Bahasa-bahasa Reguler: Definisi dan Sifat-sifat

Kuliah Teori Bahasa dan Automata
Program Studi Ilmu Komputer
Fasilkom UI

Prepared by:
Suryana Setiawan

Bahasa Reguler

- Dari suatu alfabet Σ, maka bahasa-bahasa reguler adalah:
  - Bahasa kosong (∅), bahasa hanya berisi satu string kosong ({ε}), serta bahasa-bahasa hanya berisi satu string simbol tunggal ({a}, yang mana $a \in \Sigma$).
  - Bahasa-bahasa hasil operasi **union** dari bahasa-bahasa reguler: $L_1 = L_2 \cup L_3$, yang mana L_2 dan L_3 adalah reguler
  - Bahasa hasil operasi **konkatenasi** dari bahasa-bahasa reguler: $L_1 = L_2 L_3$, yang mana L_2 dan L_3 adalah reguler
  - Bahasa hasil operasi **Kleene*** dari bahasa reguler

Contoh-contoh

- $L = \{w \in \{0, 1\}^* : w \text{ berakhiran } 10\}$
  $= \{0, 1\}^*\{10\} = (\{0\} \cup \{1\})^*(\{1\}\{0\})$
- $L = \{a^m b^n : m, n \geq 0\}$
  $= \{a\}^*\{b\}^*$
- $L = \{w \in \{a, b\}^* : |w| \text{ bilangan genap}\}$
  $= (\{a \cup b\}\ \{a \cup b\})^*$

- Pertanyaan: bisakah $\{a^n b^n : n \geq 0\}$ diekspresikan sebagai hasil operasi-operasi demikian? Tidak! (mengapa?)
- Bagaimana dengan $\{a^n b^n : n \geq 0, \text{ dan } n < 10\}$

Bahasa-bahasa Finite

- Setiap Bahasa finite (berhingga) adalah regular.
  - Mengapa? Karena bisa dianggap union dari bahasa-Bahasa yang berisi satu string, dan setiap bahasa yang berisi satu string merupakan bahasa hasil konkatenasi dari bahasa-bahasa berisi satu string satu symbol (bahasa sederhana).
- Bahasa finite bisa memiliki string sangat banyak!
  - Contoh: $\{w \in \{0\text{-}9\}^* \mid w$ nomor-nomor SIM card di Indonesia$\}$

Ekspresi Reguler (Regex)

- Penulisan deretan operasi himpunan untuk menspesifikasikan bahasa reguler dapat disederhanakan menjadi suatu **string ekspresi reguler**.

Ekspresi Himpunan: L	Ekspresi Reguler: α
$\varnothing$ atau $\{\}$	$\varnothing$
$\{\varepsilon\}$	ε
$\{a\}$	a
$L_1 \cup L_2$	$\alpha \cup \beta$
$L_1 L_2$	$\alpha \beta$
L^*	α^*
L^+	α^+
(L)	(α)

Contoh-contoh

- $L_1 = \{w \in \{0, 1\}^* : w \text{ berakhiran } 10\}$
  $= \{0, 1\}^*\{10\} = (\{0\} \cup \{1\})^*(\{1\}\{0\})$
  ekspresi reguler L_1 adalah $(0 \cup 1)^*10$
- $L_2 = \{\in \{a, b\}^* : w \text{ tidak berisi substring } aa\}$
  ekspresi reguler L_2 adalah $(ab \cup b)^*(a \cup \varepsilon)$
- $L_3 = \{a^m b^n : m, n \geq 0\} = \{a\}^*\{b\}^*$
  ekspresi reguler L_3 adalah a^*b^*
- $L_4 = \{w \in \{a, b\}^* : |w| \text{ bilangan genap}\}$
  $= (\{a \cup b\}\{a \cup b\})^*$
  ekspresi reguler L_4 adalah $((a \cup b)(a \cup b))^*$

Precedence Order dari Operator

- Seperti pada ekspresi aljabar, ekspresi reguler memerlukan pengurutan presedensi operator:
  - Tertinggi Kleene-* (seperti halnya pangkat),
  - Kemudian konkatenasi (seperti halnya perkalian),
  - Lalu union (seperti halnya penjumlahan)
  - Sepasang kurung menyatukan ekspresi di dalam tanda kurung tsb terhadap operator di luarnya.

