# Push Down Automata

Kuliah Teori Bahasa dan Automata
Program Studi Ilmu Komputer
Fasilkom UI

Prepared by:
Rahmad Mahendra
Revised by Suryana Setiawan
Narrated by Suryana Setiawan

# Pengantar

- FSM adalah automata tanpa “storage”
  - kemampuan “mengingat” hanya sebatas status-status mesin dalam jumlah berhingga.
- Bisakah FSM untuk mengenali CFL?

# Pengantar

- FSM adalah automata tanpa “storage”
  - kemampuan “mengingat” hanya sebatas status-status mesin dalam jumlah berhingga.
- Bisakah FSM untuk mengenali CFL?
  - TIDAK DAPAT, jika itu CFL yang BUKAN regular.
- Mengapa?
  - Akibat sifat *self-embedding* dari CFG, diperlukan cara “mengingat” substring dari bagian kiri, saat memeriksa substring bagian di kanan dari pasangan *self-embedding*.

# Pengantar

- FSM adalah automata tanpa “storage”
  - kemampuan “mengingat” hanya sebatas status-status mesin dalam jumlah berhingga.
- Bisakah FSM untuk mengenali CFL?
  - TIDAK DAPAT, jika itu CFL yang BUKAN regular.
- Mengapa?
  - Akibat sifat *self-embedding* dari CFG, diperlukan cara “mengingat” substring dari bagian kiri, saat memeriksa substring bagian di kanan dari pasangan *self-embedding*.
- Diperlukan mesin dengan storage untuk “mengingat”.

# Definisi PDA

**Push Down Automata** (PDA) adalah Finite State Machine yang dilengkapi dengan sebuah *unlimited stack*.

# Definisi PDA

**Push Down Automata** (PDA) adalah Finite State Machine yang dilengkapi dengan sebuah *unlimited stack*.

PDA $M$ adalah tupel $(K, \Sigma, \Gamma, \Delta, s, A)$ dengan:

- $K$ : himpunan berhingga *states*.
- $\Sigma$ : alfabet input
- $\Gamma$ : alfabet *stack*
- $s \in K$, adalah *start state*
- $A \subseteq K$, adalah himpunan *accepting states*
- $\Delta$ : relasi transisi, yang merupakan *subset* berhingga dari

$$\left( K \times (\Sigma \cup \{\varepsilon\}) \times \Gamma^* \right) \times \left( K \times \Gamma^* \right)$$

# Definisi PDA

**Push Down Automata** (PDA) adalah Finite State Machine yang dilengkapi dengan sebuah *unlimited stack*.

PDA $M$ adalah tupel $(K, \Sigma, \Gamma, \Delta, s, A)$ dengan:

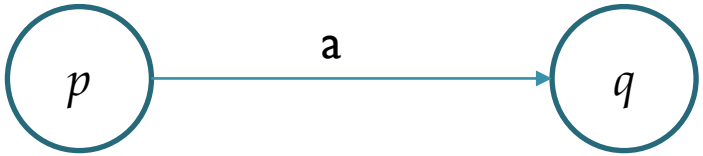
- $K$ : himpunan berhingga *states*.
- $\Sigma$ : alfabet input
- $\Gamma$ : alfabet *stack*
- $s \in K$, adalah *start state*
- $A \subseteq K$, adalah himpunan *accepting states*
- $\Delta$ : relasi transisi, yang merupakan *subset* berhingga dari

$$\left( K \times (\Sigma \cup \{\varepsilon\}) \times \Gamma^* \right) \times \left( K \times \Gamma^* \right)$$

# Highlight Perbedaan FSM dan PDA

- Rule pada NDFSM:

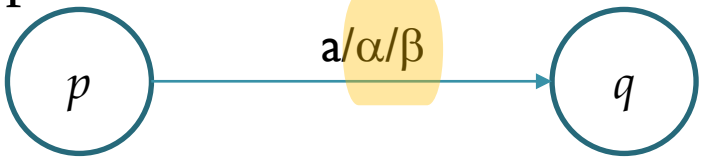
$$\left(K \times (\Sigma \cup \{\varepsilon\})\right) \times K$$




- Tambahan komponen stack Γ. Rule pada PDA:

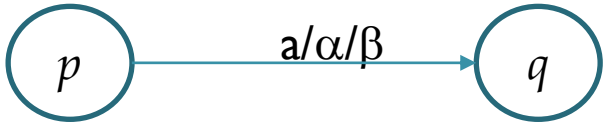
$$\underbrace{\left( K \times (\Sigma \cup \{\varepsilon\}) \times \Gamma^* \right)}_{\text{state}\quad\text{input or }\varepsilon\quad\text{string of symbols to pop from top of stack}} \times \underbrace{\left( K \times \Gamma^* \right)}_{\text{state}\quad\text{string of symbols to push on top of stack}}$$

Mekanisme update stack: sebelum transisi α menjadi β



# Transisi dan Mekanisme Stack

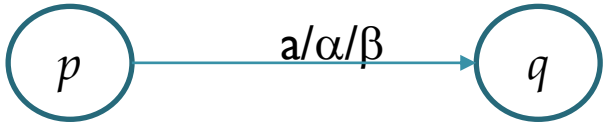
- Secara formal transisi dituliskan sebagai **relasi** $((p, a, \alpha), (q, \beta))$, dan secara grafis, digambarkan sebagai:



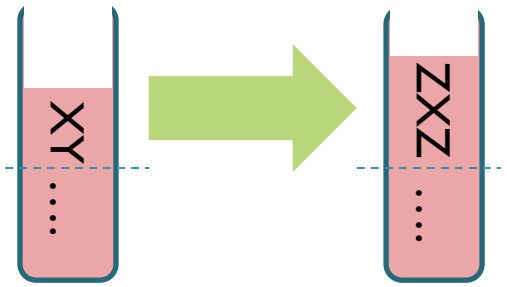

- Transisi dapat dijalankan jika input yang dibaca a **dan** top-of-stack berisi **prefiks** $\alpha$.
- Setelah transisi prefiks $\alpha$ di-replace oleh $\beta$.

# Transisi dan Mekanisme Stack

- Secara formal transisi dituliskan sebagai **relasi** $((p, a, \alpha), (q, \beta))$, dan secara grafis, digambarkab sebagai:

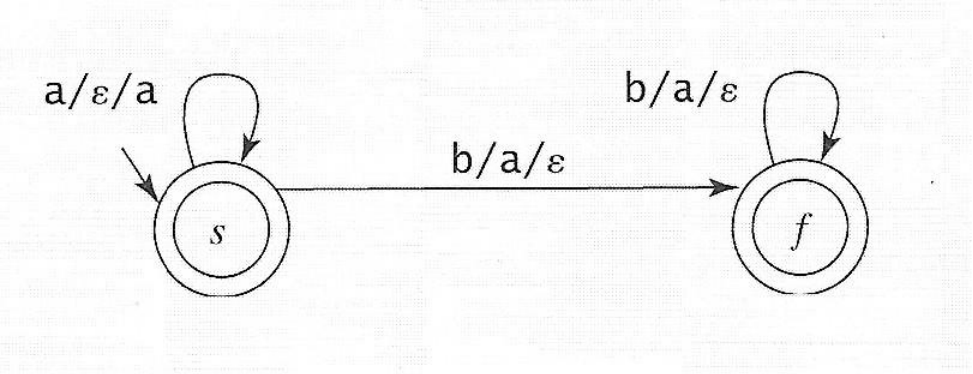



- Transisi dapat dijalankan jika input yang dibaca a **dan** top-of-stack berisi **prefiks** $\alpha$.
- Setelah transisi prefiks $\alpha$ di-replace oleh $\beta$.
- Contoh jika $\alpha$ = XY dan $\beta$ = ZXZ
  - **Note**: isi stack **top→bottom** dituliskan dari **kiri →kanan**.



