibagi menjadi dua partisi - final dan non-final dan dilambangkan dengan P_0.
  - Semua state dalam partisi adalah ekuivalen ke-0. Ambil pointer k dan inisialisasi dengan 0.
- Step 2 - Tambahkan k dengan 1. Untuk setiap partisi di P_k, bagi state dalam P_k menjadi dua partisi jika dapat dibedakan k.
  - Dua keadaan dalam partisi ini X dan Y dapat dibedakan k jika ada masukan S sehingga δ (X, S) dan δ (Y, S) adalah (k-1) - dapat dibedakan.

Step (2)

- Step 3 - Jika $P_k \neq P_k$-1, ulangi Langkah 2, jika tidak lanjutkan ke Langkah 4.
- Step 4 - Gabungkan set ekivalen ke-k dan jadikan sebagai status baru dari DFA yang dikurangi.

Contoh 5



Transisi

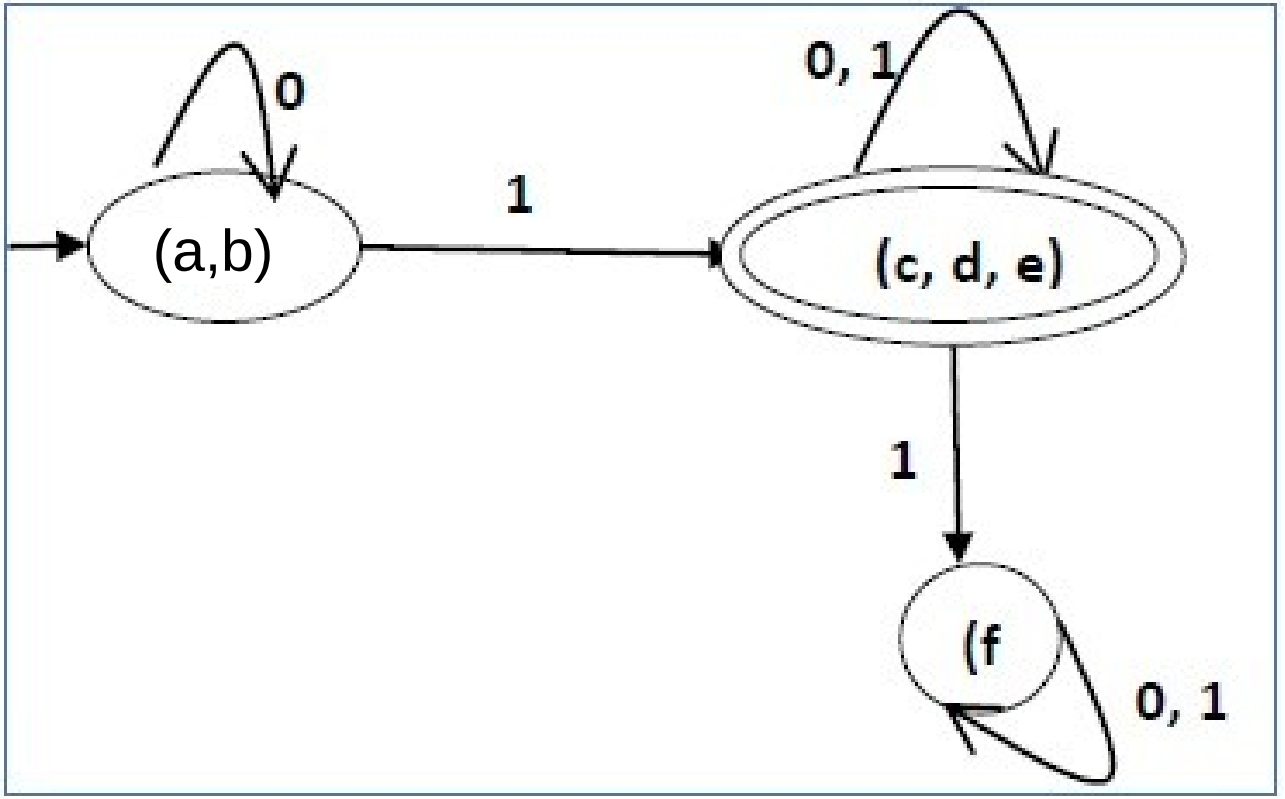
q	δ(q,0)	δ(q,1)
a	b	d
b	a	c
c	e	f
d	e	f
e	e	f
f	f	f

P_0 hingga P_k

- P_0 = {(c,d,e), (a,b,f)}
  - (c,d,e) = final, (a,b,f) = non-final
- P_1 = {(c,d,e), (a,b),(f)}
  - pada δ(q,0) (a,b) = berubah, (f) = tetap
- P_2 = {(c,d,e), (a,b),(f)}
  - pada δ(q,1) (a,b) = berubah lagi, (f) = tetap lagi

Contoh 5 : Hasil Graph

$P_1 = P_2$ sehingga menjadi pemisah untuk minimalisasi.



Contoh 5 : Hasil Transisi

Q	δ(q,0)	δ(q,1)
(a, b)	(a, b)	(c,d,e)
(c,d,e)	(c,d,e)	(f)
(f)	(f)	(f)