Bahasa vs Ekspresi

- Ingat: **bahasa reguler *L*** adalah himpunan string sementara **ekspresi reguler** α adalah string menspesifikasikan bahasa reguler atau $\boldsymbol{L(\alpha)}$.
- Suatu bahasa reguler L bisa dispesifikasikan oleh beberapa ekspresi reguler berbeda (tidak unik, tapi arti sama!)
  - Contoh: $L = \{w \in \{a, b\}^* : \#_a(w)$ bilangan ganjil$\}$
  - Ekspresi regulernya:
    - Bisa $b^*(ab^*ab^*)^*ab^*$, juga bisa $b^*ab^*(ab^*ab^*)^*$

Regex vs. Ekspresi Reguler

- Regex adalah nama popular dalam fitur pemrosesan teks dalam operating system, text editor, bahasa-bahasa pemrograman, dll.
- Mulanya Regex terkait kelas bahasa regular namun selain notasinya bertambah juga berevolusi meliputi hal-hal di luar bahasa regular.
- Pembahasan disini adalah ekspresi regular (regex, dengan huruf kecil) bukan Regex.

Sifat-sifat Closure Bahasa-bahasa Reguler

- Operasi bahasa-bahasa regular yang bersifat closure dalam bahasa-bahasa reguler:
  - jika operasi dilakukan pada bahasa regular, menghasilkan suatu bahasa regular juga.
- Bahasa-bahasa reguler bersifat closure terhadap operasi union, konkatenasi dan Kleene Star.
  - Sudah jelas dalam definisinya!
- Bahasa-bahasa reguler bersifat closure terhadap operasi komplemen, irisan, set-difference, reverse, dan substitusi symbol.
  - Akan dibahas di slide berikutnya.

Closure thd Operasi Komplemen

- "Komplemen dari setiap Bahasa regular adalah juga regular."
- Dibuktikan dengan menemukan DFSM M_2 yang bekerja berlawanan dengan DFSM M_1, dimana M_1 mesin untuk menerima bahasa reguler L.
  - Jika $M_1 = \{K, \Sigma, \delta, s, A)$ maka $M_2 = \{K, \Sigma, \delta, s, K\text{-}A)$ accepting state menjadi nonaccepting state, sementara nonaccepting state menjadi accepting state.

Closure thd Operasi Irisan

- "Hasil operasi irisan dari Bahasa-bahasa reguler adalah juga bahasa reguler."
- Menurut de'Morgan L_1 dan L_2 adalah dua himpunan maka: $L_1 \cap L_2 = \neg(\neg L_1 \cup \neg L_2)$
  - Jika L_1 dan L_2 bahasa-bahasa reguler arena union dan komplemen bersifat closure thd bahasa-bahasa reguler maka $\neg L_1$ reguler, $\neg L_2$ reguler, kemudian $(\neg L_1 \cup \neg L_2)$ reguler dan akhirnya $L_1 \cap L_2$ regular.

Closure thd Operasi Set-Difference

- "Hasil operasi "set-difference" dari bahasa-bahasa reguler L_1 oleh L_2 adalah juga bahasa reguler."

  $$L_1 - L_2 = L_1 \cap \neg L_2$$

  Karena operasi-operasi irisan dan komplemen bersifat closure terhadap bahasa-bahasa regular maka hasilnya juga suatu bahasa regular.

Pertanyaan:

- Apakah hasil operasi reverse suatu bahasa regular juga bahasa regular?

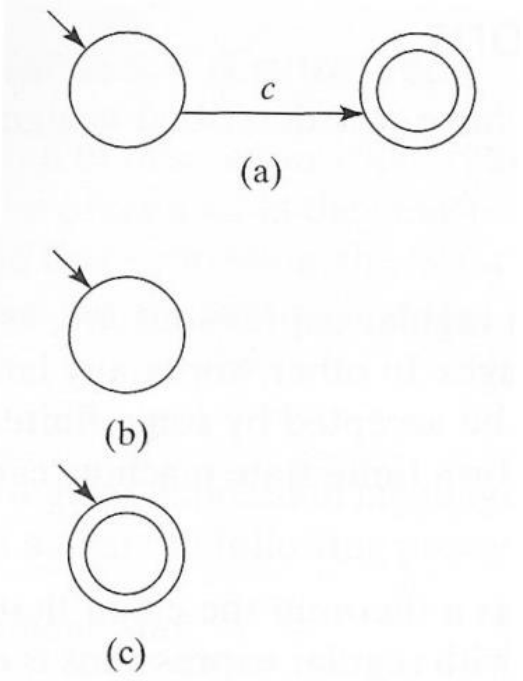
- Apakah hasil operasi substitusi suatu bahasa regular juga bahasa regular?

