MODUL 7: MINIMISASI FA

Dalam pembahasan sebelumnya untuk setiap mesin FA (baik NFA, NFA-Λ, maupun FA) pasti ada suatu bahasa regular yang dapat ia terima dan sebaliknya untuk setiap bahasa regular pasti ada mesin FA yang dapat menerimanya. Hal ini terbukti dengan metoda/algoritma untuk menkonversi ekspresi regular menjadi NFA-Λ dan dari NFA-Λ bisa dikonversi menjadi NFA, dan kemudian menjadi FA. Demikian pula dari suatu FA ekspresi regular dari bahasa ybs dapat ditemukan.

FA sesuai dengan namanya merupakan mesin dengan jumlah status terbatas. Yang menjadi pertanyaan, seberapa terbatasnya ia? Kalau terbatas tentu ada jumlah minimal status yang diperlukan untuk membuat suatu FA M yang menerima bahasa L.

Dalam pembahasan modul ini kita akan melihat bagaimana caranya menemukan FA M_1 dengan jumlah status minimal dari FA M yang diberikan. Tetapi untuk membangun pemahaman konseptual maka sebelumnya akan dibahas pula landasan teoritis yang diperlukan. Pertama adalah mengenai keberbedaan string, kemudian mengenai kelas ekivalensi string, dan kemudian algoritma penemuan FA dengan jumlah status minimal tersebut.

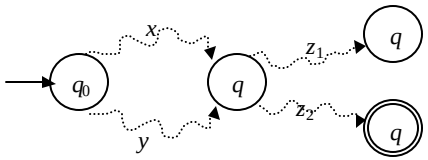
Keberbedaan String

Definisi. Untuk bahasa $L \subseteq \Sigma^*$, dua string x dan $y \in \Sigma^*$ dikatakan terbedakan (*distinguishable*) terhadap bahasa L jika terdapat string $z \in \Sigma^*$ sehingga tepatnya salah satu string xz atau yz berada dalam L. String z dikatakan sebagai pembeda x dari y terhadap L.

Dengan definisi ini bisa dikatakan pula x dan y takterbedakan (*indistinguishable*) terhadap L bila string z tersebut tidak pernah ada; jadi dalam hal ini baik xz dan yz ada dalam L (atau tidak ada dalam L).

Contoh: Terhadap L = {x ∈ {0, 1}* | *x* berakhiran 10} string 01011 dan 100 adalah terbedakan karena dengan $z = 0$, maka $01011z \in L$ dan $100z \notin L$. Sementara string 0 dan 100 tak terbedakan terhadap L setiap z menyebabkan 0z dan 100z berada dalam L (misalnya z=10) atau tidak berada dalam L (misalnya z = 11).

Bagaimana masalah keberbedaan string ini dikaitkan dengan mesin FA M yang menerima bahasa L ybs? Dalam hal ini apabila x dan y menyebabkan mesin M berhenti di suatu status yang sama maka kedua string tersebut tak terbedakan karena string z apapun dalam xz dan yz juga akan membawa string ke status yang sama. Hal ini dinyatakan dalam lemma sebagai berikut.



Lemma Ketakterbedakan. Misalnya $L \subseteq \Sigma^*$ dan $M = (Q, \Sigma, q_0, \delta, A)$ adalah FA yang menerima L. Jika x dan y dua buah string dalam Σ^* yang mana $\delta^*(q_0, x) = \delta^*(q_0, y)$, maka x dan y dikatakan takterbedakan (*indistinguishable*) terhadap bahasa L.

