Algoritmos e Fundamentos da Teoria de Computação

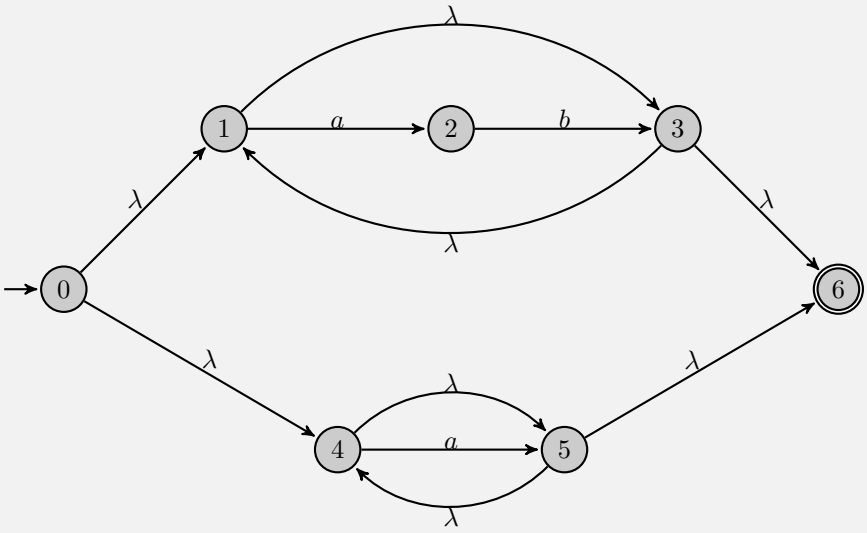
Lista de Exercícios 03

1 Seja a linguagem descrita pela expressão regular $(\boldsymbol{ab})^* \cup \boldsymbol{a}^*$. Pede-se:

a. Apresente um NFA-λ que reconhece a linguagem acima.

b. Usando o algoritmo de determinização (Algoritmo 5.6.3 do livro do Sudkamp), apresente um DFA equivalente ao NFA-λ do item anterior.

a. Segue abaixo o NFA-λ que reconhece a expressão regular pedida.



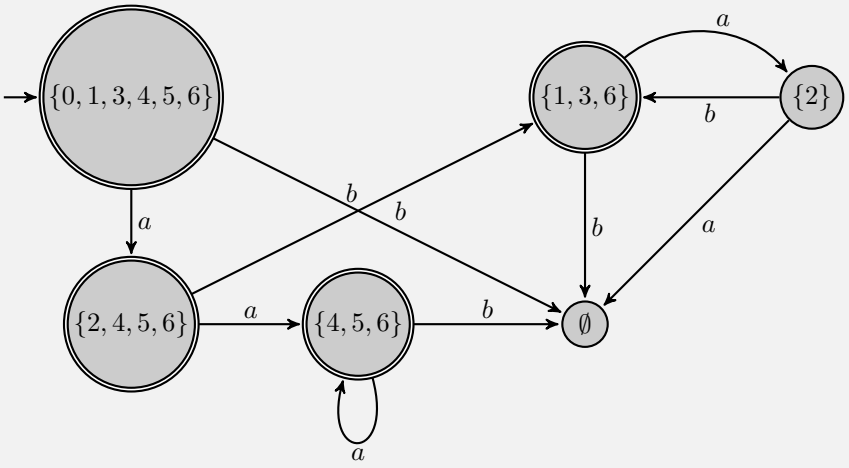
b. Tabela que indica o fecho-λ do NFA do item anterior.

i	0	1	2	3	4	5	6
λ-*closure*(i)	$\{0,1,3,4,5,6\}$	$\{1,3,6\}$	$\{2\}$	$\{1,3,6\}$	$\{4,5,6\}$	$\{4,5,6\}$	$\{6\}$

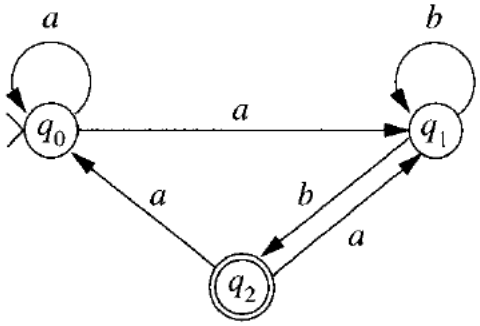
Tabela que indica a função de transição da entrada t.

t	0	1	2	3	4	5	6
a	$\{2,4,5,6\}$	$\{2\}$	$\emptyset$	$\{2\}$	$\{4,5,6\}$	$\{4,5,6\}$	$\emptyset$
b	$\emptyset$	$\emptyset$	$\{1,3,6\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

Autômato determinístico obtido do algoritmo.



