Finite State Machines

Kuliah Teori Bahasa dan Automata
Program Studi Ilmu Komputer
Fasilkom UI

Prepared by:

Rahmad Mahendra

Revised by:

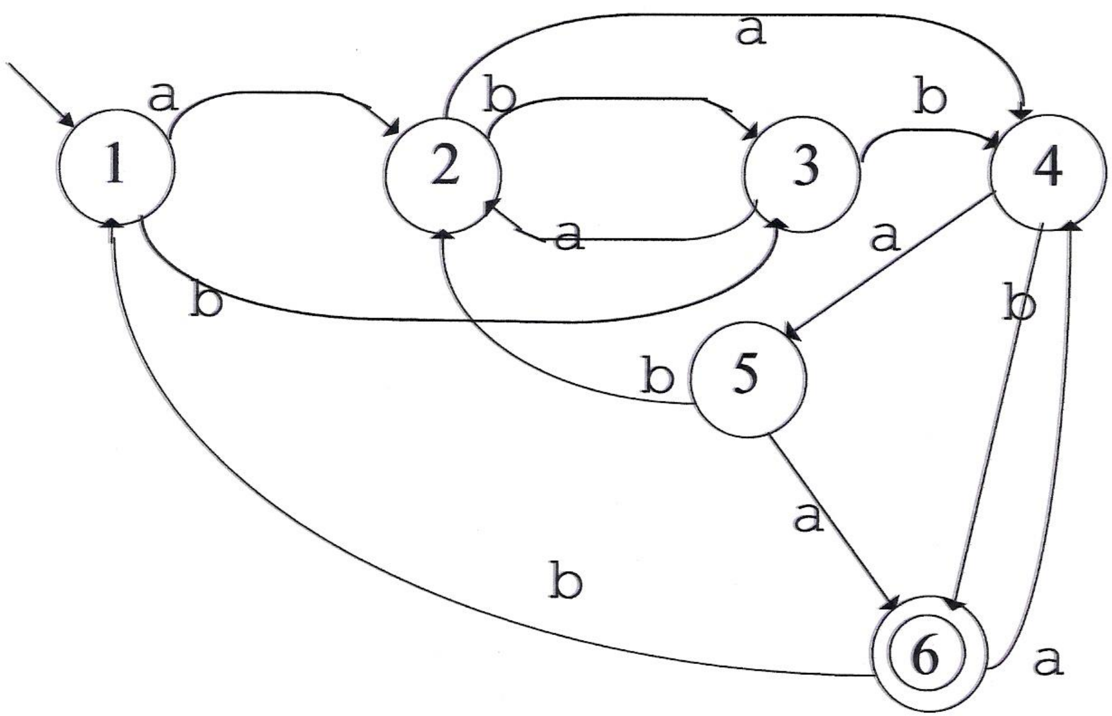
Maya Retno Ayu S

Motivasi – Minimalisasi DFSM

- Diberikan sebuah bahasa L, apakah ada sebuah DFSM minimal yang bisa menerima L?
- Jika ada suatu mesin yang minimal, apakah mesin tersebut unik?
- Diberikan suatu mesin DFSM M yang menerima beberapa bahasa L, dapatkah kita menentukan apakah M sudah minimal?
- Diberikan suatu mesin DFSM M, dapatkah kita membuat mesin M’ yang ekuivalen dengan M dan minimal?

Minimalisasi Status

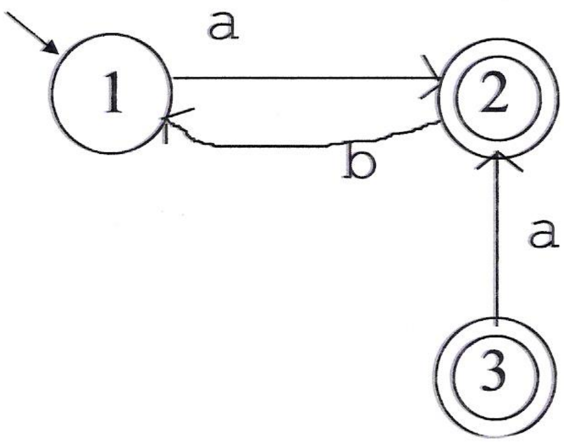
Perhatikan gambar di bawah ini:



Apakah mesin di atas sudah minimal?