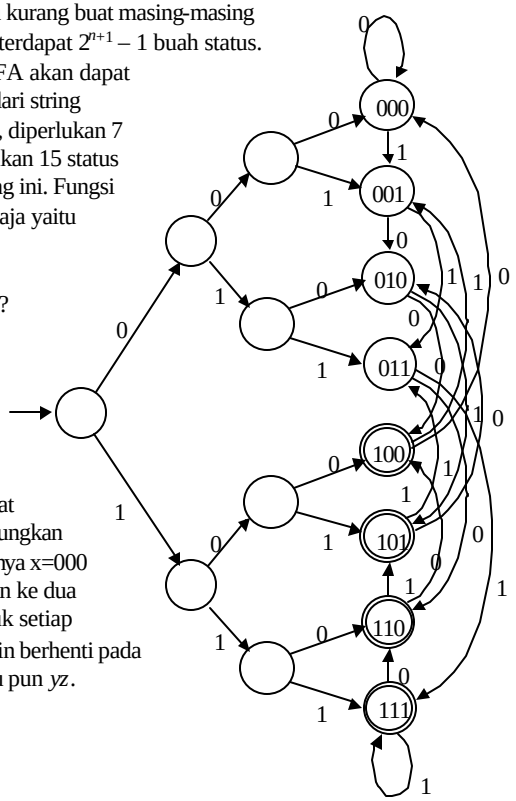
Jadi, jika ada dua string yang terbedakan terhadap bahasa L maka sekurangnya ada dua status yang berbeda dalam mesin FA yang menerima L. Bagaimana jika ternyata ditemukan n string yang satu sama lainnya terbedakan terhadap L? Sekurangnya ada n status yang berbeda dalam mesin FA tersebut. Hal ini dinyatakan dalam teorema berikut.

Teorema Jumlah Minimal Status. Misalnya $L \subseteq \Sigma^*$ dan terdapat n buah string yang terbedakan (n bilangan bulat positif) berkenaan dengan L. Maka, tidak akan ada FA untuk mengenali L yang memiliki jumlah status kurang dari n.

Bukti teorema ini bisa dijelaskan dengan "Pigeonhole Principle". Jika kurang dari n maka ada beberapa string yang menyebabkan mesin berhenti di status yang sama dan ini kontradiksi dengan lemma sebelumnya. Teorema ini mengisyaratkan adanya batas bawah dari kebutuhan memory/storage dalam setiap FA dalam pengenalannya terhadap bahasa yang bersangkutan.

Contoh: Untuk $n \geq 1$, $L_n = \{x \in \{0, 1\}^* \mid |x| \geq n$ dan simbol ke-n dari kanan adalah 1$\}$. Ekspresi regular bahasa ini adalah $(0+1)^*1(0+1)^n$.

Secara langsung FA dapat dibentuk sebagai berikut: untuk setiap kemungkinan substring dengan panjang n atau kurang buat masing-masing satu status berbeda. Untuk itu terdapat $2^{n+1} - 1$ buah status. Dengan status-status tersebut FA akan dapat "mengingat" n simbol terakhir dari string masukan. Misalnya untuk n =2, diperlukan 7 status, dan untuk n = 3, diperlukan 15 status (terlihat pada gambar di samping ini. Fungsi transisinya berpola sederhana saja yaitu $\delta(abc, d) = bdc$.



Sudah minimalkah FA tersebut?

Sejauh ini pembahasan baru pada implikasi dari jumlah string yang saling terbedakan pada jumlah status minimal. Dalam contoh ini mungkin saja terdapat sejumlah status yang bisa digabungkan karena ketakterbedakan. Misalnya x=000 dan y=00 akan membawa mesin ke dua status yang berbeda tetapi untuk setiap string $z \neq \Lambda$ menyebabkan mesin berhenti pada status yang sama karena xz atau pun yz.

Berikut ini contoh yang mana jumlah status tidak dapat diperkirakan (tak terbatas).

Contoh: Σ adalah alfabet dan bahasa Palindrom (disingkat *Pal*) pada Σ adalah tedefinisi sebagai berikut:

1. $\Lambda \in Pal$
2. untuk semua simbol $a \in \Sigma$, maka string satu simbol $a \in Pal$
3. untuk semua string x $\in Pal$, dan untuk semua simbol $a \in \Sigma$, maka string $axa \in Pal$
4. Tidak ada string $x \in Pal$, yang tidak diperoleh dari aturan 1,2, dan 3 di atas.

Berapa jumlah status yang diperlukan untuk mengenali bahwa string $x \in Pal$? Apabila kita scan dari kiri ke kanan sesuai dengan cara bekerjanya suatu FA maka akan begitu tak terhingganya jumlah simbol yang perlu kita "ingat". Berbeda dengan contoh sebelumnya yang dapat dengan sederhana menggunakan sejumlah terbatas status. Dalam konteks keterbedaan maka jumlah string yang takterbedakan adalah takhingga. Nanti akan kita ketahui bahwa Pal bukanlah suatu bahasa regular (disebut nonregular) yang juga berarti ia tidak dapat dikenali oleh suatu FA.

