Autômatos de Pilha

Linguagens Formais e Autômatos

Prof. Daniel Saad Nogueira Nunes

IFB – Instituto Federal de Brasília, Campus Taguatinga

Sumário

Sumário

Introdução

- Introduziremos nesta aula um modelo computacional chamado autômato de pilha, também conhecido por *pushdown automata* ou PDA em inglês.
- De certa forma eles são parecidos com autômatos finitos não-determinísticos, mas possuem um componente extra: **uma pilha**.
- A pilha possibilita um uso de memória adicional que, apesar de poder ser utilizada de uma maneira restrita, consegue resolver algumas linguagens.
- De fato, os PDAs são equivalentes em poder computacional as CFGs, e portanto, reconhecem as **linguagens livres-de-contexto**.

Sumário

PDA

- Os PDAs podem armazenar símbolos na pilha (*push*) e desempilhá-los em um momento oportuno (*pop*).
- O acesso a uma pilha só pode ser realizado no elemento do topo e funciona conforme a ordem LIFO (*last-in-first-out*), em que o último elemento inserido na pilha é aquele que ocupa o topo. Apenas o elemento do topo pode ser desempilhado.

PDA

Estratégia de um PDA para reconhecer $L = \{0^n 1^n \mid n \geq 0\}$

- Leia os símbolos da entrada, cada 0 lido é inserido na pilha.
- Assim que os 1s começarem a serem vistos, desempilhe um 0 para cada 1 lido.
- Se a entrada terminou no mesmo momento que a pilha ficou vazia, então aceite a entrada. Caso a pilha tenha ficado vazia enquanto há 1s a serem lidos ou a pilha ainda tenha 0s após o final da entrada, rejeite-a.

PDA: não-determinismo

- Um PDA é não-determinístico.
- Ao contrátio dos autômatos finitos, os PDAs possuem mais poder computacional do que a sua versão determinística.
- O não-determinismo, no caso dos PDAs, possibilita resolver mais problemas do que utilizando o modelo de autômato de pilha determinístico.
- Focamos nos PDAs por eles capturarem a mesma classe de linguagem que as CFGs.

PDA: definição formal

Definição formal

Um PDA é uma 6-tupla $(Q, \Sigma, \Gamma, \delta, q_0, F)$, em que:

- Q é o conjunto finito de estados.
- Σ é o alfabeto de entrada.
- Γ é o alfabeto da pilha.
- $\delta : Q \times \Sigma_\epsilon \times \Gamma_\epsilon \to \mathcal{P}(Q \times \Gamma_\epsilon)$ é a função de transição. No caso $\Sigma_\epsilon = \Sigma \cup \{\epsilon\}$ e $\Gamma_\epsilon = \Gamma \cup \{\epsilon\}$.
- $q_0 \in Q$ é o estado inicial.
- $F \subseteq Q$ é o conjunto de estados de aceitação.

PDA

Computação em PDA

A computação em um PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ aceita uma entrada w se pode ser escrita como $w = w_1 w_2 \dots w_m$, com $w_i \in \Sigma_\epsilon$ e existem sequências de estados $r_0, r_1, \dots, r_m \in Q$ e palavras $s_0, s_1, \dots, s_m \in \Gamma^*$ satisfazendo três condições a seguir. A palavra s_i representa a sequência de conteúdos da pilha que M tem no ramo não-determinístico de aceitação.

PDA

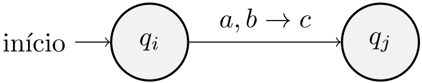
Computação em PDA

1. $r_0 = q_0$ e $s_0 = \epsilon$. Isso representa a configuração inicial de M.
2. Para $i = 0, \ldots, m-1$, temos $(r_{i+1}, b) \in \delta(r_i, w_{i+1}, a)$, em que $s_i = at$ e $s_{i+1} = bt$ para algum $a, b \in \Gamma_\epsilon$ e $t \in \Gamma^*$. Essa condição nos diz que a função de transição é respeitada, trocando um símbolo a no topo da pilha, por um símbolo b no topo da pilha.
3. $r_m \in F$. Essa condição representa uma configuração de aceitação.