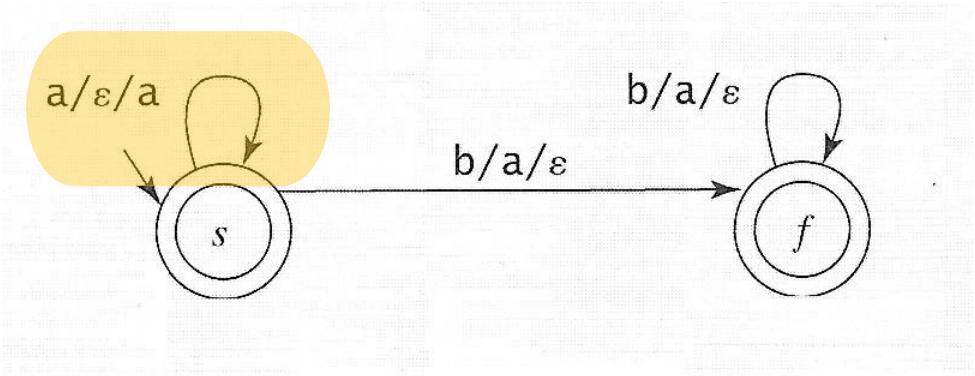
# Contoh-1 PDA

- $A^nB^n = \{a^nb^n : n \geq 0\}$
- Bahasa $A^nB^n$ diterima oleh $M = (K, \Sigma, \Gamma, \Delta, s, A)$ di mana
  - $K = \{s, f\}$ $A = \{s, f\}$
  - $\Sigma = \{a, b\}$ $\Gamma = \{a\}$
  - $\Delta = \{((s, a, \varepsilon), (s, a)), ((s, b, a), (f, \varepsilon)), ((f, b, a), (f, \varepsilon))\}$



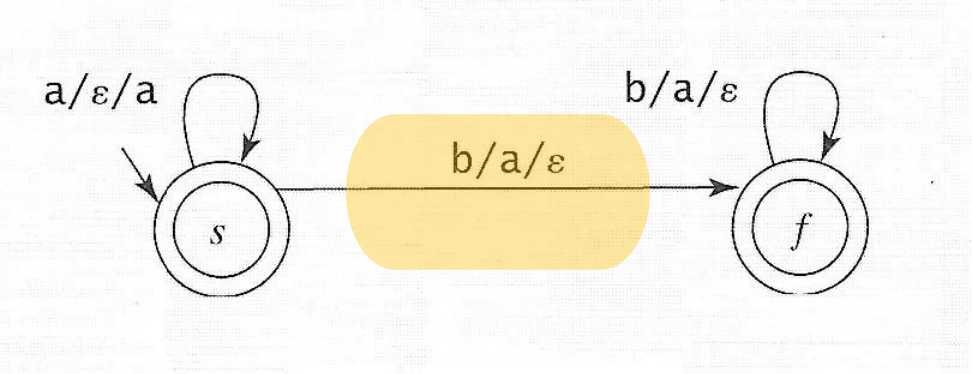
# Contoh-1 PDA

- $A^nB^n = \{a^nb^n : n \geq 0\}$
- Bahasa $A^nB^n$ diterima oleh $M = (K, \Sigma, \Gamma, \Delta, s, A)$ di mana
  - $K = \{s, f\}$ $A = \{s, f\}$
  - $\Sigma = \{a, b\}$ $\Gamma = \{a\}$
  - $\Delta = \{((s, a, \varepsilon), (s, a)), ((s, b, a), (f, \varepsilon)), ((f, b, a), (f, \varepsilon))\}$



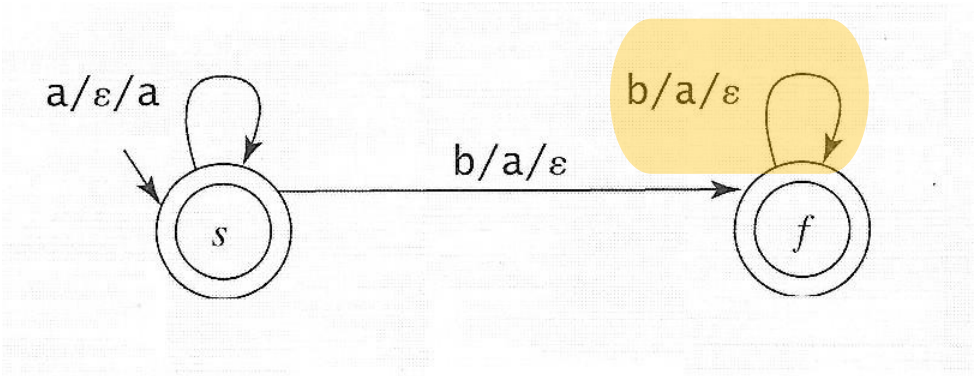
# Contoh-1 PDA

- $A^nB^n = \{a^nb^n : n \geq 0\}$
- Bahasa $A^nB^n$ diterima oleh $M = (K, \Sigma, \Gamma, \Delta, s, A)$ di mana
  - $K = \{s, f\}$ $A = \{s, f\}$
  - $\Sigma = \{a, b\}$ $\Gamma = \{a\}$
  - $\Delta = \{((s, a, \varepsilon), (s, a)), ((s, b, a), (f, \varepsilon)), ((f, b, a), (f, \varepsilon))\}$