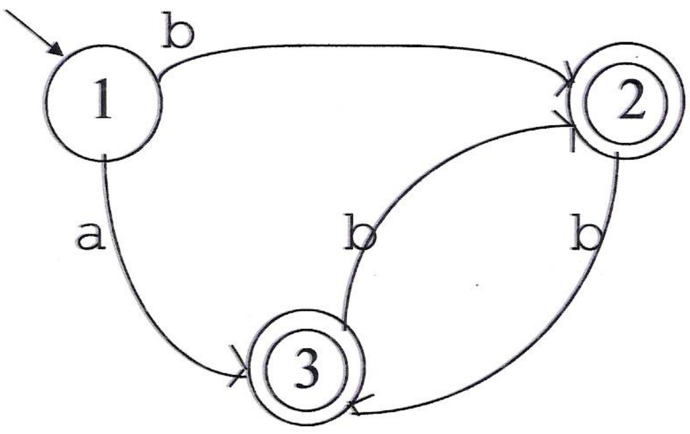
Minimalisasi Status

Hilangkah status yang unreachable:



Status 3

Hilangkan status yang *redundant*



Status 2 dan 3 redundant.

Minimalisasi FSM – Ekuivalensi String

- Penentuan kelas-kelas ekuivalen dari semua string yang dapat diterima oleh suatu bahasa
- Bagaimana menentukan apakah suatu string ekuivalen untuk suatu bahasa tertentu?
- Contoh:

(1)	a	b	a	b	a	b
(2)	b	a	a	b	a	b

Jika $L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* : |w| \text{ genap}\}$. Apakah (1) dan (2) ekuivalen?

Jika $L = \{w \in \{\mathtt{a}, \mathtt{b}\}^*$: setiap a segera diikuti oleh b$\}$. Apakah (1) dan (2) ekuivalen?

Distinguishability terhadap L

- String x dan y disebut ekuivalen (*indistinguishable*) jika tidak dapat dibedakan terhadap bahasa L, ditulis $x \approx_L y$, jika dan hanya jika:

  $$\forall z \in \Sigma^* \ (xz, yz \in L \text{ atau } xz, yz \notin L)$$

- x dan y dapat dibedakan (*distinguishable*) terhadap L, jika dan hanya tidak *indistingusihable*.
- Contoh: $\Sigma = \{a, b\}$ dan $L = \{w \in \Sigma^* : |w| \text{ genap}\}$
  - a $\approx_L$ aaa (*indistinguishable*)

    Proof by case:

    Ambil z = string Σ^* panjang ganjil, az $\in L$ dan aaaz $\in L$.

    Ambil z = string Σ^* panjang genap, az $\notin L$ dan aaaz $\notin L$
  - a dan aa *distinguishable* terhadap L karena misalkan jika z = b, maka ab $\in L$ sedangkan aab $\notin L$

Distinguishability terhadap L

- Relasi $\approx_L$ adalah relasi ekuivalen
  - Refleksif: $\forall x \in \Sigma^* \, (x \approx_L x)$
  - Simetri: $\forall x,y \in \Sigma^* \, (x \approx_L y \rightarrow y \approx_L x)$
  - Transitif: $\forall x,y,z \in \Sigma^* \, (((x \approx_L y) \land (y \approx_L z)) \rightarrow (x \approx_L z))$
- Karena $\approx_L$ adalah relasi ekuivalen, kelas-kelas ekuivalen membentuk partisi himpunan Σ^*, sehingga
  - Tidak ada kelas ekuivalen $\approx_L$ yang merupakan himpunan kosong
  - Setiap string dalam Σ^* hanya berada pada tepat satu kelas ekuivalen dari $\approx_L$

Menentukan $\approx_L$

- Setiap kelas ekuivalen hanya dapat mengandung string yang termasuk dalam L, atau string yang bukan L saja. Jika $x \in L$, maka $x\varepsilon \in L$. Jika $y \notin L$, maka $y\varepsilon \notin L$. Jadi, x dan y *distinguishable* oleh ε
- Jika ada sejumlah string yang dalam proses komputasi oleh DFSM masuk ke *dead state* (string-string tersebut tidak termasuk L), maka ada satu kelas ekuivalen $\approx_L$ yang berkorespondensi dengan *dead state*.
- Kelas ekuivalen yang mengandung ε berkorespondensi *start state* dari mesin minimal yang menerima (*accept*) L
- Mungkin ada lebih satu kelas ekuivalen yang mengandung string yang termasuk dalam L

Contoh-1

- $L = \{w \in \{a, b\}^* :$ tidak ada karakter bersisian yang identik$\}$

Kelas-kelas ekuivalen $\approx_L$

- [1] $[\varepsilon]$
- [2] [a, aba, ababa]

  Semua string tidak kosong yang berakhir dengan a dan tidak mengandung karakter bersisian yang identik.
- [3] [b, ab, bab, abab]

  Semua string tidak kosong yang berakhir dengan b dan tidak mengandung karakter bersisian yang identik.
- [4] [aa, abaa, ababb]

  Semua string tidak kosong yang mengandung karakter bersisian yang identik.

