



# Finite State Machines

Kuliah Teori Bahasa dan Automata  
Program Studi Ilmu Komputer  
Fasilkom UI

Prepared by:

Suryana Setiawan

Revised by:

Maya Retno Ayu S.

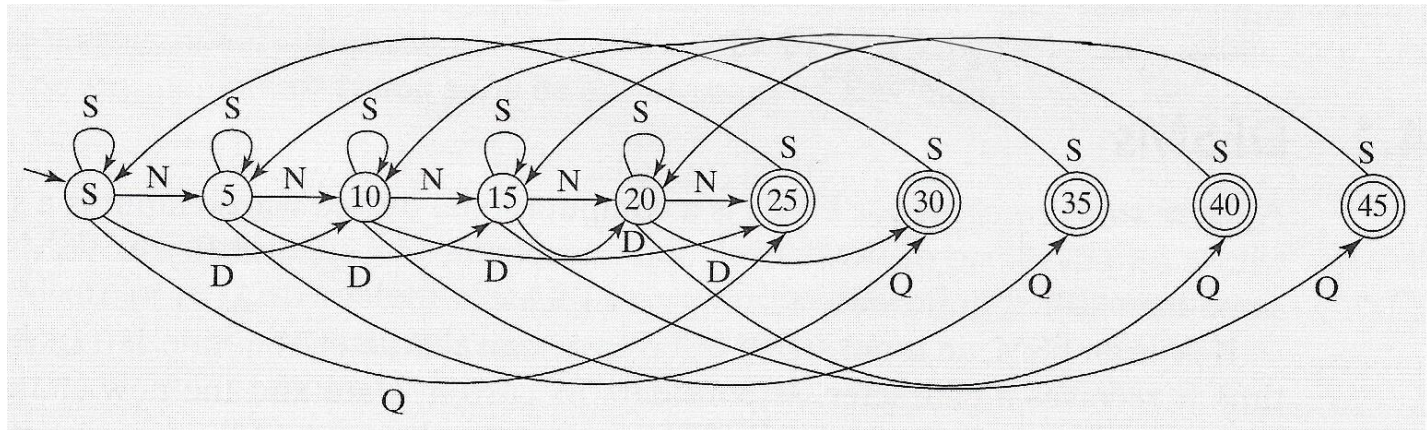
# Deskripsi Intuitif FSM

- FSM adalah model mesin komputasi yang paling sederhana dan efisien.
- FSM terdefinisi dalam sejumlah status.
- FSM bekerja dengan berubah dari satu status ke status lainnya dari sejumlah kemungkinan berhingga status sesuai dengan masukan yang diterimanya.
- Mesin-mesin demikian di sekeliling kita:
  - Mesin jahit
  - Mesin kendaraan
  - Senapan
  - Kunci pintu
  - Switch lampu, dll

# Contoh FSM: Vending Machine

- Mesin menjual minuman seharga @ 25 sen, dan koin yang ada 5 sen (N), 10 sen (D), 25 sen (Q).
- Mesin menerima koin-koin satu demi satu sebagai masukan.
- Mesin dapat berada dalam status-status yang menyatakan nilai total koin yang sudah dimasukkan:  
 $q_0$  (atau S),  $q_5$ ,  $q_{10}$ ,  $q_{15}$ ,  $q_{20}$ ,  $q_{25}$ ,  $q_{30}$ ,  $q_{40}$  dan  $q_{45}$ .
- Mesin hanya akan mengeluarkan minuman jika tombol S ditekan sementara mesin berada pada **dispensing state**, lalu diikuti perubahan
  - Status semula  $q_x$ , kemudian menjadi  $q_{(x-25)}$
- **Non-dispensing states**: menyatakan mesin tidak bisa mengeluarkan minuman meskipun sudah masuk koin.

# Contoh Diagram



- Status  $q_x$  digambarkan sebagai lingkaran berlabel bilangan  $x$ 
  - *Dispensing states*  $\rightarrow$  double circle dan
  - *Non-dispensing states*  $\rightarrow$  single circle.
- Input N, D, Q, dan S digambarkan pada panah berlabel input tsb
- Jika status semula  $x$  dan *non-dispensing*, pemasukan koin berharga  $q_y$  mengubah status menjadi  $q_{x+y}$ .
- Penekanan S pada non-dispensable tidak mengubah status (self-loop).
- Penekanan S pada dispensable mengubah  $q_x$  menjadi  $q_{x-25}$ .
- Catatan: mesin ini belum cukup lengkap. Carilah fitur apa saja yang bisa ditambahkan dan ubahlah diagram tsb.