Jadi sifat bahwa jumlah status dalam suatu FA akan bergantung berapa banyak pasangan string yang dapat terbedakan terhadap L, merupakan sifat khas dari bahasa regular, yang berati juga tidak dimiliki oleh bahasa-bahasa non regular. Untuk suatu bahasa non-regular L jumlah pasangan string yang terbedakan terhadap L adalah tak berhingga.

Relasi Ekivalensi Tak Terbedakan

Definisi: L adalah suatu bahasa dalam Σ^*. Relasi "Takterbedakan" I_L pada Σ^* terdefinisi sebagai berikut. Dua string x dan y dinyatakan dalam relasi $x\ I_L\ y$ jika dan hanya jika x dan y tak terbedakan terhadap bahasa L. Dengan kata lain, akan terdapat string $z \in \Sigma^*$ berlaku xz dan yz keduanya dalam L, atau keduanya tidak berada dalam L.

Partisi Kelas Ekivalensi

Relasi ini merupakan relasi ekivalensi karena relasi ini refleksif, simetris, dan transitif. Sementara himpunan sejumlah string yang saling memiliki relasi ekivalensi umumnya disebut sebagai satu kelas ekivalensi. Jadi relasi ekivalensi keterbedaan I_L ini mempartisi Σ^* ke sejumlah kelas ekivalensi. Apakah jumlah kelas ini terbatas?

Jika ternyata ada n kelas ekivalensi maka pasti juga terdapat n string yang terbedakan yang diambil dari masing-masing kelas. Apakah n ini memang terbatas?

Jumlah kelas ekivalensi dalam bahasa regular adalah berhingga (**Teorema Myhill-Nerode**). Untuk membuktikan teorema ini maka kita mencoba melakukan partisi lain selain partisi kelas ekivalensi tersebut.

Partisi Menurut Status Berhenti

Untuk suatu bahasa regular L dan suatu FA $M = (Q, \Sigma, q_0, A, d)$ yang dapat mengenali L, dengan $q \in Q$, kita definisikan

$L_q = \{x \in \Sigma^* \mid d^*(q_0, x) = q\}$.

Jadi L_q adalah himpunan dari string-string yang membuat M berhenti di status q. Oleh sebab itu, partisi yang terjadi disebut partisi menurut status berhenti. Jika dalam FA terdapat n status maka akan terbentuk n buah L_q. Karena jumlah status dalam FA berhingga maka jumlah partisi ini pun berhingga.

Sesuai dengan Lemma Ketakterbedakan, jika dua string x dan $y \in L_q$ maka x dan y takterbedakan yang juga berarti $x\, I_L\, y$. Tetapi implikasi ini belum tentu berlaku sebaliknya seperti yang telah dilihat pada pembahasan contoh bahasa $(0+1)^*1(0+1)^n$. Dengan demikian dapat dikatakan partisi L_q ini lebih "halus" dari partisi kelas ekivalensi karena setiap L_q merupakan subset dari suatu kelas ekivalensi.

Bilakah kedua partisi ini tepat sama? Ketika FA sudah merupakan yang paling sederhana! Notasi berikut ini untuk menjelaskan mendapatkan FA yang paling sederhana berdasarkan relasi ekivalensi.

Notasi $[z]$

Definisi: Untuk suatu string z, maka $[z]$ menyatakan kelas ekivalensi yang berisikan z.

Dengan notasi ini untuk dua string berlainan x dan y, jika dituliskan $[x] = [y]$ (baca: kelas ekivalensi yang berisi x sama dengan kelas ekivalensi yang berisi y), maka artinya juga $x\, I_L\, y$. Demikian pula jika $x \in L_p$ dan $xa \in L_q$ (yang artinya $d(p, a) = q$), maka $d([x], a) = [xa]$.

Lemma Invariansi Kanan: setiap $x, y \in \Sigma^*$ dan setiap $a \in \Sigma$, bila $x\, I_L\, y$, maka $xa\, I_L\, ya$. Jadi jika $[x] = [y]$ maka $[xa] = [ya]$.

Mengingat $d([x], a) = [xa]$ dan $d([y], a) = [ya]$, maka kalau $[x] = [y]$ jelaslah bahwa $[xa] = [ya]$.