Coba Sendiri

Tentukan kelas-kelas ekuivalen dari bahasa berikut

- $L_1 = \{w \in \{a, b\}^* :$ setiap karakter a diikuti oleh b$\}$
- $L_2 = \{w \in \{0, 1\}^* :$ tidak mengandung substring 010$\}$
- $L_3 = \{w \in \{a, b\}^* : w = a^n b^n, n \geq 0\}$

Apakah $\approx_L$ selalu memiliki berhingga (*finite*) jumlah kelas ekuivalen?

Jumlah State pada DFSM

- Teorema

  L bahasa reguler dan $M = (K, \Sigma, \delta, s, A)$ sebuah DFSM yang menerima (*accept*) bahasa L.

  Jumlah *state* pada $M \geq$ jumlah kelas ekuivalen $\approx_L$

- Pembuktian:

  Jika jumlah *state* pada $M <$ jumlah kelas ekuivalen $\approx_L$, maka berdasarkan prinsip kandang burung dara (*pigeonhole principle*) ada sekurangnya satu *state* yang mengandung string dari sekurangnya dua kelas ekuivalen $\approx_L$ yang berbeda. Karakteristik M pada string ini seharusnya identik, kontradiktif dengan asumsi bahwa string berasal dari kelas ekuivalen berbeda.

Minimal DFSM yang Unik

- Untuk setiap bahasa reguler terdapat DFSM minimal yang unik

- Teorema

  L adalah bahasa reguler yang dibentuk dari alfabet Σ maka terdapat sebuah DFSM *M* yang menerima *L* dengan jumlah *state* sebanyak *n*, di mana *n* adalah jumlah kelas ekuivalen $\approx_L$.

  DFSM lainnya yang menerima *L* harus ekuivalen dengan *M* atau memiliki jumlah *state* $> n$

Minimal DFSM yang Unik

- Untuk setiap bahasa reguler terdapat DFSM minimal yang unik

- Pembuktian

  Konstruksi DFSM $M = (K, \Sigma, \delta, s, A)$ di mana:
  - K mengandung *n state*, masing-masing satu untuk setiap kelas ekuivalen $\approx_L$
  - $s = [\varepsilon]$, kelas ekuivalen string ε terhadap $\approx_L$
  - $A = \{[x] : x \in L\}$
  - $\delta([x], a) = [xa]$

Contoh-1

$L = \{w \in \{a, b\}^* :$ tidak ada karakter bersisian yang identik$\}$

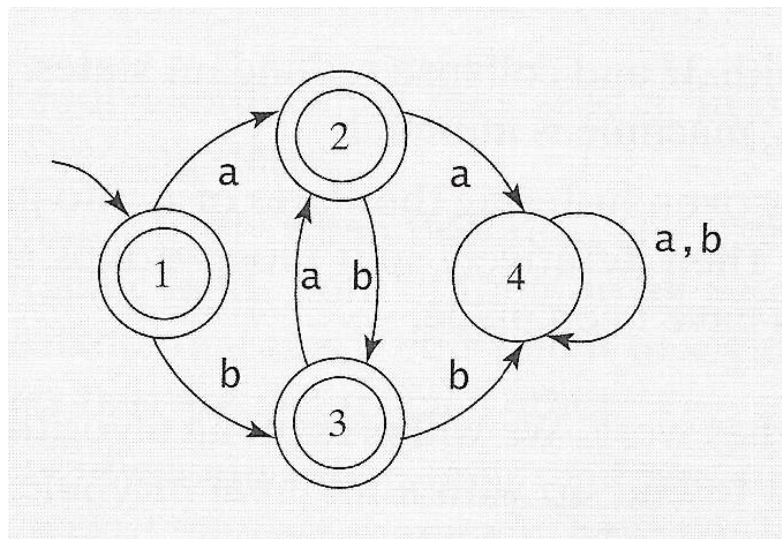
- *Start state* $[\varepsilon] = [1]$
- *Accepting state* adalah seluruh kelas ekuivalen yang beranggotakan string pada L, yaitu [1], [2], [3]
- $\delta([x], a) = [xa]$

  Kelas ekuivalen [1] mengandung string $\boldsymbol{\varepsilon}$. Jika $\boldsymbol{\varepsilon}$ diikuti oleh karakter a menghasilkan string **a** yang terdapat pada kelas ekuivalen [2], maka didefinisikan fungsi transisi dari [1] ke [2] dengan label a.