# Contoh-1 PDA

- $A^nB^n = \{a^nb^n : n \geq 0\}$
- Bahasa $A^nB^n$ diterima oleh $M = (K, \Sigma, \Gamma, \Delta, s, A)$ di mana
  - $K = \{s, f\}$ $A = \{s, f\}$
  - $\Sigma = \{a, b\}$ $\Gamma = \{a\}$
  - $\Delta = \{((s, a, \varepsilon), (s, a)), ((s, b, a), (f, \varepsilon)), ((f, b, a), (f, \varepsilon))\}$



# Konfigurasi PDA

- Konfigurasi dari PDA $M$ adalah elemen dari $K \times \Sigma^* \times \Gamma^*$, yang secara berurutan menunjukkan:
  - status saat ini (*current state*)
  - input yang belum dibaca
  - isi *stack*
- Konfigurasi awal (*initial configuration*) dari PDA $M$ untuk string input $w$ adalah $(s, w, \varepsilon)$

# Relasi Yield-in-One-Step

- Seperti pada FSM, pada PDA juga dapat didefinisikan relasi *yield-in-one-step*, $\vdash_M$
- Misalkan $c \in \Sigma \cup \{\varepsilon\}$, $\gamma_1, \gamma_2, \gamma \in \Gamma^*$, $w \in \Sigma^*$

  $(q_1, cw, \gamma_1\gamma) \vdash_M (q_2, w, \gamma_2\gamma)$ **iff** $((q_1, c, \gamma_1), (q_2, \gamma_2)) \in \Delta$
- Transisi $((q_1, c, \gamma_1), (q_2, \gamma_2))$ dapat digambarkan pada diagram sebagai label $\boldsymbol{c / \gamma_1 / \gamma_2}$
  - *M* hanya mengambil transisi jika string $\gamma_1$ bersesuaian dengan elemen teratas pada *stack*. *M pop* $\gamma_1$ kemudian *push* $\gamma_2$
  - Jika $\gamma_1 = \varepsilon$, maka *M* tidak perlu mengecek status *stack* saat ini. TIDAK sama artinya dengan *stack* dalam keadaan kosong.

# Komputasi oleh PDA

- Jika $C$ adalah komputasi yang dilakukan oleh $M$ pada input $w \in \Sigma^*$
  - $C$: *accepting computation* **iff** $C = (s, w, \varepsilon) \vdash_{M^*} (q, \varepsilon, \varepsilon)$ untuk suatu $q \in A$

# Komputasi oleh PDA

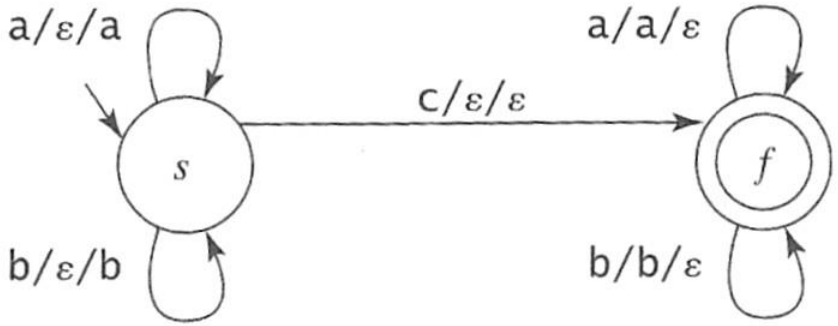
- Jika $C$ adalah komputasi yang dilakukan oleh $M$ pada input $w \in \Sigma^*$
  - $C$: *accepting computation* **iff** $C = (s, w, \varepsilon) \vdash_{M^*} (q, \varepsilon, \varepsilon)$ untuk suatu $q \in A$
  - $C$: *rejecting computation* **iff** $C = (s, w, \varepsilon) \vdash_{M^*} (q, w', \alpha)$ di mana $C$ bukan *accepting computation* dan tidak ada *move* yang mungkin dari $(q, w', \alpha)$, dengan kata lain terjadi *crash* (*halt* tanpa menerima).

# Komputasi oleh PDA

- Jika $C$ adalah komputasi yang dilakukan oleh $M$ pada input $w \in \Sigma^*$
  - $C$: *accepting computation* **iff** $C = (s, w, \varepsilon) \vdash_{M^*} (q, \varepsilon, \varepsilon)$ untuk suatu $q \in A$
  - $C$: *rejecting computation* **iff** $C = (s, w, \varepsilon) \vdash_{M^*} (q, w', \alpha)$ di mana $C$ bukan *accepting computation* dan tidak ada *move* yang mungkin dari $(q, w', \alpha)$, dengan kata lain terjadi *crash* (*halt* tanpa menerima).
- $M$ menerima (*accept*) $w$ **iff** paling tidak satu komputasi menerimanya. $M$ menolak (*reject*) $w$ **iff** semua komputasi menolaknya.
- Semua string $w$ yang diterima oleh $M$, notasi $L(M)$, disebut bahasa yang diterrima $M$.