# Finite State Machine

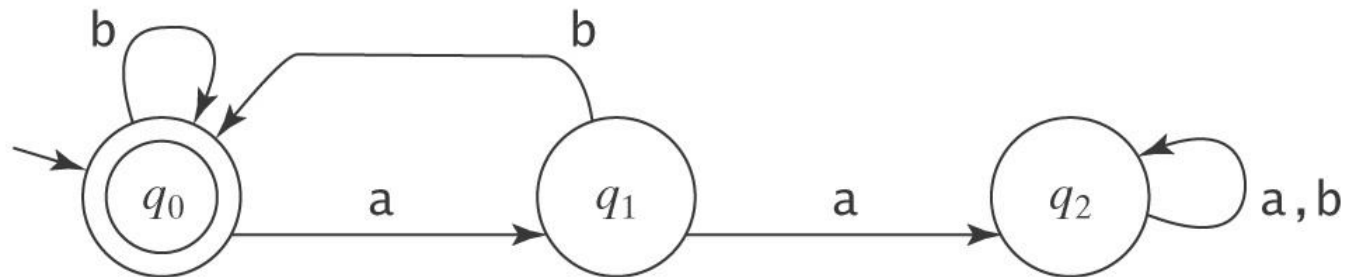
- Vending machine merupakan contoh dari suatu finite state machine.
  - Istilah “finite state” mengacu pada himpunan berhingga status!
  - Mesin hanya “mengingat” berdasarkan keberadaannya pada status-status yang berhingga itu.
  - Mesin selalu mulai dari satu status khusus (**Start state**).
- Dua varian Utama: **Deterministic FSM (DFSM)** dan **Nondeterministic FSM (NDFSM)**
  - Note: Vending Machine sebelumnya adalah salah satu contoh DFSM.
- Sejumlah varian lainnya: **Finite State Transducer** (FSM dengan output), **Büchi Automata** (bisa mulai dari status mana saja).

# Definisi Formal FSM

Deterministic FSM  $M$  adalah kuintupel  $(K, \Sigma, \delta, s, A)$  dengan:

- $K$  adalah himpunan berhingga status-status.
- $\Sigma$ : alfabet input
- $s \in K$ , adalah status mulai (*start state*)
- $A \subseteq K$ , adalah himpunan status menerima (*accepting states*)
- $\delta$ : **fungsi transisi** yang memetakan  
$$K \times \Sigma \rightarrow K$$

# Contoh-1 DFSM



- Mesin mengenal  $L = \{w \in \{a, b\}^* : \text{setiap } a \text{ segera diikuti oleh } b\}$   
 $K = \{q_0, q_1, q_2\}, \quad \Sigma = \{a, b\}, \quad S = q_0, \quad A = \{q_0\},$   
 $\delta = \{((q_0, a), q_1), ((q_0, b), q_0), ((q_1, a), q_2), ((q_1, b), q_0),$   
 $((q_2, a), q_2), ((q_2, b), q_2)\}$
- Note:  $q_2$  disebut dead state! Apa itu?

# Konfigurasi dari DFMS

- Saat DFMS  $M$  melakukan komputasi untuk suatu input string  $w$ , mesin akan berada pada satu konfigurasi dan kemudian berpindah ke konfigurasi berikutnya.
- Konfigurasi adalah element dari  $K \times \Sigma^*$ .
  - $K$  adalah current state.
  - $\Sigma^*$  adalah sisa string yang belum diproses.
- Konfigurasi awal (*initial configuration*) untuk string masukan  $w$  adalah  $(s_M, w)$  dimana  $s_M$  adalah status mulai dari  $M$ .
  - Contoh: jika mesin pada Contoh 1 akan memproses string *abbabab*, konfigurasi awalnya adalah  $(q_0, \text{abbabab})$