  Kelas ekuivalen [2] mengandung string **a**. Jika a diikuti oleh karakter **b** menghasilkan string **ab** yang terdapat pada kelas ekuivalen [3], maka didefinisikan fungsi transisi dari [2] ke [3] dengan label b.

Contoh-1

$L = \{w \in \{a, b\}^* :$ tidak ada karakter bersisian yang identik$\}$



Teorema Myhill-Nerode

- Teorema

  Sebuah bahasa L adalah reguler jika dan hanya jika jumlah kelas ekuivalen $\approx_L$ berhingga (*finite*)

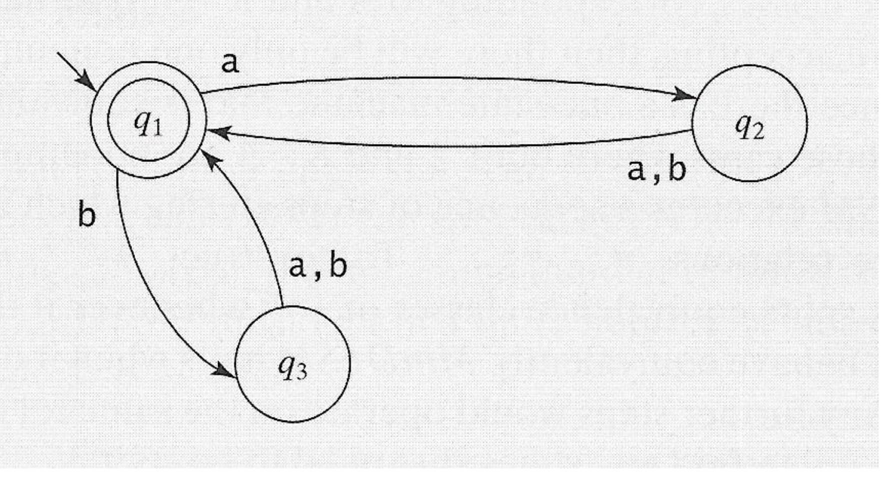
- Pembuktian:

  Coba sendiri!

  Petunjuk: buktikan dua pernyataan implikasi berikut
  - L reguler $\rightarrow$ jumlah kelas ekuivalen $\approx_L$ berhingga
  - jumlah kelas ekuivalen $\approx_L$ berhingga $\rightarrow$ L reguler

DFSM yang Tidak Minimal

- $L = \{w \in \{a, b\}^* : |w| \text{ genap}\}$



- Pada mesin di atas, *state* $q_2 \equiv q_3$

Minimisasi DFSM

- Misalkan ada sebuah DFSM *M* yang menerima *L*, ada dua pendekatan untuk memperoleh DFSM minimal
  1. Mulai dengan *M* dan hilangkan *state* yang redundan. Lakukan satu persatu sehingga ditemukan mesin minimal.
  2. Mulai dengan cara mengelompokkan *state L* menjadi dua grup, *accepting* dan *non-accepting*. Secara iteratif, bagi masing-masing grup ini menjadi beberapa *state*.

Algoritma Minimisasi DFSM

Input: DFSM $M = (K, \Sigma, \delta, s, A)$

1. Kelompokkan *state-state* menjadi dua: *accepting* dan *non-accepting*

   $classes = \{A, K - A\}$

2. Ulangi langkah berikut sampai tidak ada perubahan pada *classes*

   a. $newclasses = \varnothing$

   b. Untuk setiap kelas ekuivalen *e* pada *classes*, jika *e* mengandung lebih dari satu *state*, tinjau apakah kelompok *state* perlu dipecah

   ...

Algoritma Minimisasi DFSM

2. ...
   b. ...