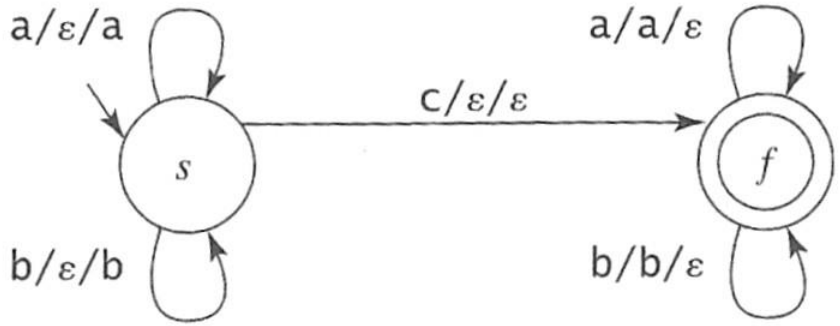
# Contoh-2 dan Contoh-3 PDA

- $WcW^R = \{wcw^R: \ w \in \{a, b\}^*\}$

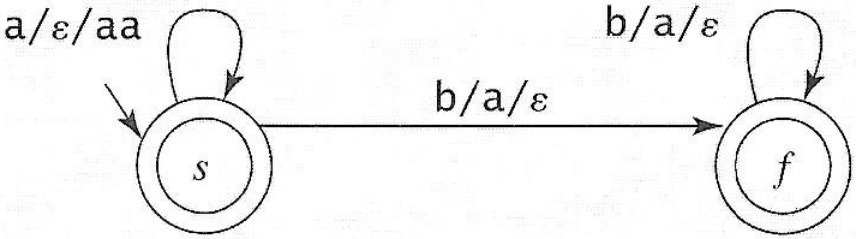


# Contoh-2 dan Contoh-3 PDA

- $WcW^R = \{wcw^R: \ w \in \{a, b\}^*\}$




- $A^nB^{2n} = \{a^nb^{2n}: n \geq 0\}$



# Latihan Membuat PDA

- Buatlah PDA untuk menerima Bahasa
  - $\{a^i b^j c^k : i+j=3k\}$
    - Hint: setiap input a push **satu** simbol, kemudian setiap input b juga push **satu** simbol, setiap input c pop **tiga** simbol. Note: urutan deretan a kemudian deretan b.
  - $\{a^i b^j c^k : i+k=j\}$
    - Hints: terdapat konkatenasi antara dua self embedding $A^n B^n$ dan $B^m C^m$, jadi saat membaca b dari bagian pertama melakuan pop, setelah stack kosong, beralih ke bagian kedua dan setiap membaca b melakukan push.
  - $\{w \in \{a, b\}^* : \#_a(w) = \#_b(w)\}$
    - Hints: Terdapat multiple self-embedding, kadang-kadang a push dan b pop, atau a pop dan b push.