Teorema Kleene

- Teorema Kleene mengambil nama Stephen Cole Kleene
- Teorema aslinya menyatakan ekivalensi tiga pernyataan berikut.
  - Bahasa yang diekspresikan ekspresi reguler
  - Bahasa yang diterima NDFSM
  - Bahasa yang diterima DFSM
- Kedua terakhir sudah dibuktikan dengan adanya algoritma konversi NDFS ➔➔ DFSM
- Materi berikutnya adalah terkait ekspresi reguler dengan FSM, dalam 2 bagian
  - Ekspresi reguler ➔ NDFSM
  - FSM ➔ ekspresi reguler
- Keduanya dibuktikan adanya algoritma konversi juga

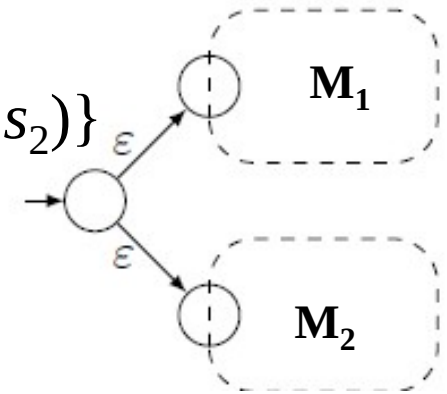
Ekspresi Reguler → FSM

- **Teorema**: bahasa untuk setiap ekspresi reguler, dapat diterima oleh bbrp FSM tertentu, dan berarti juga bahasa reguler.
- **Bukti**: dengan membangun FSM *M* dari ekspesi reguler α sehingga $L(\alpha) = L(M)$
  - Untuk $\alpha = c$, $c \in \Sigma$, mesin *M* seperti Gambar (a).
  - Untuk $\alpha = \varnothing$, mesin *M* seperti Gambar (b)
  - Untuk $\alpha = \varepsilon$, mesin *M* seperti Gambar (c).
  - Untuk α lain….(next page)



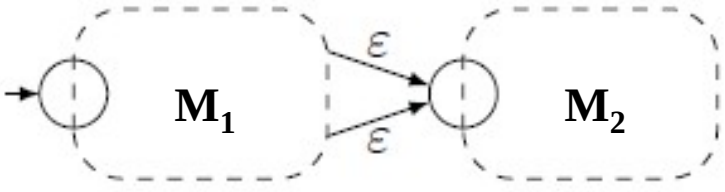
Regex→ FSM [Union]

- Untuk α lain, diketahui juga bahwa β dan γ ekspresi-ekspresi reguler dengan
  - $L(\beta)$ diterima oleh $M_1 = (K_1, \Sigma, \delta_1, s_1, A_1)$,
  - $L(\gamma)$ diterima oleh $M_2 = (K_2, \Sigma, \delta_2, s_2, A_2)$, dan
  - $K_1 \cap K_2 = \varnothing$
- Untuk $\alpha = \beta \cup \gamma$, sehingga $L(\alpha) = L(\beta) \cup L(\gamma)$, maka $L(\alpha)$ diterima oleh $M_3 = (K_3, \Sigma, \delta_3, s_3, A_3)$ sbb:
  - $K_3 = K_1 \cup K_2 \cup \{s_3\}$, $s_3 \notin K_1 \cup K_2$
  - $\delta_3 = \delta_1 \cup \delta_2 \cup \{((s_3, \varepsilon), s_1), ((s_3, \varepsilon), s_2)\}$
  - $A_3 = A_1 \cup A_2$