# Relasi Yields-in-One-Step

- Relasi *yields-in-one-step*  $\vdash_M$  mendefinisikan perubahan suatu konfigurasi ke konfigurasi berikutnya dalam satu Langkah transisi sbb:

$$(q_1, cw) \vdash_M (q_2, w) \text{ iff } ((q_1, c), q_2) \in \delta.$$

- Contoh: Deretan perubahan konfigurasinya sbb:
  - $(q_0, abbabab) \vdash_M (q_1, bbabab) \vdash_M (q_0, babab)$   
 $\vdash_M (q_0, abab) \vdash_M (q_1, bab) \vdash_M (q_0, ab)$   
 $\vdash_M (q_1, b) \vdash_M (q_0, \varepsilon)$

# Relasi $yields \vdash^*_M$

- Relasi  $yields \vdash^*_M$  secara intuitif adalah yield in  $n$  steps termasuk  $n = 0$ 
  - $(q_0, abbabab) \vdash^*_M (q_0, abbabab)$
  - $(q_0, abbabab) \vdash^*_M (q_1, bbabab)$
  - $(q_0, abbabab) \vdash^*_M (q_0, babab)$
  - $(q_0, abbabab) \vdash^*_M (q_0, abab)$
  - $(q_0, abbabab) \vdash^*_M (q_1, bab)$
  - $(q_0, abbabab) \vdash^*_M (q_0, ab)$
  - $(q_0, abbabab) \vdash^*_M (q_1, b)$
  - $(q_0, abbabab) \vdash^*_M (q_0, \varepsilon)$
- Relasi  $yields \vdash^*_M$  bersifat *reflexive*, jika  $n = 0$ , dan bersifat *transitive closure* dari  $\vdash_M$  untuk  $n$  lainnya ( $n > 0$ )

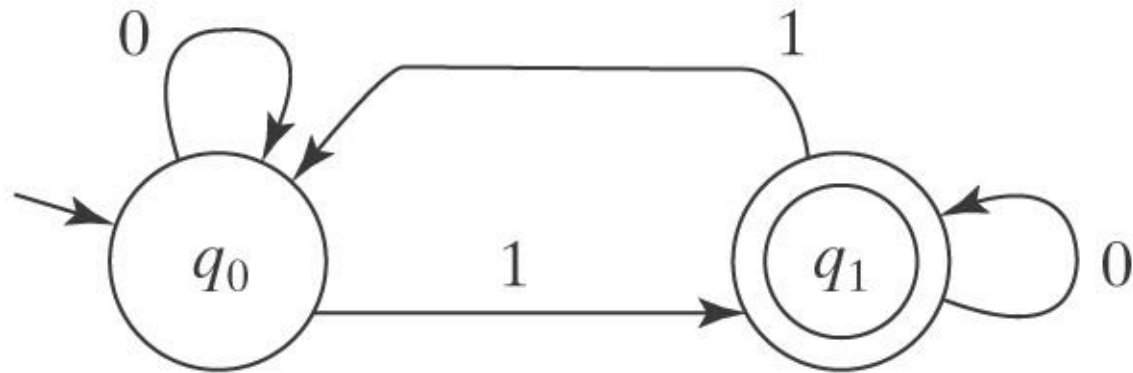
# Komputasi oleh $M$

- Suatu komputasi oleh  $M$  adalah suatu deret berhingga dari konfigurasi  $C_0, C_1, \dots, C_n$  untuk setiap  $n \geq 0$  dimana:
  - $C_0$  adalah konfigurasi awal
  - $C_n$  dalam bentuk  $(q, \varepsilon)$ , untuk status  $q \in K_M$  (artinya setiap symbol dari input string telah dibaca), sehingga
  - $C_0 \vdash_M C_1 \vdash_M \dots \vdash_M C_n$
- Contoh: komputasi oleh mesin dalam Contoh-1 untuk string *abbabab* adalah:  
 $(q_0, \text{abbabab}), (q_1, \text{bbabab}), (q_0, \text{babab}),$   
 $(q_0, \text{abab}), (q_1, \text{bab}), (q_0, \text{ab}), (q_1, \text{b}), \text{dan } (q_0, \varepsilon)$

