Push Down Automata (PDA)

Teori Bahasa dan Automata

Semester Ganjil 2013

Jum'at, 13.12.2013

Dosen pengasuh:

Kurnia Saputra ST, M.Sc

Email: kurnia.saputra@gmail.com

Jurusan Informatika

Fakultas Matematika dan Ilmu Pengetahuan Alam

Universitas Syiah Kuala

Push Down Automata

Model automata apa yang digunakan pada bahasa context free?

Berbeda dengan bahasa regular, pada bahasa context free model automata menggunakan **Push Down Automata (PDA)**, dimana automata ini menggunakan **stack**.

Push Down Automata

Jika diketahui bahasa sebagai berikut:

$$L = \{w\$w^R \mid w \in \{a, b, c, d\}^*\}$$

dimana $\Sigma = \{a, b, c, d, \$\}$.
Word w^R artinya reverse/kebalikan, contoh: $(abc)^R = cba$.

Push Down Automata

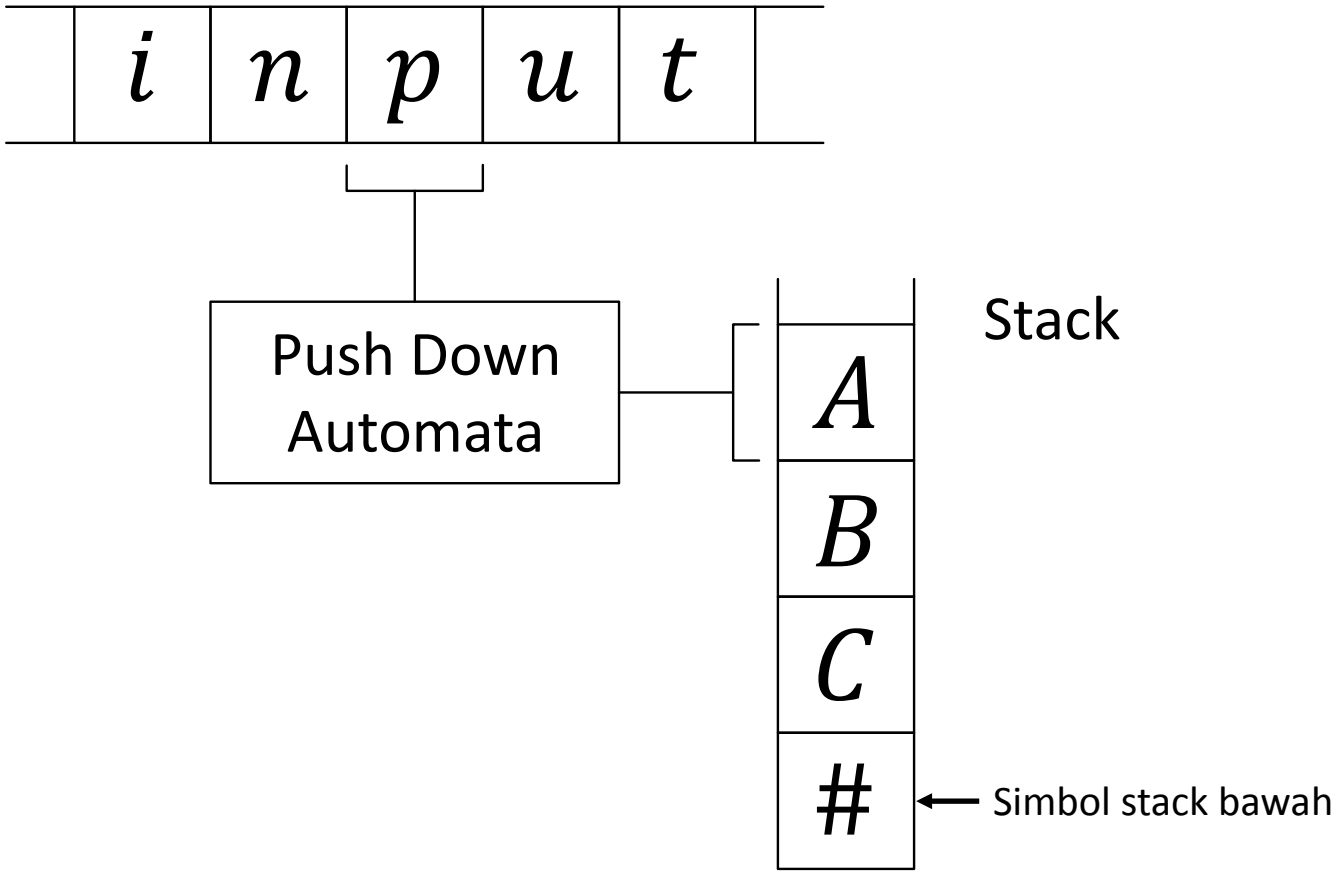
Agar dapat menggunakan model automata pada bahasa context free, maka diperlukan:

- Sebuah stack atau memori push down yang dapat menyimpan sederetan simbol dengan panjang yang sebarang dan tak berhingga.
- Selama proses pembacaan simbol pada stack, simbol teratas pada sebuah stack PDA memiliki kemungkinan sebagai berikut:
  - Stack tidak dapat diubah, atau
  - Simbol pada stack teratas akan dihapus (pop) dan digantikan dengan simbol yang lain (push).

Juga ada kemungkinan stack tidak dapat dibaca.

Push Down Automata

Skema push down automata:



Push Down Automata

Jika diketahui bahasa sebagai berikut:

$$L = \{w\$w^R \mid w \in \{a, b, c, d\}^*\}$$

dimana $\Sigma = \{a, b, c, d, \$\}$.

Push down automata akan mengenal sebuah bahasa dengan cara sebagai berikut:

- Sebuah word w dibaca dari kiri ke kanan.
- Sebuah automata memiliki dua buah state:
  State 1: Menyimpan bagian pertama dari word.
  State 2: Mengecek bagian kedua dari word.

Push Down Automata

- State 1: Selama simbol $ belum dibaca, masukkan (push) simbol kapital (huruf besar) untuk setiap simbol yang dibaca ke dalam stack $(a \rightsquigarrow A, b \rightsquigarrow B, \dots)$. Jika simbol $ dibaca, maka stack tidak terjadi perubahan dan akan berpindah ke state 2.