Teorema Minimisasi FA dengan Kelas Ekivalensi: $L \subseteq \Sigma^*$, dan Q_L himpunan kelas-kelas ekivalensi dari relasi I_L pada Σ^*. Bila Q_L adalah himpunan berhingga, maka $M_L = (Q_L, \Sigma, q_0, A_L, d)$ merupakan suatu FA yang menerima L, dimana

- $q_0 = [\Lambda]$,
- $A_L = \{q \in Q_L \mid q \cap L \neq \varnothing\}$,
- dan $d : Q_L \times \Sigma \rightarrow Q_L$ terdefinisi sebagai $d([x], a) = [xa]$
- selain itu M_L memiliki jumlah status yang paling sedikit dari antara FA yang mengenali L.

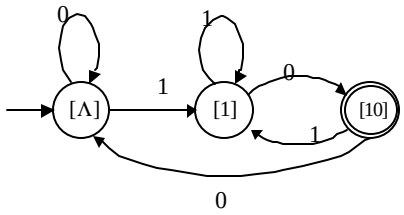
Teorema ini menghasilkan suatu FA dengan jumlah status minimal. Namun persyaratannya, kita perlu mengidentifikasi kelas-kelas ekivalensi dari L.

Contoh: L = {x ∈ {0, 1}* | x memiliki akhiran 10}.
String Λ, 1, dan 10 adalah saling terbedakan terhadap L. Untuk setiap string y yang lain, maka y memiliki relasi ekivalen ketakterbedakan dengan salah satu dari ketiga string itu. Jadi untuk L hanya ada ketiga kelas ekivalensi ini yaitu [Λ], [1], dan [10].. Kemudian dari

$d([\Lambda], 0) = [\Lambda]$ $d([\Lambda], 1) = [1]$
$d([1], 0) = [10]$ $d([1], 1) = [1]$

$d([10], 0) = [\Lambda]$ $d([10], 1) = [1]$

Maka diperoleh Fa sebagai berikut.



Pencarian kelas ekivalensi ini dalam beberapa kasus tidaklah sederhana. Secara umum kita perlu suatu metoda yang algoritmis. Algoritma ini berangkat dari partisi menurut status berhenti kemudian melakukan penggabungan-penggabungan atas sejumlah kesamaan sifat yang ditemukan.

Lemma: untuk $p, q \in Q$, diketahui bahwa L_p dan L_q berada dalam kelas ekivalensi yang berbeda jika dan hanya jika terdapat string $z \in \Sigma^*$ sehingga tepat hanya satu dari $d^*(p, z)$ atau $d^*(q, z)$ berada di A.

Bukti (dari kiri ke kanan): diketahui L_p dan L_q berada dalam kelas ekivalensi yang berbeda. Misalkan $x \in L_p$ dan $y \in L_q$ yang berarti x dan y berada dalah kelas ekivalensi yang berbeda. Maka terdapat z yang dapat memberbedakan x dan y terhadap L. Dengan demikian kita dapatkan:

$d^*(p, z) = d^*(d^*(q_0, x), z) = d^*(q_0, xz)$
$d^*(q, z) = d^*(d^*(q_0, y), z) = d^*(q_0, yz)$

yang karena z memberbedakan x dan y terhadap L maka salah satunya ($d(q_0, xa)$ atau $d(q_0, yz)$) akan berada di A.

Bukti (dari kanan ke kiri): diketahui salah satu $d^*(p, z)$ dan $d^*(q, z)$ berada di A maka maka terdapat $x \in L_p$ dan $y \in L_q$ sehingga string z memberbedakan x dan y terhadap L. Jadi L_p dan L_q adalah subset-subset dari kelas ekivalensi yang berbeda.

Jadi ide algoritma ini adalah menemukan setiap pasangan status (p, q) yang mana L_p dan L_q berada di dua kelas ekivalensi yang berbeda. Dari semua kemungkinan pasangan maka pasangan yang tersisa (p, q) merupakan pasangan dengan L_p dan L_q berada di dalam kelas ekivalensi yang sama.

Pertama-tama yang paling mudah untuk mengetahui pasangan (p, q) dimana L_p dan L_q ada di dua kelas ekivalensi yang berbeda adalah jika tepat hanya satu dari p atau q (jadi string $z = \Lambda$) merupakan status menerima.