# Menguji dengan JFlap

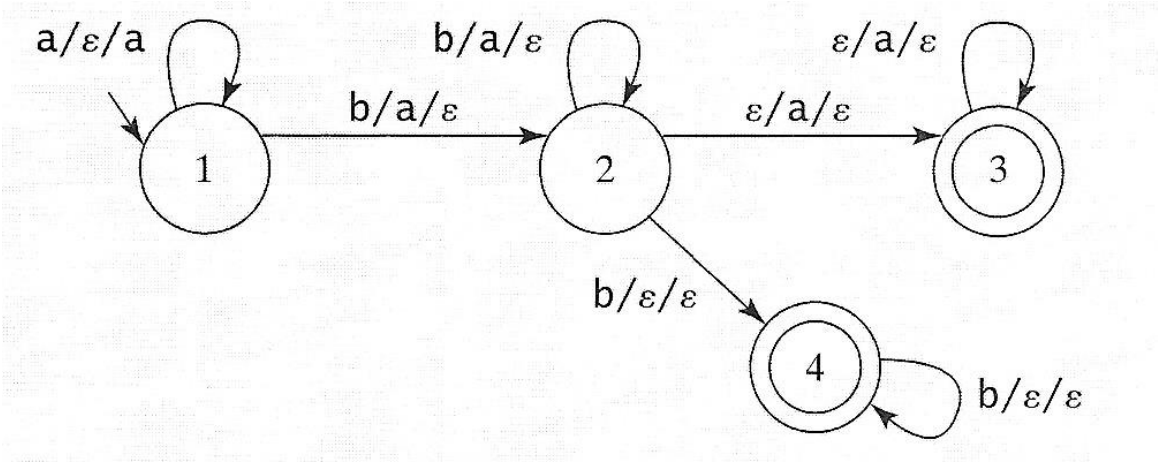
- Anda dapat menguji hasil kerja anda dengan menggambarkannya dengan JFlap dan mensubmit ke server aren (akan disediakan gradernya).
- Catatan khusus versi PDA di JFlap:
  - JFlap menginisialisasi stack dengan suatu simbol khusus yang dinamakan buttom-of-stack symbol (jadi abaikan itu karena grader nanti dibuat tanpa itu).
  - JFlap memberi opsi konfigurasi menerima mesin dapat dengan stack kosong atau tidak harus kosong. Karena akan diuji di grader maka buat sesuai dengan konsep yang diajarkan saja.
- Kecuali, kalua mau menguji di JFlap sendiri maka harus disesuaikan dengan JFlap.

# PDA Deterministik

- PDA *M* adalah deterministik **iff** tidak terdapat konfigurasi *M* yang memiliki pilihan lebih dari satu transisi.
  1. $\Delta_M$ tidak mengandung sekelompok transisi yang berkompetisi satu sama lain
  2. Jika $q \in A$, tidak terdapat transisi $((q, \varepsilon, \varepsilon), (p, x))$. Transisi pada *accepting state M* deterministik harus membaca input atau mengeluarkan isi (*pop*) *stack*.
- PDA deterministik hanya memiliki *path* komputasi tunggal untuk setiap string.

# Contoh-4 PDA

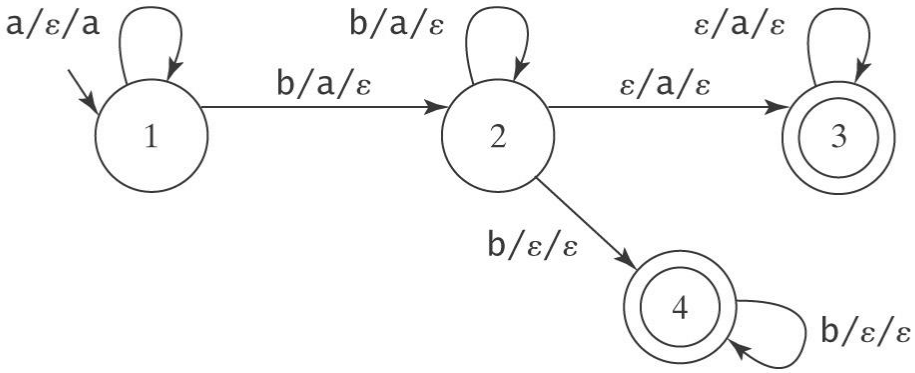
- $L = \{a^m b^n : m \neq n;\ m, n > 0\}$
- $L$ dipecah menjadi dua *sub-language* $\{a^m b^n : 0 < m < n\}$ dan $\{a^m b^n : 0 < n < m\}$




- Apakah PDA di atas deterministik?

# Contoh-4 PDA (contd)

$L = \{a^m b^n : m \neq n; m, n > 0\}$




- Sampai state 2 masih deterministik
- Dari state 2, PDA memiliki 3 pilihan langkah, self loop ke 2, atau ke state 3 atau ke state 4.
- Nondeterministik!

# Konversi ND ke D?

- Non determinisme pada PDA:
  - Transisi yang tidak melakukan pop dengan asumsi ***stack* kosong** berkompetisi dengan transisi normal lain.
  - Transisi yang tidak melakukan pembacaan simbol input dengan asumsi **input habis** yang berkompetisi berkompetisi dengan transisi normal lain.

# Konversi ND ke D?

- Non determinisme pada PDA:
  - Transisi yang tidak melakukan pop dengan asumsi ***stack* kosong** berkompetisi dengan transisi normal lain.
    - Dapat ditangani dengan menambahkan penanda bagian bawah *stack*.
  - Transisi yang tidak melakukan pembacaan simbol input dengan asumsi **input habis** yang berkompetisi berkompetisi dengan transisi normal lain.