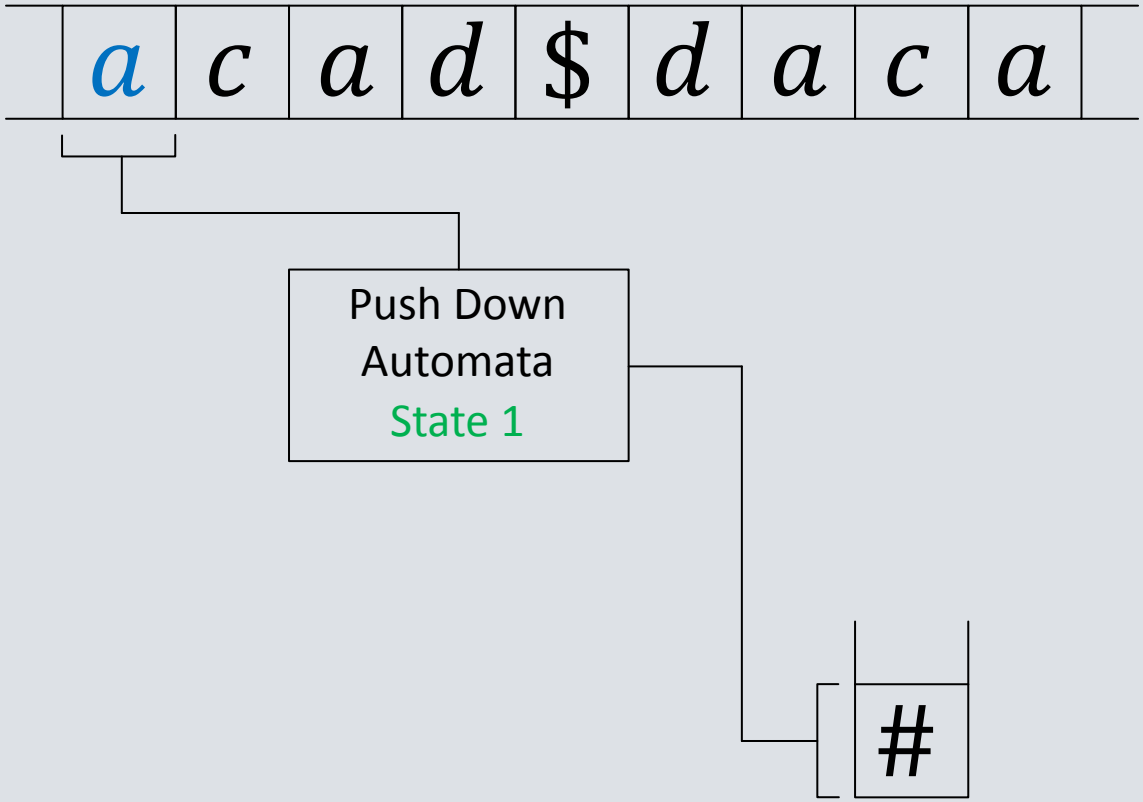
- State 2: Lakukan pengecekan untuk setiap simbol yang dibaca, apakah ada kesesuaian dengan simbol kapital yang ada pada stack teratas. Jika ada, simbol kapital ini harus dihapus (pop) dari stack teratas.

  Jika antara simbol yang dibaca dengan simbol kapital pada stack tidak ada kesesuaian, maka tidak terjadi transisi. Push down automata akan melakukan blok dan word tidak bisa diterima.

Jika pada state 2 selalu terjadi kesesuaian antara simbol yang dibaca dengan simbol kapital pada stack teratas, maka simbol terbawah dari stack # akan dihapus dan push down automata akan menerima word tersebut ketika stack telah kosong.