Untuk status-status lain, misalnya pasangan (r, s) dapat diketahui apakah L_r dan L_s berada dalam dua kelas ekivalensi yang berbeda jika terdapat $a \in \Sigma$ sehingga $d(r, a) = p$ dan $d(s, a) = q$ serta L_p dan L_q berada dalam dua kelas ekivalensi yang berbeda.

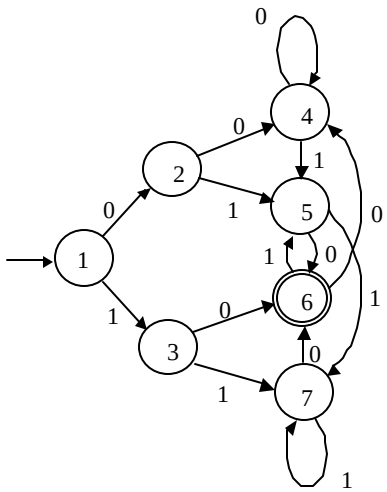
Jadi, algoritma memulai dari yang paling mudah tersebut kemudian dalam iterasi menemukan pasangan-pasangan berikutnya berdasar penemuan pasangan sebelumnya hingga tidak ada lagi pasangan baru yang ditemukan. Setelah iterasi, pasangan yang tidak ditemukan

Algoritma:

- List semua kemungkinan pasangan status (p, q) (pasangan tak berurutan)
- Lakukan dalam sejumlah pass.
  - Pada pass pertama tandai pasangan yang tepat hanya satu statusnya $\in A$
  - Pada setiap pass berikutnya, tandai pasangan (r, s) bila terdapat $a \in \Sigma$ menyebabkan $d(r, a) = p$ dan $d(s, a) = q$ sementara (p, q) sudah bertanda.
  - Pass selesai jika sudah tidak ada lagi pasangan yang ditandai.
- Pasangan (p, q) yang telah bertanda adalah status-status sehingga L_p dan L_q adalah subset-subset dari kelas ekivalensi yang berlainan.

Jadi (p, q) tak bertanda L_p dan L_q dapat digabung menjadi satu kelas ekivalensi. Dengan demikian pula jika semua pasangan bertanda maka FA sudah minimal.

Contoh Untuk FA yang sama seperti yang dibahas dengan cara sebelumnya.



Pada pass pertama:
karena status 6 adalah status terima maka pasangan (1, 6), (2, 6), (3, 6), (4, 6), (5, 6), dan (6, 7) ditandai.

Pada pass kedua:
(1, 3) ditandai karena $d(1, 0) = 2$ dan $d(3, 0) = 6$ serta (2, 6) sudah ditandai
(1, 5) ditandai karena $d(1, 0) = 2$ dan $d(5, 0) = 6$ serta (2, 6) sudah ditandai
(1, 7) ditandai karena $d(1, 0) = 2$ dan $d(7, 0) = 6$ serta (2, 6) sudah ditandai
(2, 3) ditandai karena $d(2, 0) = 4$ dan $d(3, 0) = 6$ serta (4, 6) sudah ditandai
(2, 5) ditandai karena $d(2, 0) = 4$ dan $d(5, 0) = 6$ serta (4, 6) sudah ditandai
(2, 7) ditandai karena $d(2, 0) = 4$ dan $d(7, 0) = 6$ serta (4, 6) sudah ditandai
(3, 4) ditandai karena $d(4, 0) = 4$ dan $d(3, 0) = 6$ serta (4, 6) sudah ditandai
(4, 5) ditandai karena $d(4, 0) = 4$ dan $d(5, 0) = 6$ serta (4, 6) sudah ditandai
(4, 7) ditandai karena $d(4, 0) = 4$ dan $d(7, 0) = 6$ serta (4, 6) sudah ditandai
(3, 4) ditandai karena $d(4, 0) = 4$ dan $d(3, 0) = 6$ serta (4, 6) sudah ditandai

Pada pass ketiga tidak ada lagi pasangan yang dapat ditandai, dan selesai.
Pasangan (1, 2), (1, 4), (2, 4), (3, 5), (3, 7), (5, 7) tidak ditandai sehingga didapat tiga kelas ekivalensi $p_1 = L_1 \cup L_2 \cup L_4$, $p_2 = L_3 \cup L_5 \cup L_7$ dan $p_3 = L_6$.

2						
3	2	2				
4			2			
5	2	2		2		
6	1	1	1	1	1	
7	2	2		2		1
	1	2	3	4	5	6