# Accepting dan Rejecting

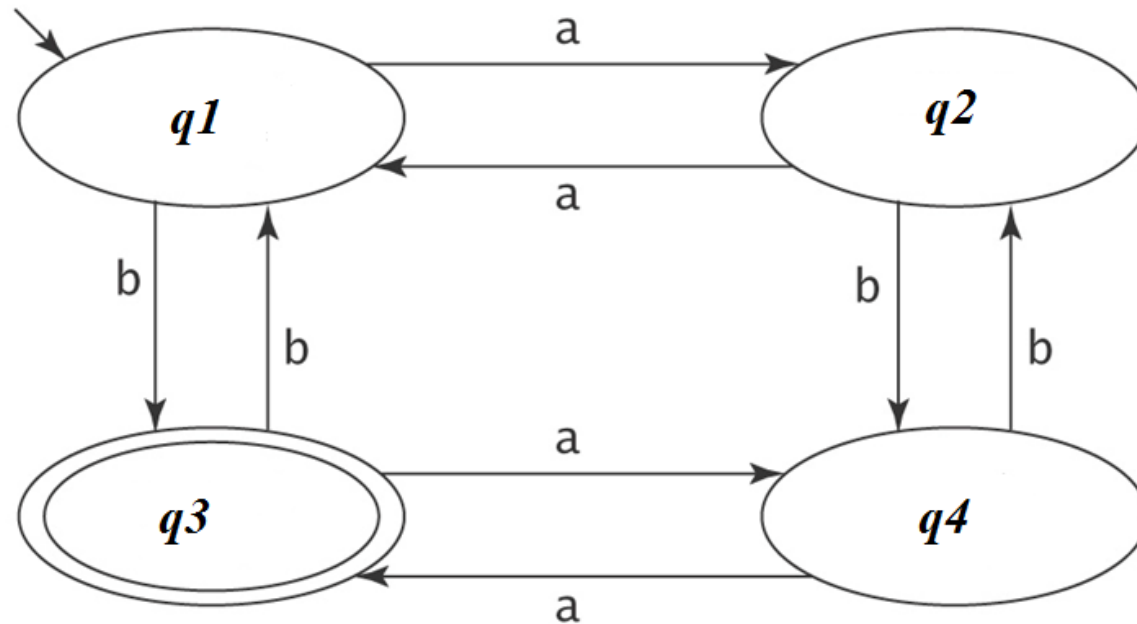
- Dengan input  $w \in \Sigma^*$ , kita akan menyatakan:
  - **M menerima (*accept*)**  $w$  **iff**  $(s, w) \vdash^*_M (q, \varepsilon)$ , untuk setiap  $q \in A$  dan  $(q, \varepsilon)$  disebut **accepting configuration** dari  $M$ .
    - Contoh:  $M$  Contoh-1 menerima *abbabab* karena  $(q_0, abbabab) \vdash^*_M (q_0, \varepsilon)$
  - **M menolak (*reject*)**  $w$  **iff**  $(s, w) \vdash^*_M (q, \varepsilon)$ , untuk setiap  $q \notin A$  dan  $(q, \varepsilon)$  disebut **rejecting configuration** dari  $M$ .
    - Contoh:  $M$  Contoh-1 menolak *abaab* karena  $(q_0, abaab) \vdash^*_M (q_2, \varepsilon)$  dan  $q_2 \notin A$
- Bahasa yang diterima  $M$  ditulis  $L(M)$  adalah himpunan semua string yang diterima oleh  $M$ .

## Contoh-2.1



*Bahasa-bahasa apa yang diterima mesin ini?*

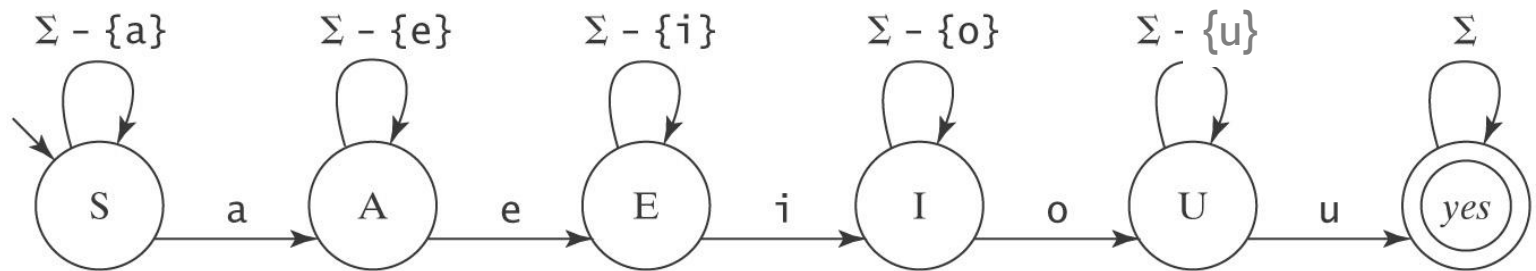
## Contoh 2.2



*Bahasa-bahasa apa yang diterima mesin ini?*

*Jika accepting state berubah ke q1, bahasa apa yang diterima oleh mesin tersebut?*