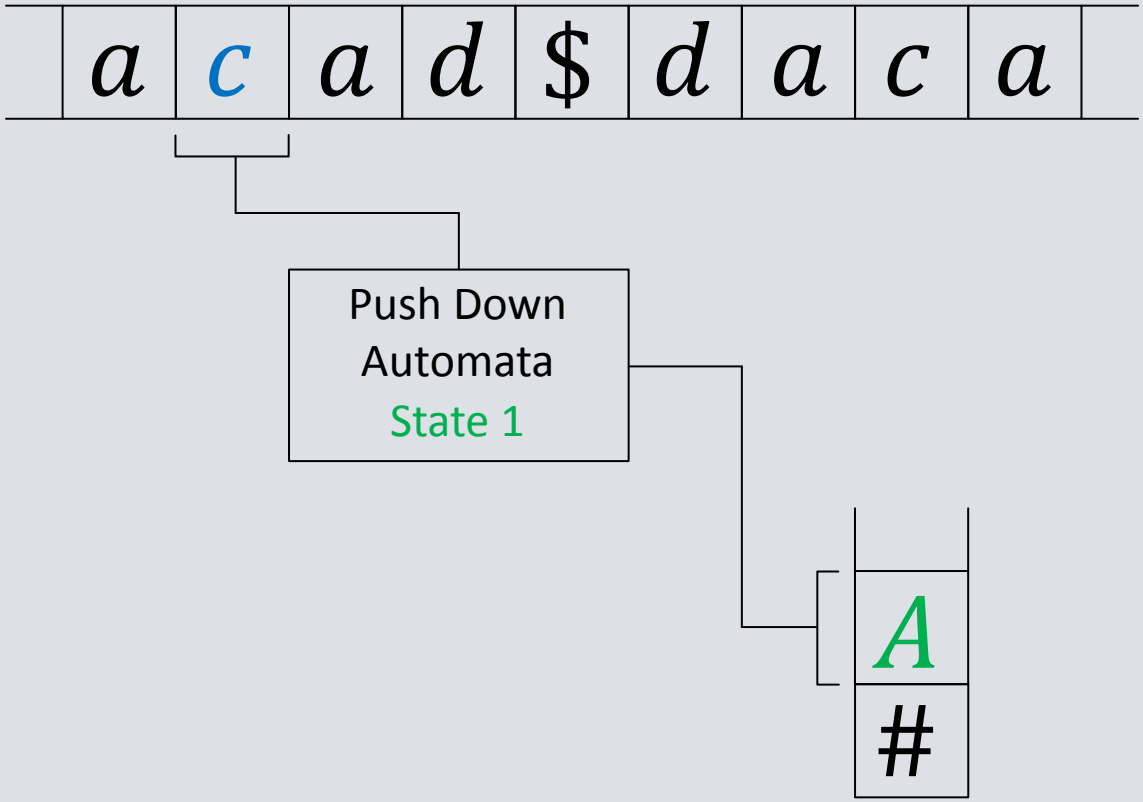
Push Down Automata

Simulasi



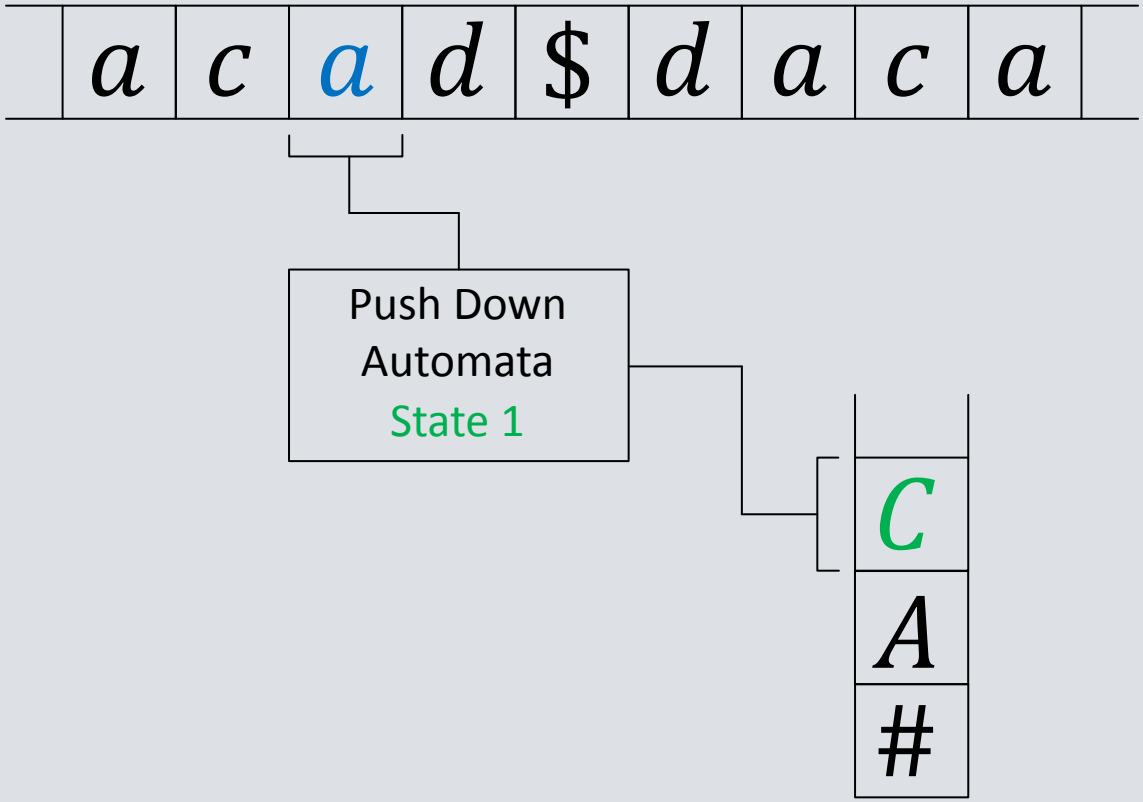
Push Down Automata

Simulasi



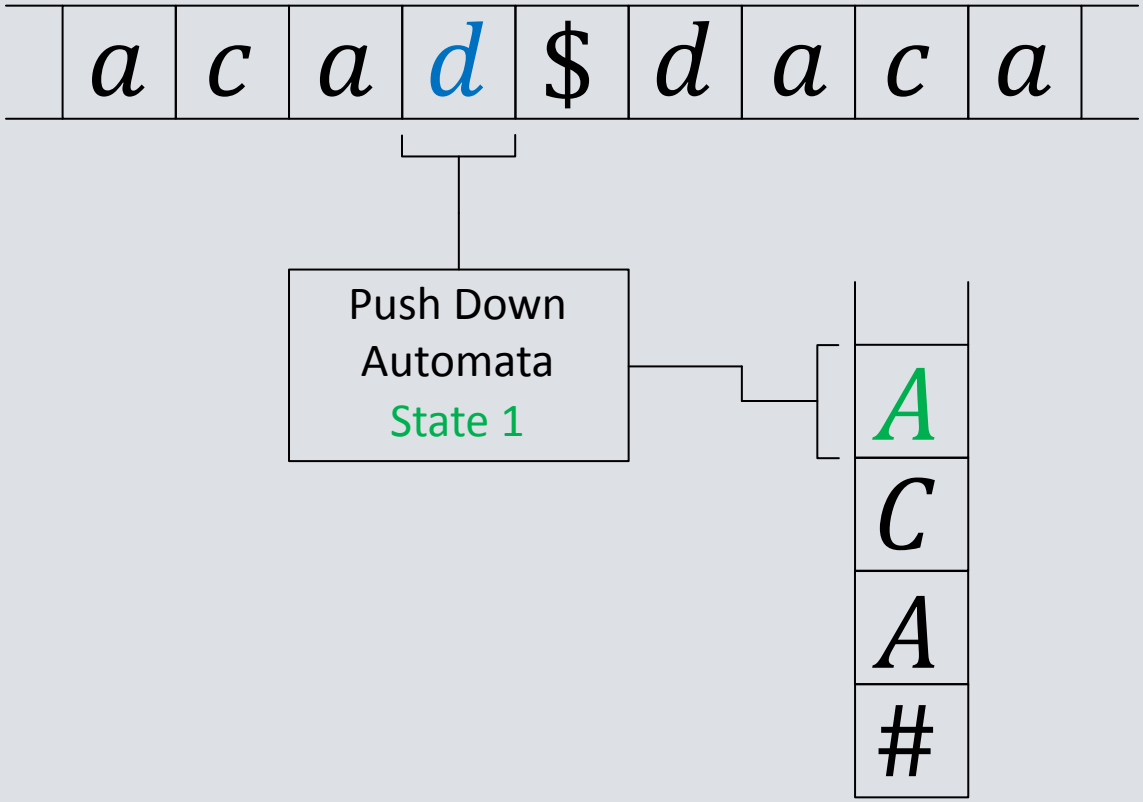
Push Down Automata

Simulasi



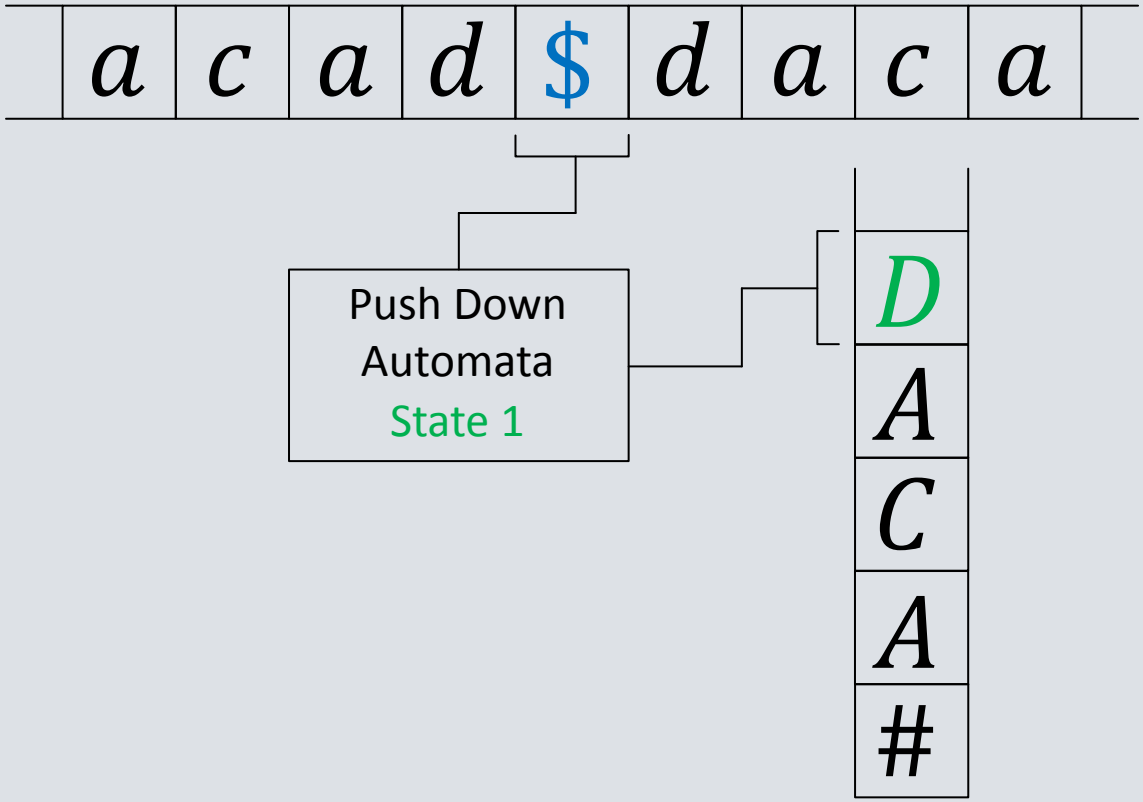
Push Down Automata

Simulasi



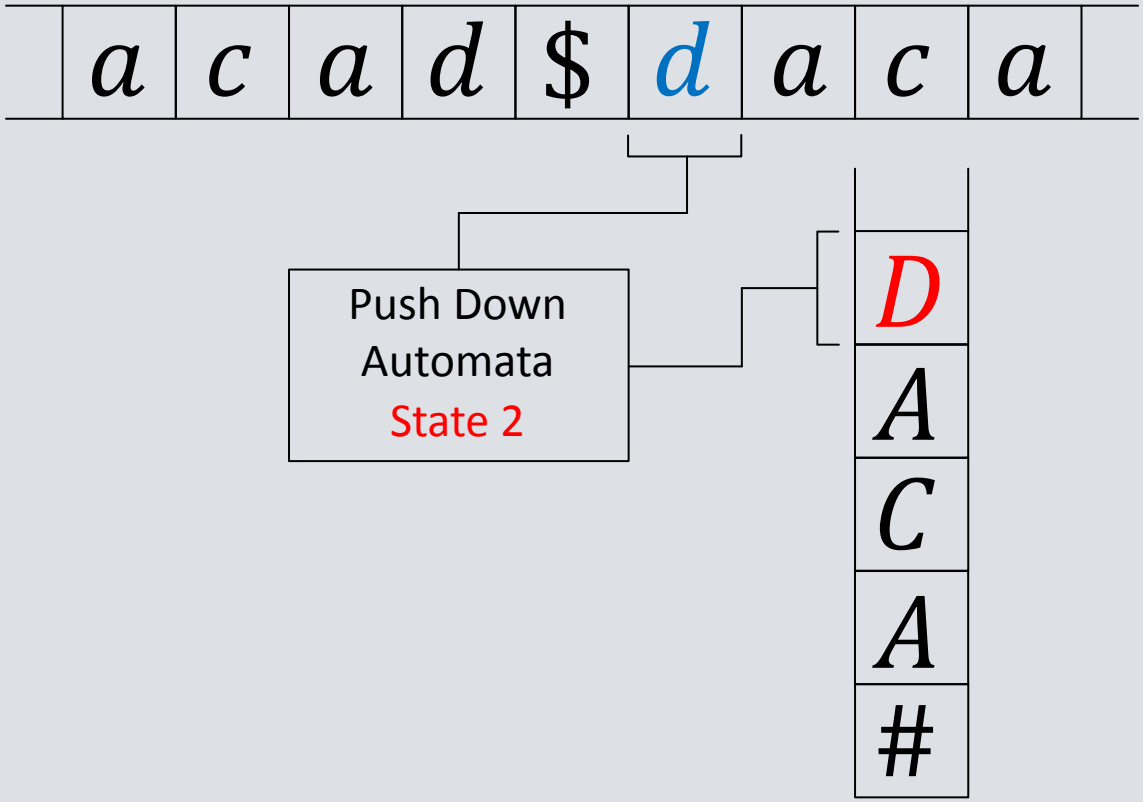
Push Down Automata

Simulasi



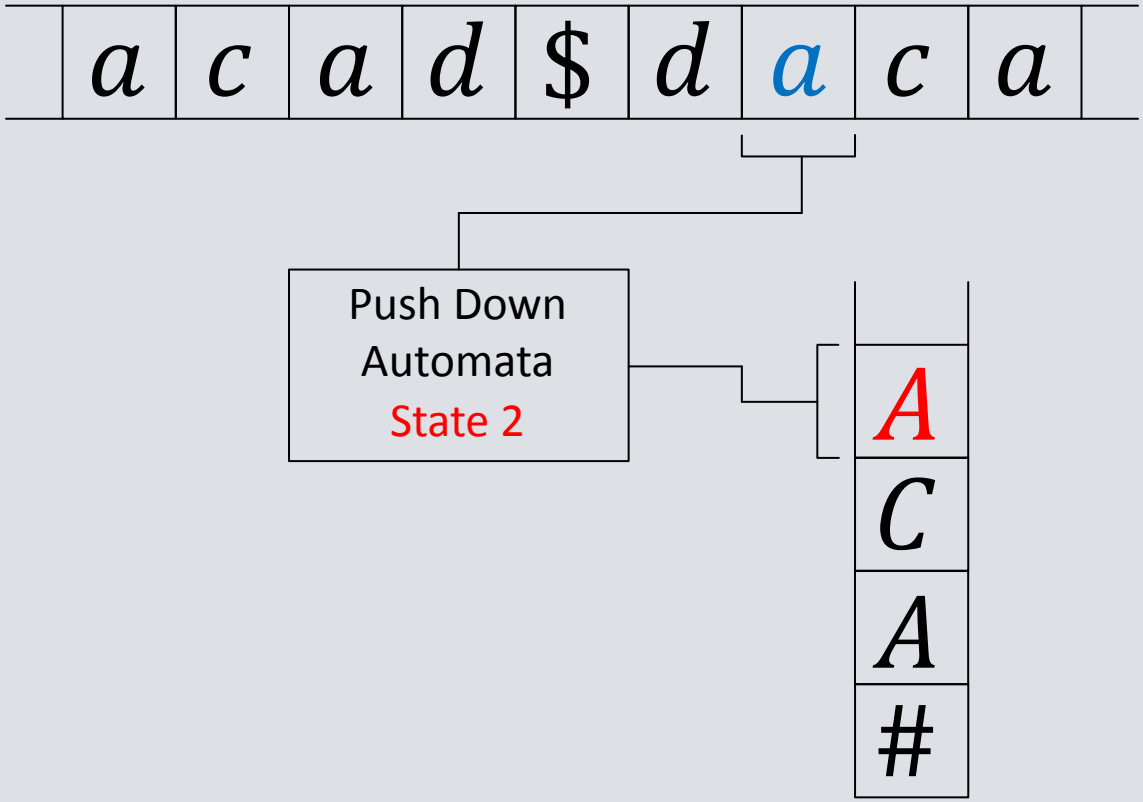
Push Down Automata

Simulasi



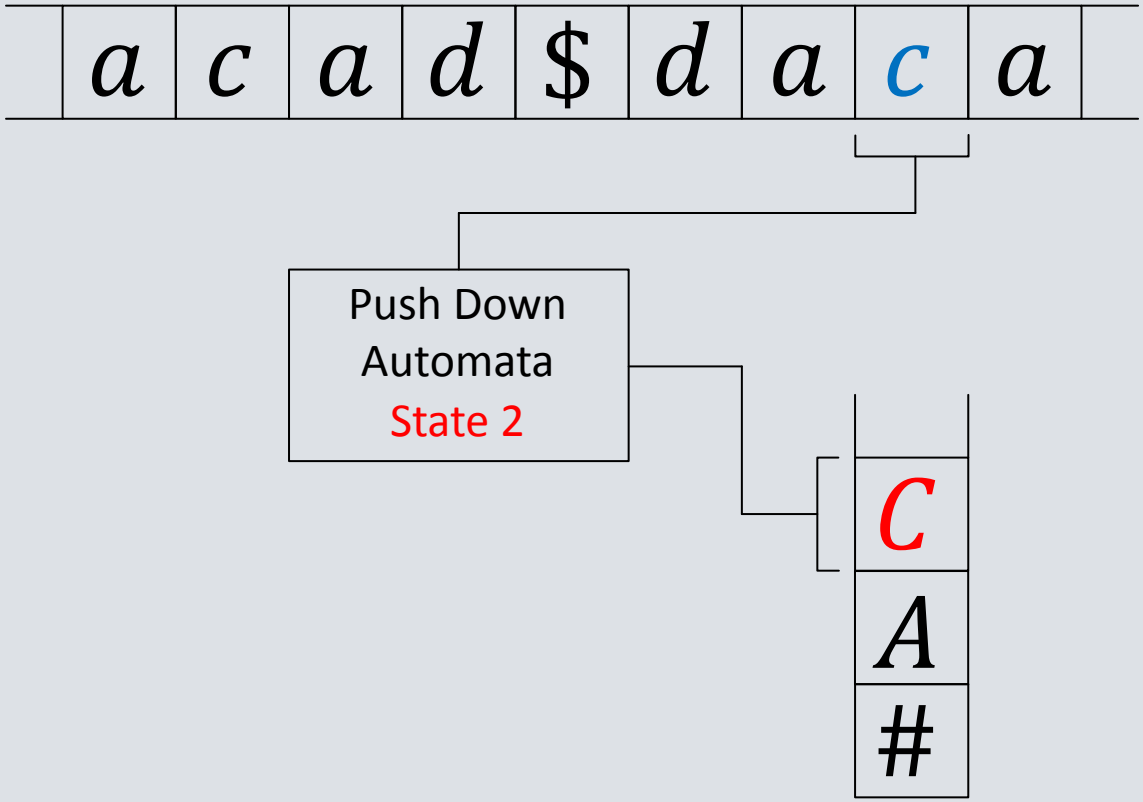
Push Down Automata

Simulasi