   Untuk setiap *state* q pada e, do:

   Untuk setiap karakter c pada Σ, do:

   Tentukan elemen kelas mana q menuju jika karakter c dibaca

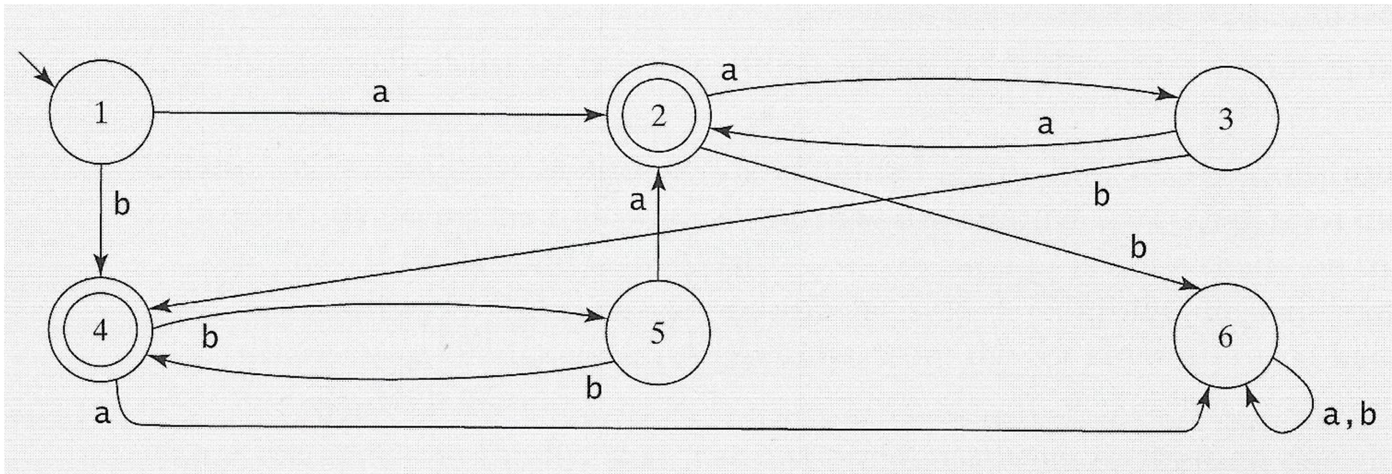
   Jika ada dua *state* p dan q sehingga ada karakter c. Ketika c dibaca oleh mesin, p menuju ke salahsatu elemen *classes* dan q menuju elemen *classes* lainnya, maka p dan q harus dipisah. Tambahkan *classes* pada *newclasses*

   Jika tidak ada *state* dengan karakteristik berbeda, pemisahan *state* tidak diperlukan. Tambahkan e pada *newclasses*.

   c. *classes* = *newsclasses*

Output: $M' = (classes, \; \Sigma, \; \delta, \; [s_M], \{[q\text{: elemen } q \text{ pada } A_M]\})$
di mana jika $\delta_M(q,c) = p$, maka $\delta_{M'}([q],c) = [p]$

Contoh-2



- $\Sigma = \{a, b\}$

- Tentukan *DFSM* minimal yang juga menerima bahasa *L* yang diterima oleh *DFSM* di atas

Contoh-2

- *classes* = {[2, 4], [1, 3, 5, 6]}
- Step 1

  ((2,a), [1, 3, 5, 6]) ((4,a), [1, 3, 5, 6])

  ((2,b), [1, 3, 5, 6]) ((4,b), [1, 3, 5, 6])

  Tidak diperlukan pemisahan *state*

  ((1,a), [2, 4]) ((3,a), [2, 4]) ((5,a), [2, 4])

  ((1,b), [2, 4]) ((3,b), [2, 4]) ((5,b), [2, 4])

  ((6,a), [1, 3, 5, 6]) ((6,b), [1, 3, 5, 6])

  State 6 perlu dipisah dari [1, 3, 5]

Contoh-2

- *classes* = {[2, 4], [1, 3, 5], [6]}
- Step 2

((2,a), [1, 3, 5]) ((4,a), [6])

((2,b), [6]) ((4,b), [1, 3, 5])

State 2 dipisah dari kelas *state* 4

((1,a), [2, 4]) ((3,a), [2, 4]) ((5,a), [2, 4])

((1,b), [2, 4]) ((3,b), [2, 4]) ((5,b), [2, 4])

Tidak diperlukan pemisahan *state*

Contoh-2

- *Classes* = {[2], [4], [1, 3, 5], [6]}
- Step 3

((1,a), [2]) ((3,a), [2]) ((5,a), [2])

((1,b), [4]) ((3,b), [4]) ((5,b), [4])

Tidak diperlukan pemisahan *state*

DFSM minimal