## Contoh 2.3



*Bahasa-bahasa apa yang diterima mesin ini?*

# Latihan

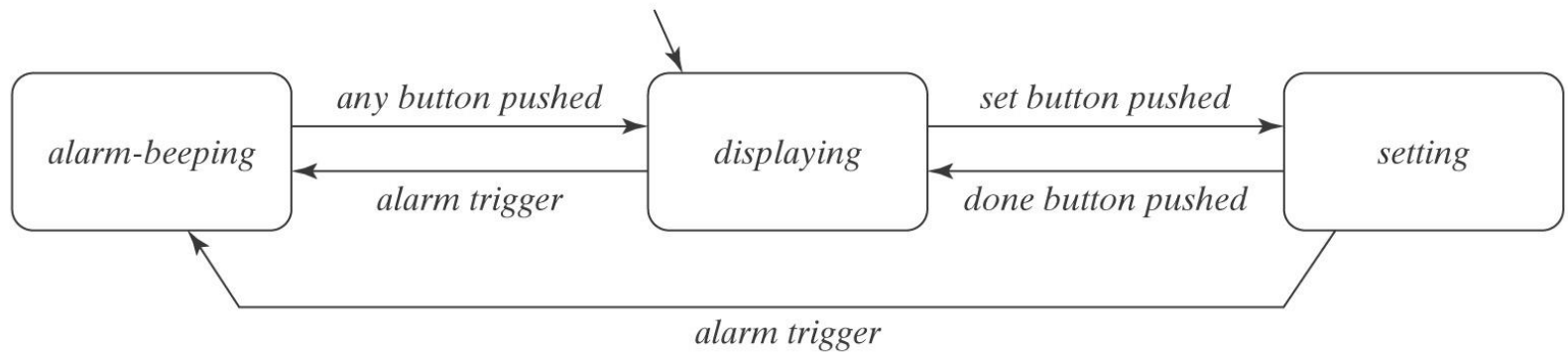
Buatlah FSM yang menerima bahasa:

$L = \{w \in \{a, b\}^* : w \text{ tidak mengandung substring } aab\}$

*Hint: Bisa diawali dengan membuat mesin yang menerima  $\neg L$ .*



# Representasi Finite State pada Software Engineering



*A high-level state chart model untuk jam digital*



# **NONDETERMINISTIC FINITE STATE MACHINE (NDFS)**

# Nondeterminisme

- Jika terdapat pilihan sejumlah aksi untuk dijalankan `choose(aksi1;; aksi2 ;; ...; aksi n)`,
  - Yang mana masing-masing **aksi** akan mengembalikan harga **sukses** atau harga **false**.
- Nondeterminisme dari **choose** akan menjalankan :
  - Mengembalikan harga **sukses** ketika terdapat **sekurangnya satu** yang **sukses**.
  - Jika **tidak ada** satupun yang **sukses**, `choose` akan
    - **Halt** dan mengembalikan **false**, jika setiap pilihan `halt`. atau
    - **Gagal halt** jika semua pilihan aksi **gagal halt**.

# Definisi Formal Nondeterministic FSM (NDFSM)

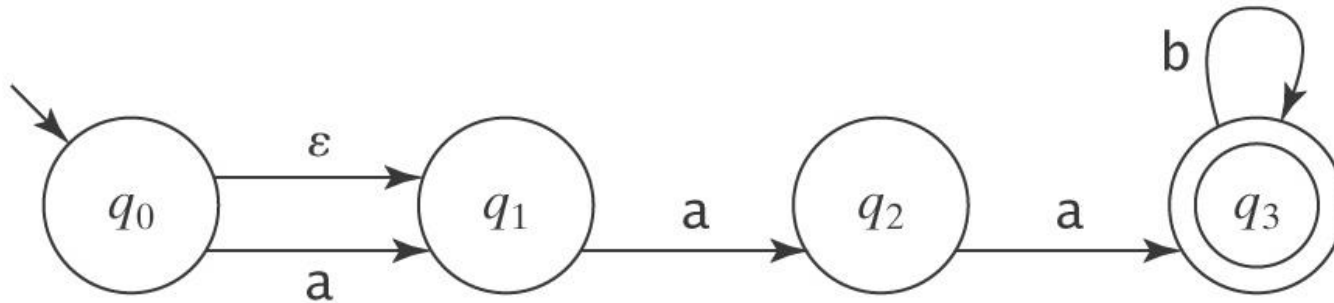
Nondeterministic FSM  $M$  adalah kuintupel  $(K, \Sigma, \Delta, s, A)$  dengan:

- $K$  adalah himpunan berhingga status-status.
- $\Sigma$ : alfabet input
- $s \in K$ , adalah status mulai (*start state*)
- $A \subseteq K$ , adalah himpunan status menerima (*accepting states*)
- $\Delta$  : **relasi transisi** yang merupakan subset dari  $(K \times (\Sigma \cup \{\varepsilon\})) \times K$   
Setiap elemen  $\Delta$  berisikan pasangan (status, simbol masukan atau  $\varepsilon$ ), serta satu status baru.

# Nondeterminisme dengan NDFSM

- Pada setiap konfigurasi
  - DFSM memiliki **tepat satu pilihan transisi** selama masih ada simbol masukan, tetapi
  - NDFSM bisa **0, 1, atau lebih kemungkinan pilihan transisi**. Jika 0 pilihan, maka NDFSM halt.
- Suatu konfigurasi NDFSM dapat dicapai melalui lebih dari satu kemungkinan langkah.
- Transisi NDFSM dapat dilakukan tanpa membaca simbol masukan (disebut **transisi  $\epsilon$** , karena panah diberi label  $\epsilon$ ).
- Hasil NDFSM mengikuti prinsip nondeterminisme.

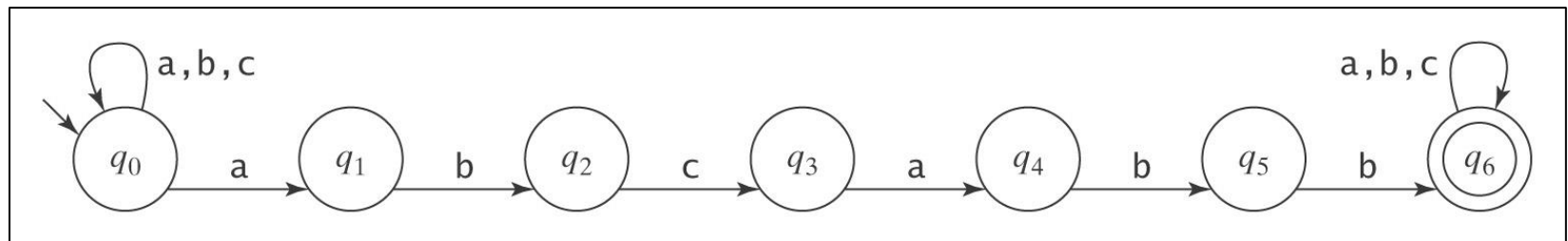
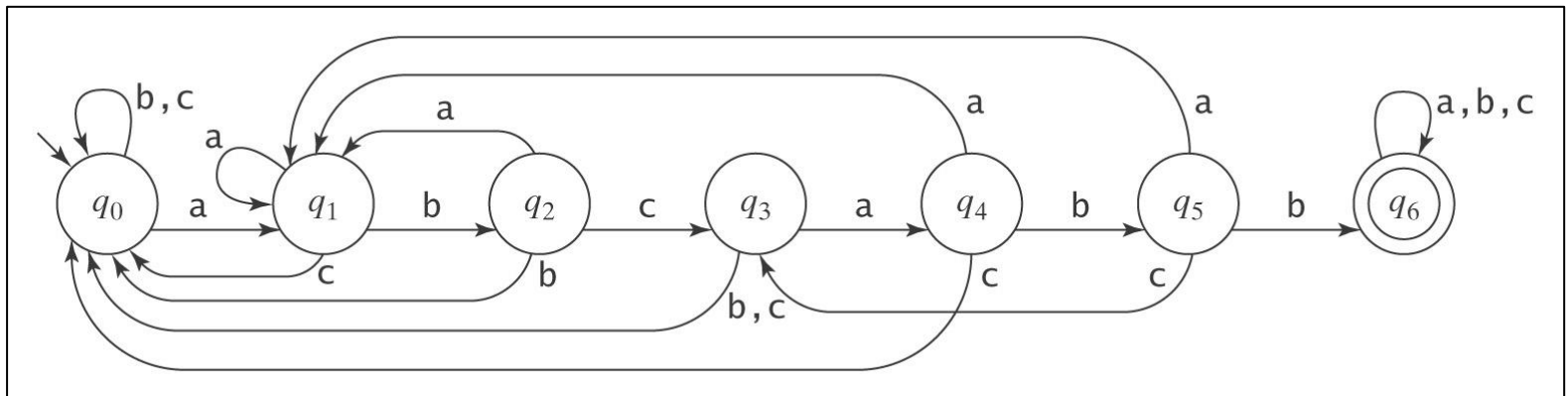
# Contoh 3