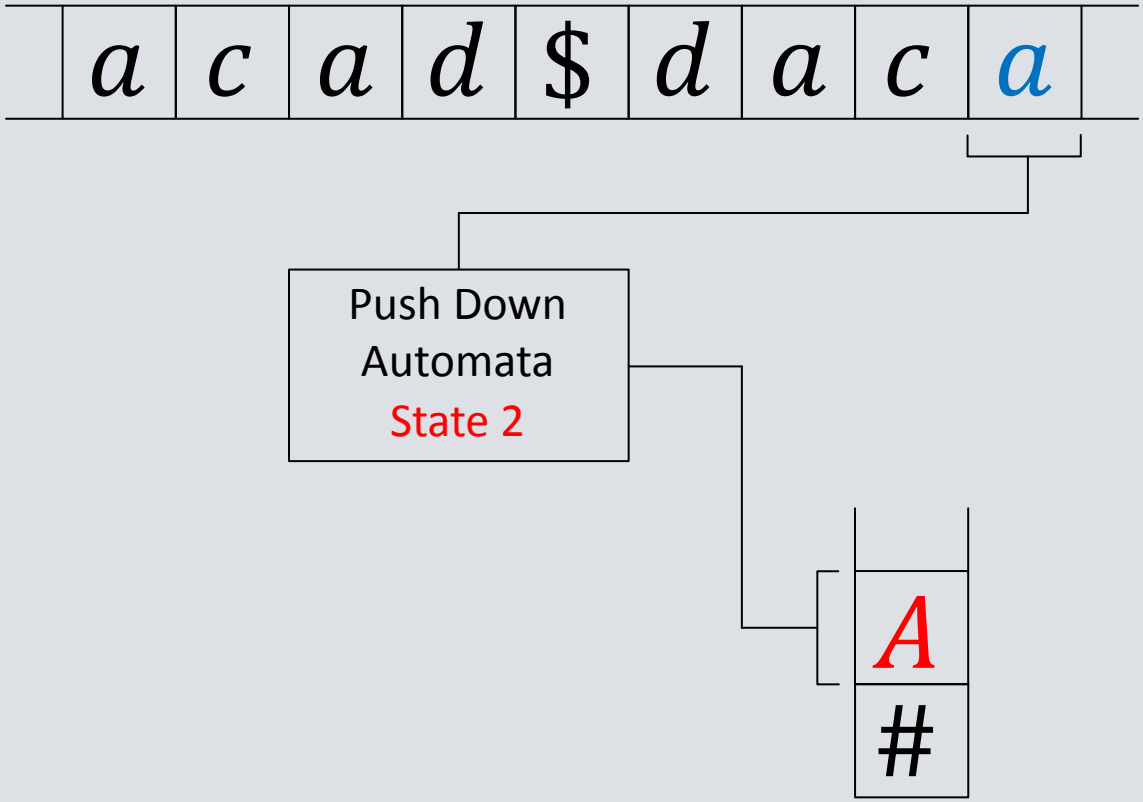
Push Down Automata

Simulasi



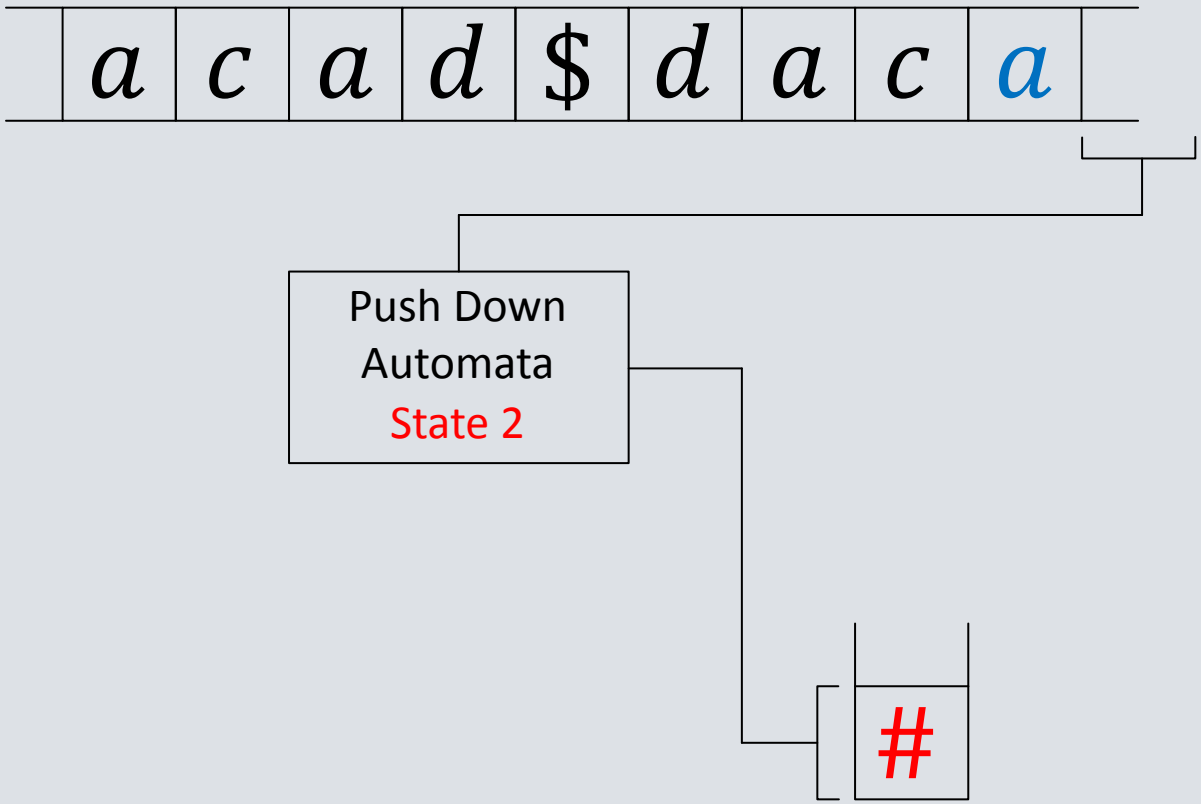
Push Down Automata

Simulasi



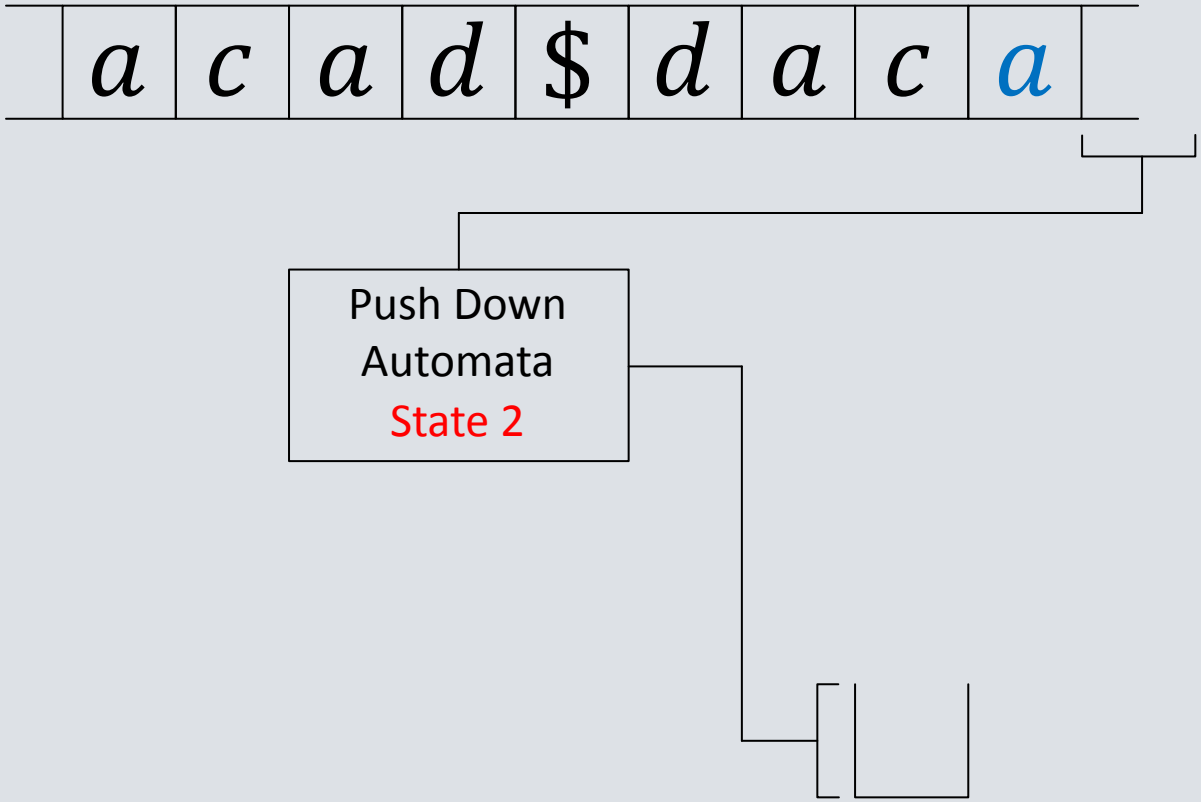
Push Down Automata

Simulasi



Push Down Automata

Simulasi



Push Down Automata

Definisi Push Down Automata

Push Down Automata non-deterministik M terdiri dari enam elemen

$M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$, dimana:

- Z adalah himpunan state,
- Σ adalah alphabet input (dimana $Z \cap \Sigma = \emptyset$),
- Γ adalah alphabet stack,