# Konversi ND ke D?

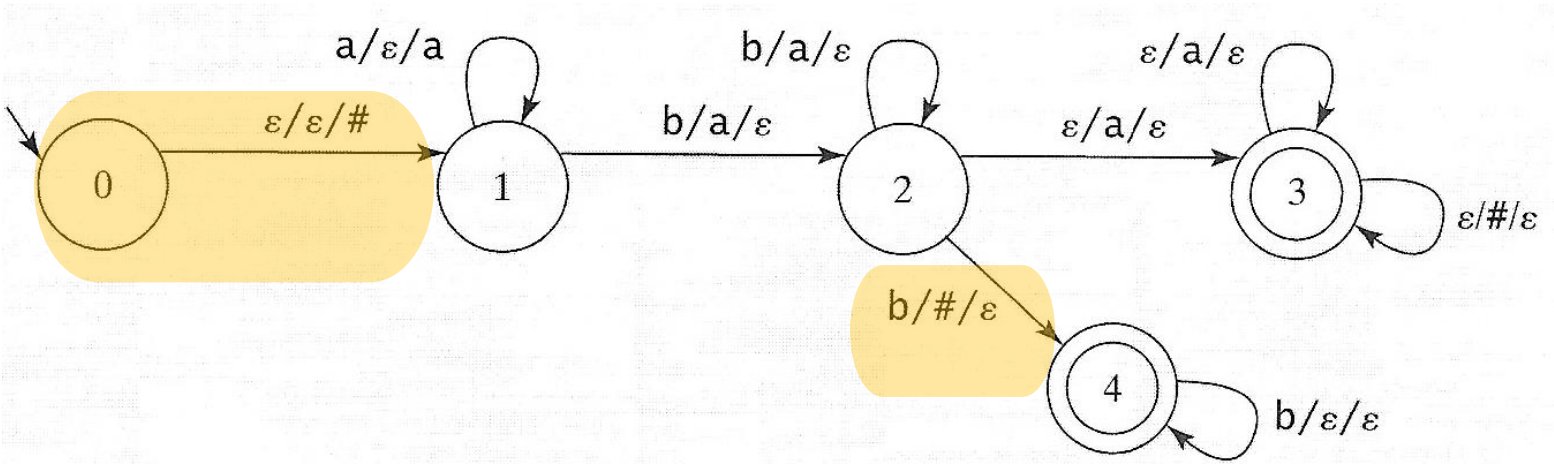
- Non determinisme pada PDA:
  - Transisi yang tidak melakukan pop dengan asumsi ***stack* kosong** berkompetisi dengan transisi normal lain.
    - Dapat ditangani dengan menambahkan penanda bagian bawah *stack*.
  - Transisi yang tidak melakukan pembacaan simbol input dengan asumsi **input habis** yang berkompetisi berkompetisi dengan transisi normal lain.
    - Dapat ditangani dengan menambahkan penanda akhir string.