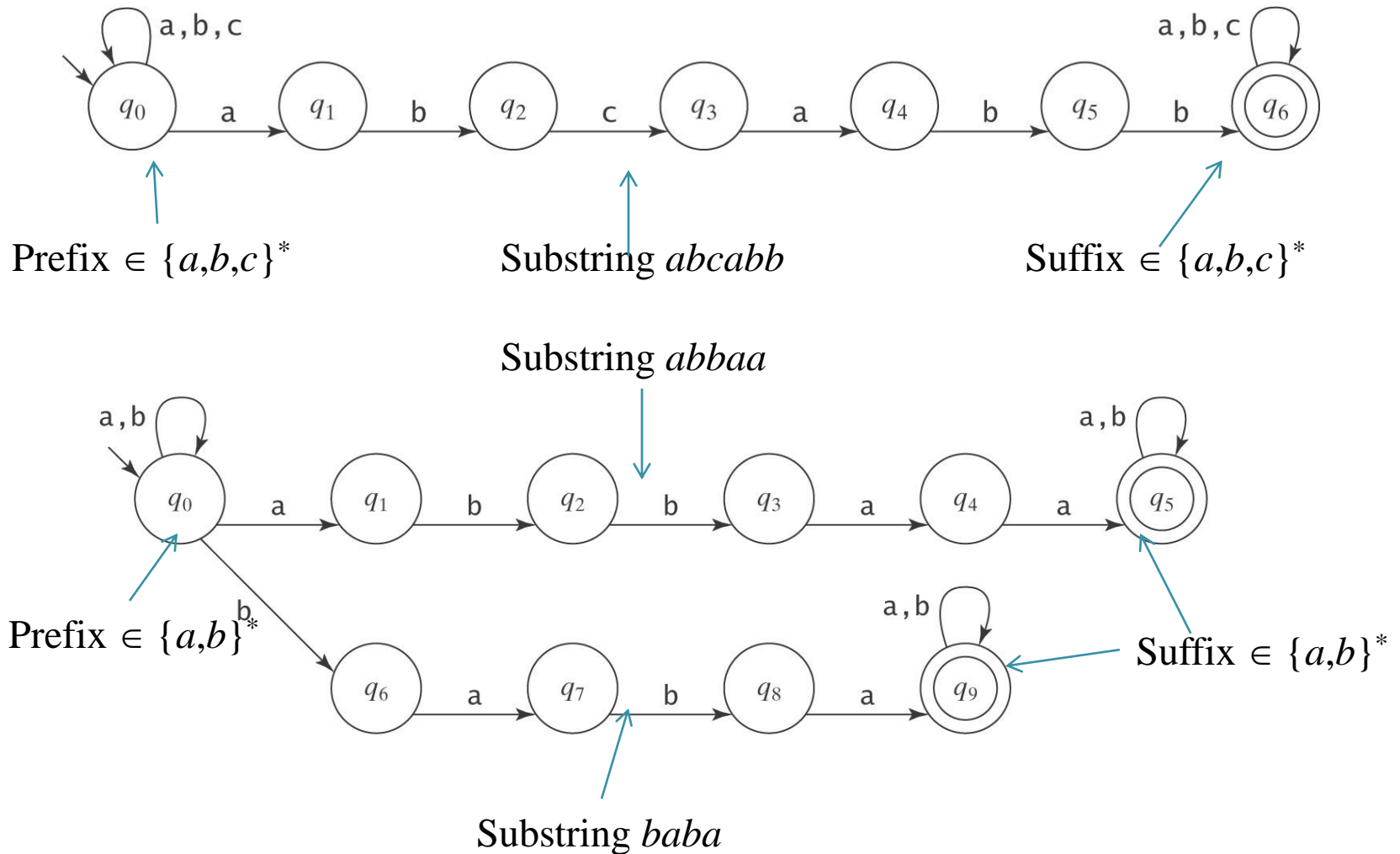
- Mesin menerima  $aa$  atau  $aaa$ , kemudian diikuti sederetan simbol  $b$  dengan panjang 0 atau lebih.
- Input  $aaab$  akan menghasilkan dua pilihan  
 $(q_0, aaab) \vdash^*_M (q_1, aaab) \vdash^*_M (q_2, aab) \vdash^*_M (q_3, ab)$  gagal  
 $(q_0, aaab) \vdash^*_M (q_1, aab) \vdash^*_M (q_2, ab) \vdash^*_M (q_3, b) \vdash^*_M (q_3, \epsilon)$  sukses
- Pilihan pertama dengan transisi  $\epsilon$  berakhir tanpa berhasil mencapai accepting configuration  $(q_3, \epsilon)$  sementara pilihan kedua berhasil.
- Karena ada satu yang sukses maka mesin menerima string  $aaab$ .