- $z_0 \in Z$ adalah initial state,
- $\delta: Z \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \longrightarrow \mathcal{P}_e(Z \times \Gamma^*)$ adalah fungsi transisi dan
- $\# \in \Gamma$ adalah simbol stack terbawah.

Catatan:

- Z, Σ harus finite.
- $\mathcal{P}_e(Z \times \Gamma^*)$ adalah himpunan semua finite dari $Z \times \Gamma^*$.

Push Down Automata

Diketahui fungsi transisi sebagai berikut:

$$\delta = Z \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \longrightarrow \mathcal{P}_e(Z \times \Gamma^*)$$

Jika $(z', B_1, \dots B_k) \in \delta(z, a, A)$, ini berarti:

- Ketika simbol input a dibaca pada state z dan simbol A berada pada stack teratas, maka
- Simbol A dihapus dari stack dan digantikan dengan $B_1, \dots B_k$ (B_1 sekarang adalah simbol teratas pada stack) dan automata berpindah ke state z'.

Ada juga kemungkinan $a = \varepsilon$. Jika kasus ini terjadi berarti tidak ada simbol input yang dibaca.

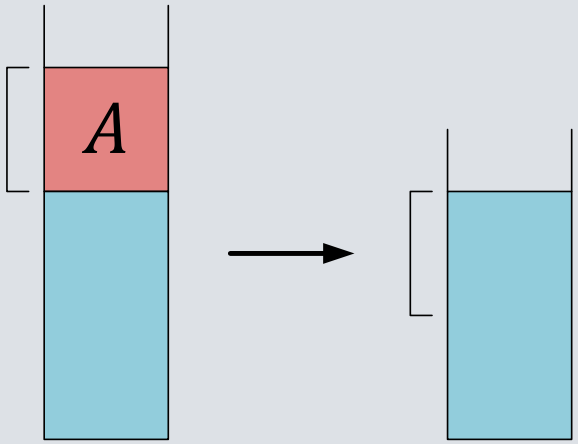
Push Down Automata

Beberapa contoh kasus fungsi transisi δ pada PDA:

Contoh Kasus 1:

$(z', \varepsilon) \in \delta(z, a, A)$

- Simbol a dibaca
- State berubah dari z ke z'
- Simbol A dihapus dari stack:



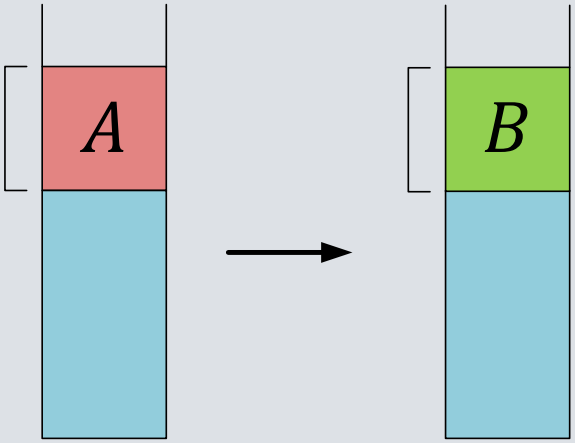
Push Down Automata

Beberapa contoh kasus fungsi transisi δ pada PDA:

Contoh Kasus 2:

$(z', B) \in \delta(z, a, A)$

- Simbol a dibaca
- State berubah dari z ke z'
- Simbol A pada stack diganti dengan B:



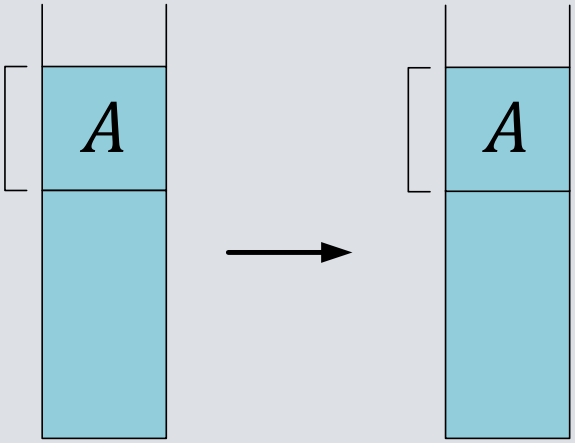
Push Down Automata

Beberapa contoh kasus fungsi transisi δ pada PDA:

Contoh Kasus 3:

$(z', A) \in \delta(z, a, A)$

- Simbol a dibaca
- State berubah dari z ke z'
- Simbol A tetap berada pada stack:



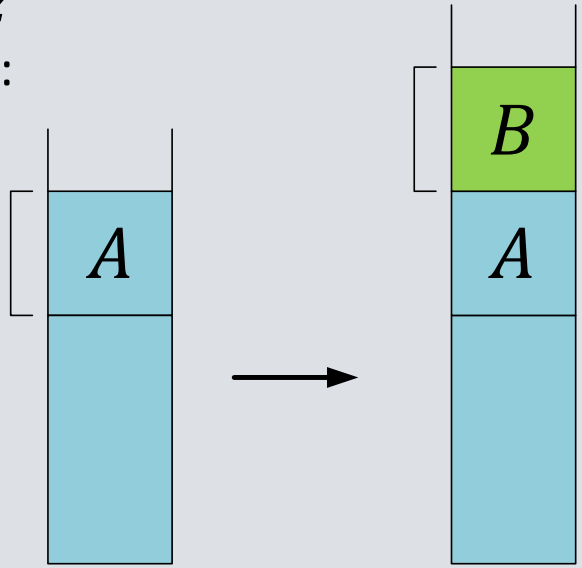
Push Down Automata

Beberapa contoh kasus fungsi transisi δ pada PDA:

Contoh Kasus 4:

$(z', BA) \in \delta(z, a, A)$

- Simbol a dibaca
- State berubah dari z ke z'
- Simbol B masuk ke stack:



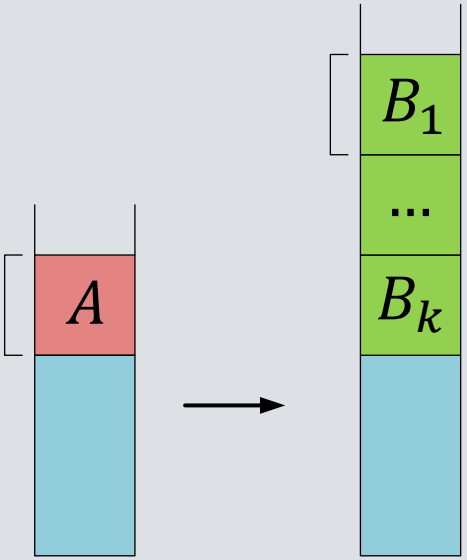
Push Down Automata

Beberapa contoh kasus fungsi transisi δ pada PDA:

Contoh Kasus 5:

$(z', B_1 \dots B_k) \in \delta(z, a, A)$

- Simbol a dibaca
- State berubah dari z ke z'
- Simbol A diganti dengan banyak simbol:



Push Down Automata

- Di setiap awal perhitungan, pada stack terbawah akan selalu terdapat simbol #.
- Stack bersifat tidak terbatas (unbounded), artinya terdapat kemungkinan yang tidak berhingga pada stack dalam membaca simbol. Inilah yang membedakan push down automata dengan finite automata.
- Push down automata akan selalu dapat menerima stack kosong, yang berarti menandakan state final.

Push Down Automata

Reperesentasi pada gambar:

State: z

Intial State: $\rightarrow z_0$

Transisi: $z_1 \longrightarrow z_2$ dengan label $a, X \longrightarrow \gamma$

- a: Simbol yang dibaca
- X: Simbol teratas pada stack
- γ: Perubahan stack

Push Down Automata

Contoh:

Berikan push down automata M untuk bahasa berikut:

$$L(M) = \{w\$w^R \mid w \in \{a, b\}^*\}$$

Push Down Automata

Definisi konfigurasi PDA

Konfigurasi pada push down automata adalah:

$$k \in Z \times \Sigma^* \times \Gamma^*$$

Arti dari komponen $k = (z, w, \gamma) \in Z \times \Sigma^* \times \Gamma^*$ adalah:

- $z \in Z$ adalah posisi state yang sedang berjalan pada PDA.
- $w \in \Sigma^*$ adalah input yang akan dibaca oleh PDA.
- $\gamma \in \Gamma^*$ adalah simbol stack yang sedang berjalan pada PDA. Simbol teratas stack diletakkan pada sisi kiri.

Push Down Automata

Transisi konfigurasi pada PDA terjadi dari fungsi δ transisi berikut:

Definisi konfigurasi PDA

Berlaku kondisi:

$$(z, aw, A\gamma) \vdash (z', w, B_1 \dots B_k\gamma)$$

jika $(z', B_1 \dots B_k) \in \delta(z, a, A)$ dimana

$$(z, w, A\gamma) \vdash (z', w, B_1 \dots B_k\gamma)$$

jika $(z', B_1 \dots B_k) \in \delta(z, \varepsilon, A)$

Pada kasus pertama, simbol dibaca dari input.

Push Down Automata

Definisi bahasa yang dapat diterima

Jika diketahui push down automata $M = (Z, \Sigma, \Gamma, \delta, z_0, \#)$. Maka bahasa yang dapat diterima oleh M adalah:

$$N(M) = \{x \in \Sigma^* \mid (z_0, x, \#)\} \vdash^* (z, \varepsilon, \varepsilon) \text{ dimana } z \in Z\}$$

Jika bahasa yang diterima terdiri dari semua words maka stack PDA akan kosong. Dikarenakan push down automata adalah non-deterministik, ada kemungkinan stack PDA tidak akan kosong.

Push Down Automata

Contoh:

Carilah push down automata untuk bahasa berikut:

$$L = \{ww^R \mid w \in \{a, b\}^*\}$$

Ide: Pada contoh di atas, PDA dapat berpindah ke state z_2 secara non-deterministik tanpa harus menunggu simbol \$. Perpindahan state dapat terjadi dengan mengecek bagian kedua dari simbol input, dimana jika simbol input yang sedang berjalan ada kesesuaian dengan simbol teratas pada stack maka akan terjadi perpindahan state.

Referensi

1. Hopcroft, Motwani, Ullman: *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 2001
2. James A. Anderson: *Automata Theory with Modern Applications*, Cambridge University Press, 2006.
3. Uwe Schöning: *Theoretische Informatik – kurzgefaßt*. Spektrum, 2008. (5. Auflage)