# Contoh-4 PDA + bottom-of-stack

- $L = \{a^m b^n : m \neq n;\ \ m, n > 0\}$

Mereduksi Non Determinisme

1. Menambahkan karakter penanda bagian bawah *stack*

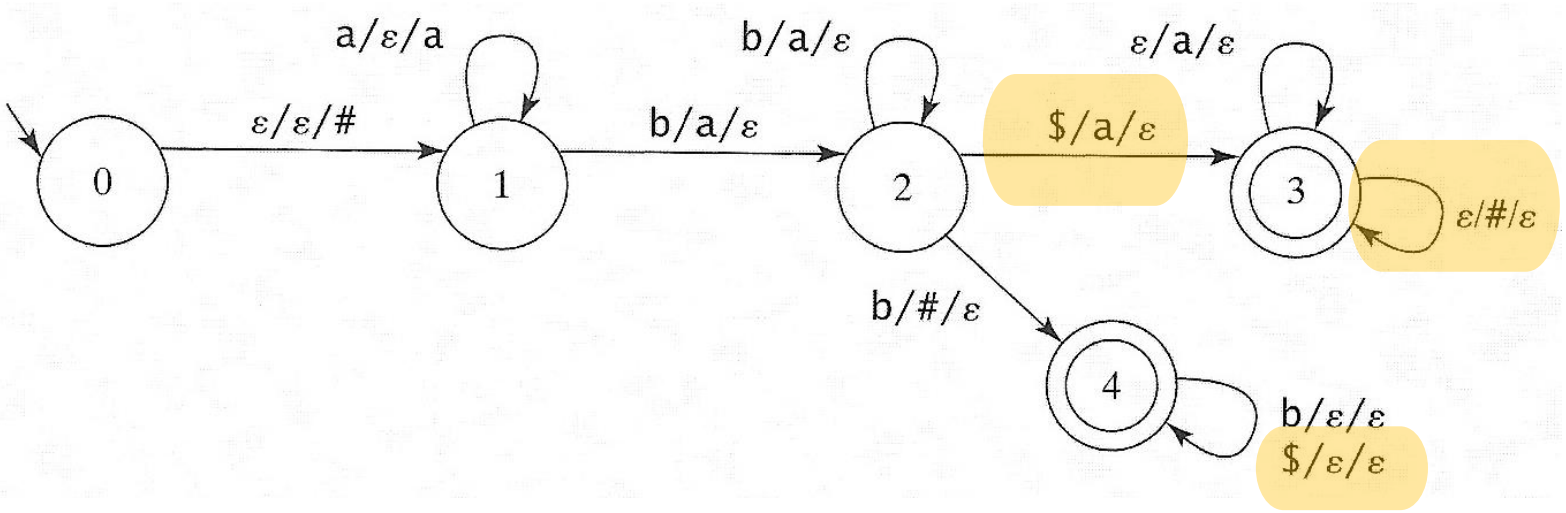


# Contoh-4 PDA (end-of-input)

- $L = \{a^m b^n : m \neq n;\ \ m, n > 0\}$

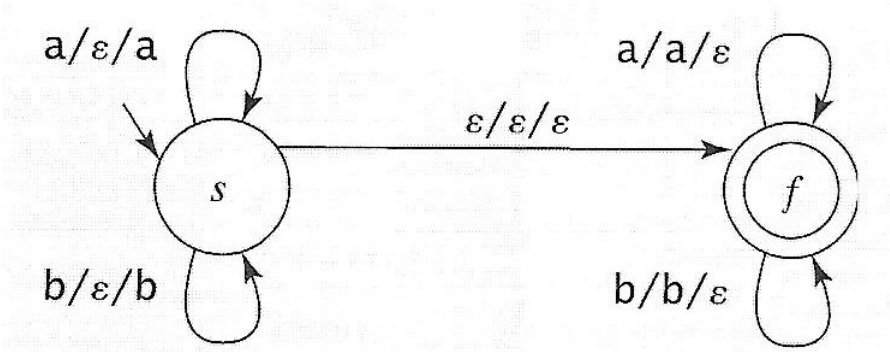
Mereduksi Non Determinisme *(Out of Input Problem)*

2. Menambahkan karakter penanda akhir string



# Contoh-5 PDA

- *PalEven* = $\{ww^R: \ w \in \{a, b\}^*\}$




*M* non deterministik karena tidak tahu kapan akan mencapai tengah input. Setiap membaca input, *M* menebak apakah sudah sampai di tengah input.

# Catatan Tambahan-1

- Penambahan *bottom-of-stack* # **tidak mengubah** Bahasa yang diterima *M*, hanya menambah mekanisme di mesinnya.
- Penambahan *end-of-input* $ **mengubah** bahasa yang diterima *M*, dari semula *L* menjadi *L*$.

# Catatan Tambahan-2

- Apakah semua CFL bisa dibuatkan DPDA?
  - Tidak! Ada bahasa yang sama sekali tidak dapat dibuatkan DPDA-nya.
  - Ada Bahasa yang dapat dibuatkan DPDA-nya jika diberi end-of-line (yaitu menjadi $L\$$).
  - Ada Bahasa yang dapat dibuatkan DPDA-nya tanpa penambahan apa-apa pada input.
- Kedua kelompok terakhir disebut DCFL (akan dibahas di materi DCFL).

# Latihan: Apakah Diterima oleh DPDA (dengan atau tanpa E-O-I)?

- $\{a^i b^j c^k : i+j=3k\}$
- $\{a^i b^j c^k : i+k=j\}$
- $\{w \in \{a, b\}^* : \#_a(w) = \#_b(w)\}$
- $\{a^i b^j : i = j \text{ atau } j = 0\}$
- $\{a^i b^j : j \leq i \leq 2j\}$
- $\{wcw^R : w \in \{a, b\}^*\}$
- $\{w \in \{a, b\}^* : w = w^R\}$ Palindrom (All)