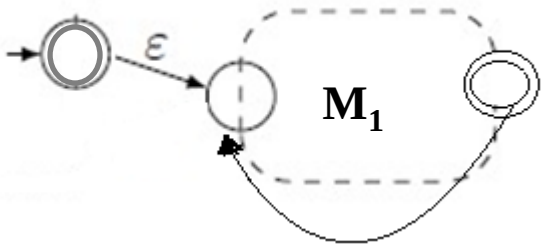
Regex → FSM [Konkatenasi]

- Untuk α lain, diketahui juga bahwa β dan γ ekspresi-ekspresi reguler dengan
  - $L(\beta)$ diterima oleh $M_1 = (K_1, \Sigma, \delta_1, s_1, A_1)$,
  - $L(\gamma)$ diterima oleh $M_2 = (K_2, \Sigma, \delta_2, s_2, A_2)$, dan
  - $K_1 \cap K_2 = \varnothing$
- Untuk $\alpha = \beta\,\gamma$, sehingga $L(\alpha) = L(\beta)\,L(\gamma)$, maka $L(\alpha)$ diterima oleh $M_3 = (K_3, \Sigma, \delta_3, s_3, A_3)$, sbb:
  - $K_3 = K_1 \cup K_2$
  - $\delta_3 = \delta_1 \cup \delta_2 \cup \{((q, \varepsilon), s_2) : q \in A_1\}$
  - $A_3 = A_2$