PDA

Representação em diagrama

Escrevemos:

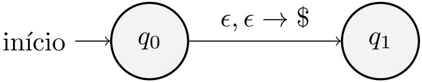


Para indicar que ao ler o símbolo a da entrada, ele troca o símbolo b no topo da pilha para o símbolo c. a, b ou c podem ser ϵ. Se $a = \epsilon$, então o PDA pode fazer a transição sem consumir a entrada. Se $b = \epsilon$, então o PDA pode fazer a transição sem ler e retirar símbolos da pilha. Se $c = \epsilon$, o PDA não escreve símbolos na pilha ao processar a transição. '

PDA

Pequenos truques

- Um PDA não tem um mecanismo explícito para testar se a pilha está vazia.
- Contudo, para emular essa capacidade, podemos, inicialmente, inserir um símbolo $\$$ na pilha.
- Se $\$$ estiver no topo da pilha, então, temos que a pilha está vazia.

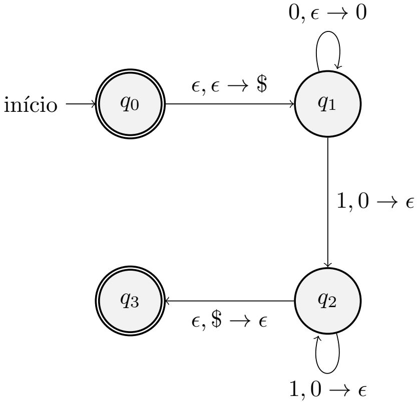


Sumário

$L = \{0^n 1^n \mid n \geq 0\}$

Exemplo

PDA que reconhece a linguagem $L = \{0^n 1^n \mid n \geq 0\}$.



$L = \left\{a^i b^j c^k \mid i, j, k \geq 0 \text{ e } i = j \text{ ou } i = k\right\}$

Exemplo

Projete um PDA que reconheça a linguagem:

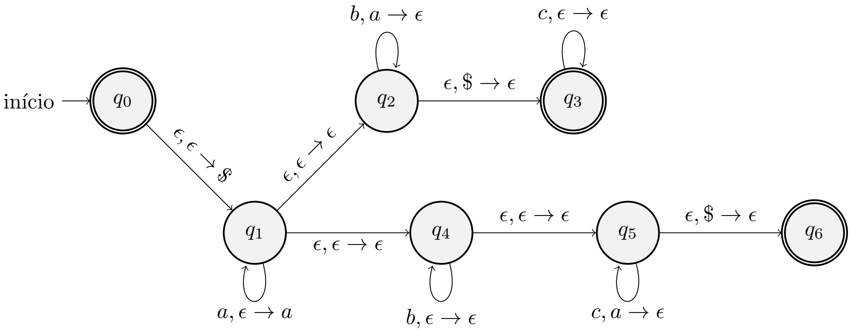
$$L = \left\{a^i b^j c^k \mid i, j, k \geq 0 \text{ e } i = j \text{ ou } i = k\right\}$$

- Para projetar um PDA que reconheça L, precisamos, obrigatoriamente, usar o não-determinismo.
- Não sabemos se temos que casar os a's com os b's ou os a's com os c's. Como não temos a habilidade de voltar na entrada, resta usar o não-determinismo.
- Teremos duas trajetórias no PDA, uma para cada possibilidade.

$L = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ e } i = j \text{ ou } i = k\}$

Exemplo

Projete um PDA que reconheça a linguagem:
$L = \{a^i b^j c^k \mid i, j, k \geq 0 \text{ e } i = j \text{ ou } i = k\}$



$L = \left\{ww^R \mid w \in \{0,1\}^*\right\}$

Exemplo

Projete um PDA que reconheça a linguagem:

$$L = \left\{ww^R \mid w \in \{0,1\}^*\right\}$$

- A ideia é colocar os símbolos na pilha e "adivinhar"' ' quando chegamos ao meio da palavra.
- Em seguida, basta conferir se os símbolos desempilhados estão batendo com os da entrada.
- Como podemos adivinhar o meio da palavra?

$L = \{ww^R \mid w \in \{0,1\}^*\}$

Exemplo

Projete um PDA que reconheça a linguagem:

$$L = \{ww^R \mid w \in \{0,1\}^*\}$$

- A ideia é colocar os símbolos na pilha e "adivinhar"' quando chegamos ao meio da palavra.
- Em seguida, basta conferir se os símbolos desempilhados estão batendo com os da entrada.
- Como podemos adivinhar o meio da palavra?
- Basta usar o **não-determinismo** para testar todas as possibilidades.

$L = \left\{ww^R \mid w \in \{0,1\}^*\right\}$

Exemplo

Projete um PDA que reconheça a linguagem:
$L = \left\{ww^R \mid w \in \{0,1\}^*\right\}$