# DFSM vs NDFSM

- $L = \{w \in \{a, b, c\}^* : \exists x, y \in \{a, b, c\}^* (w = xabcabby)\}$
- FSM yang menerima  $L$



# Substring Searching

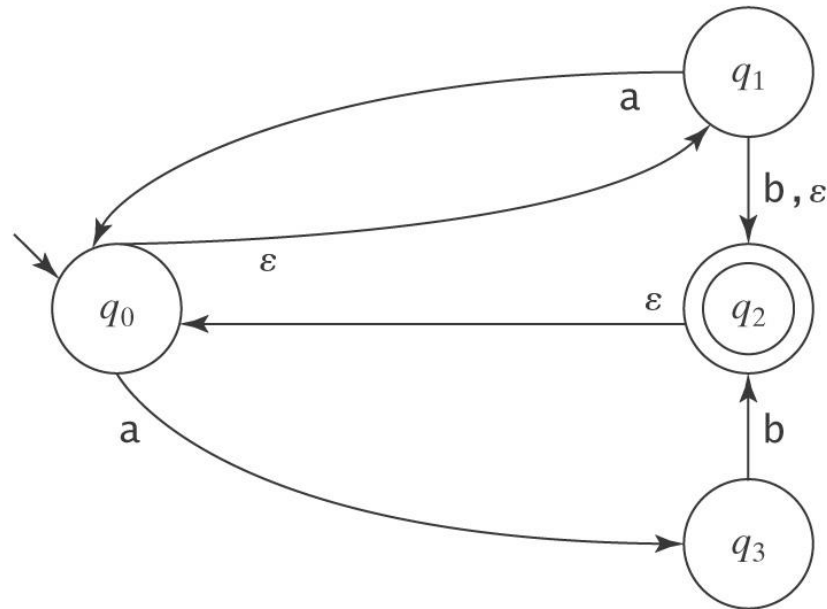




# Reachable States akibat Transisi $\epsilon$

- Transisi  $\epsilon$  dari status  $q_i$  ke  $\{q_j, q_k, \dots\}$  membentuk kelompok status yang reachable dari  $q_i$  walaupun tidak langsung dari  $q_i$ 
  - Jika konfigurasi saat ini adalah  $(q_i, w_i)$ , yield berikut berangkat dari setiap status di dalam kelompok tersebut.
- Fungsi  $eps(q)$  mendefinisikan reachability ini sbb.  
$$eps(q) = \{p \in K : (q, w) \vdash^*_M (p, w)\}$$
- Algoritma menghitung fungsi  $eps(q)$ :
  1.  $result = \{q\}$
  2. Untuk setiap  $p \in result$ , dan terdapat transisi  $(p, \epsilon, r)$ ,  
 $result += \{r\}$
  3. Return result

# Contoh-4



$$eps(q_0) = \{q_0, q_1, q_2\}$$

$$eps(q_1) = \{q_0, q_1, q_2\}$$

$$eps(q_2) = \{q_0, q_1, q_2\}$$

$$eps(q_3) = \{q_3\}$$