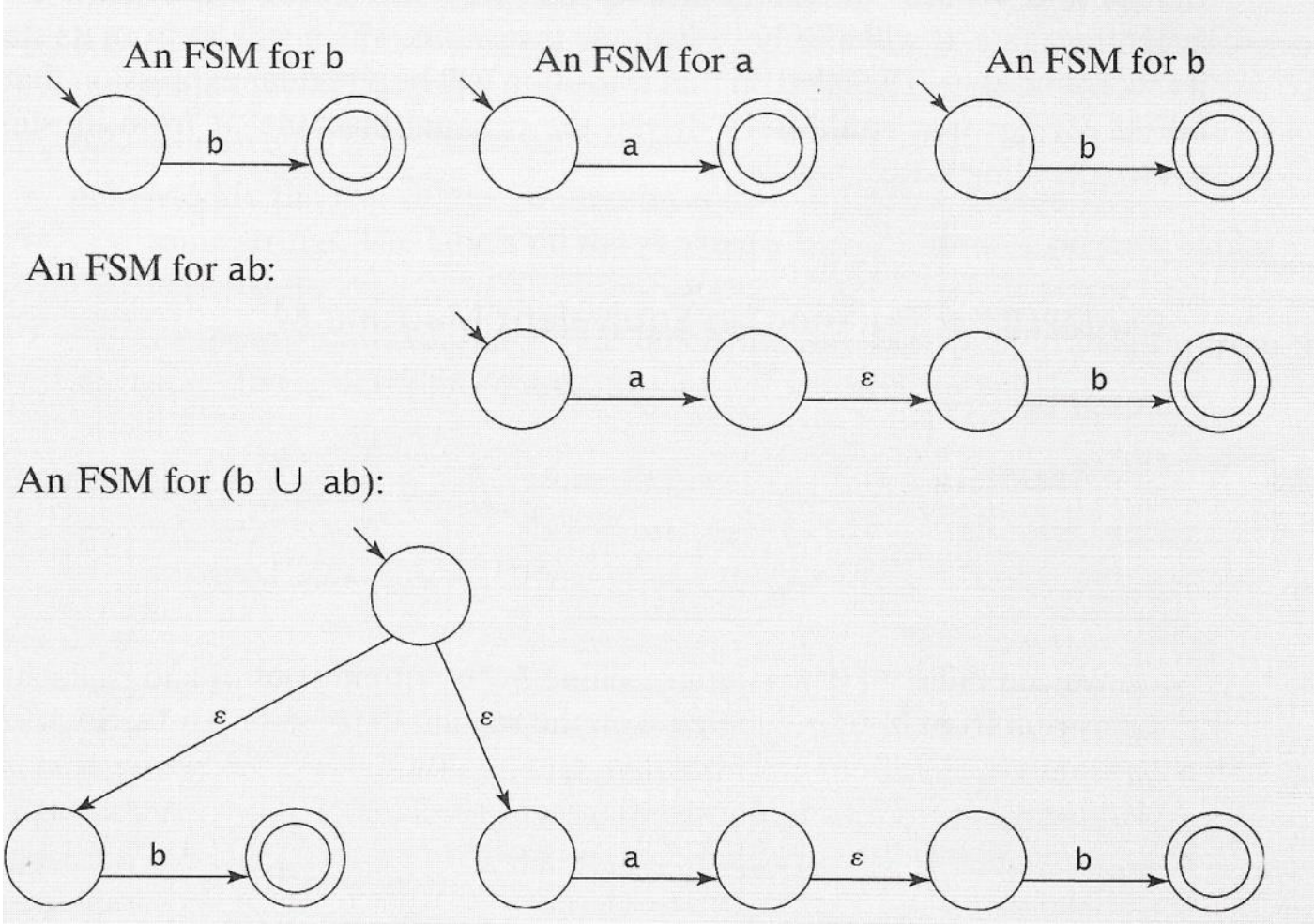


Regex → FSM [Kleene*]

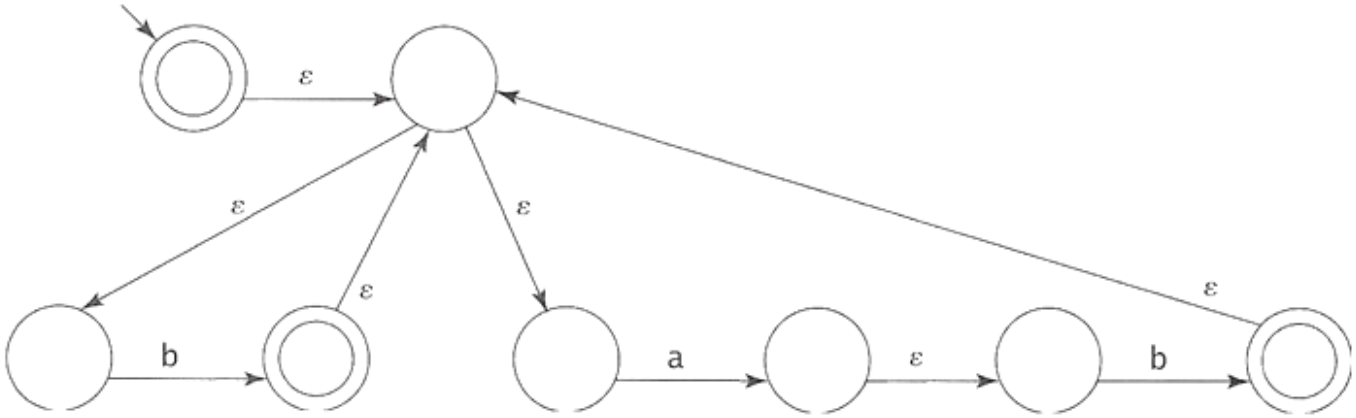
- Untuk α lain, diketahui juga bahwa β ekspresi reguler dengan $L(\beta)$ diterima oleh $M_1 = (K_1, \Sigma, \delta_1, s_1, A_1)$,
- Untuk $\alpha = \beta^*$, sehingga $L(\alpha) = L(\beta)^*$, maka $L(\alpha)$ dapat diterima oleh $M_3 = (K_3, \Sigma, \delta_3, s_3, A_3)$ sbb:
  - $K_3 = K_1 \cup \{s_3\}$, dengan $s_3 \notin K_1$
  - $\delta_3 = \delta_1 \cup \{((s_3, \varepsilon), s_1), ((q, \varepsilon), s_1) : q \in A_1\}$
  - $A_3 = A_1 \cup \{s_3\}$



Contoh Untuk $(b \cup ab)^*$



An FSM for (b ∪ ab)*



Catatan Tambahan

- Operasi-operasi pada mesin tersebut bersifat algoritmis (untuk kepentingan implementasi) demi untuk membuktikan bahwa dari ekspresi regular α bisa di b angun NDFSM yang dapat menerima $L(\alpha)$.
- Jika hanya untuk dilakukan secara manual beberap penyederhanaan bisa dilakukan.
  - Misalnya dalam konkatenasi, jika accepting state M_1 hanya satu maka bias langsung menjadi start state dari M_2.
  - Dalam union, jika pada kedua mesin tidak ada transisi kembali ke start-state, maka kedua start state bisa digabung menjadi start-state mesin gabungan (tanpa ada penambahan start-state baru).