2 Seja M o NFA cujo diagrama de estados é dado abaixo. Pede-se:



a. Utilizando a notação $[q_i, w]$, que define a configuração de um autômato em cada passo de computação, descreva as quatro sequências de execução distintas de M para a entrada abb. Essa *string* abb pertence a $\mathsf{L}(\mathsf{M})$? Justifique.

b. Apresente uma expressão regular para $\mathsf{L}(\mathsf{M})$.

a. As quatro sequências de execução possíveis são:

1. $[q_0, abb] \vdash [q_0, bb]$. Trava em q_0.
2. $[q_0, abb] \vdash [q_1, bb] \vdash [q_1, b] \vdash [q_1, \lambda]$. Trava em q_1 após consumir a entrada toda.
3. $[q_0, abb] \vdash [q_1, bb] \vdash [q_1, b] \vdash [q_2, \lambda]$. Para em q_2 após consumir a entrada toda. Aceita.
4. $[q_0, abb] \vdash [q_1, bb] \vdash [q_2, b]$. Trava em q_2 sem consumir a entrada toda.

Por conta da sequência de execução 3, temos que $abb \in \mathsf{L}(\mathsf{M})$.

b. $\mathsf{L}(\mathsf{M}) = (a^+b^+)^+$.

3 Seja M o PDA definido como abaixo.

$$\mathsf{Q} = \{q_0, q_1, q_2\}$$
$$\Sigma = \{a, b\}$$
$$\Gamma = \{A\}$$
$$\mathsf{F} = \{q_1, q_2\}$$

$$\delta(q_0, a, \lambda) = \{[q_0, A]\}$$
$$\delta(q_0, \lambda, \lambda) = \{[q_1, \lambda]\}$$
$$\delta(q_0, b, A) = \{[q_2, \lambda]\}$$
$$\delta(q_1, \lambda, A) = \{[q_1, \lambda]\}$$
$$\delta(q_2, b, A) = \{[q_2, \lambda]\}$$
$$\delta(q_2, \lambda, A) = \{[q_2, \lambda]\}$$

a. Descreva a linguagem aceita por M.

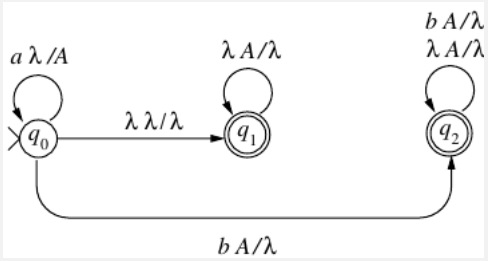
b. Apresente o diagrama de estados de M.

c. Mostre que $aabb, aaab \in \mathsf{L}(\mathsf{M})$.

a. O PDA M aceita a linguagem $\mathsf{L}(\mathsf{M}) = \{a^i b^j \mid 0 \leq j \leq i\}$. O processamento de um a empilha A na pilha. *Strings* da forma a^i são aceitas no estado q_1. As transições em q_1 esvaziam a pilha quando a entrada já tiver sido consumida. Uma computação com entrada $a^i b^j$ entra no estado q_2 após ler o primeiro b. Para que toda a entrada seja lida, a pilha deve conter pelo menos j A's. Caso a entrada termine, a pilha é esvaziada em q_2 para satisfazer a condição de aceite de um PDA: ler toda a entrada e pilha vazia.

   A computação para uma entrada que possui menos a's do que b's, ou que possui um a sucedendo um b, fica parada no estado q_2. Mas, como a entrada não foi totalmente processada, a condição de aceite de PDAs não foi satisfeita e a entrada é rejeitada.

b. O diagrama de estados de M é apresentado abaixo.



c. Para mostrar que as *strings* $aabb$ e $aaab$ pertencem a $\mathsf{L}(\mathsf{M})$, basta mostrar o *trace* da computação de M que aceita essas entradas.

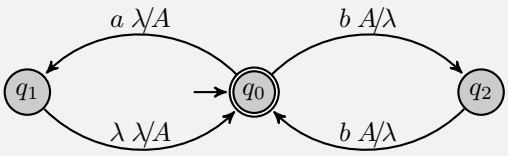
State	String	Stack
q_0	$aabb$	λ
q_0	abb	A
q_0	bb	AA
q_2	b	A
q_2	λ	λ

State	String	Stack
q_0	$aaab$	λ
q_0	aab	A
q_0	ab	AA
q_0	b	AAA
q_2	λ	AA
q_2	λ	A
q_2	λ	λ

4 Construa PDAs que aceitem as linguagens abaixo.

a. $\{a^i b^{2i} \mid i \geq 0\}$.
b. $\{a^i b^j \mid 0 \leq i \leq j \leq 2i\}$.

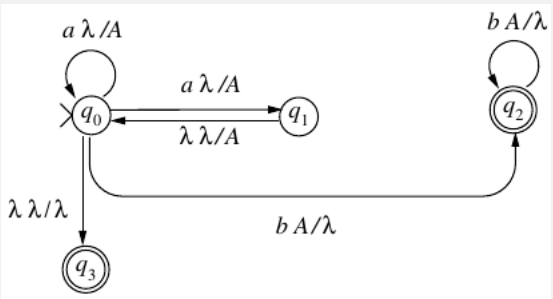
a. O PDA definido pelo diagrama abaixo aceita a linguagem $\{a^i b^{2i} \mid i \geq 0\}$.



b. A linguagem $\mathsf{L} = \{a^i b^j \mid 0 \leq i \leq j \leq 2i\}$ pode ser gerada pela gramática livre de contexto abaixo.

$$\begin{aligned} S &\rightarrow aSB \mid \lambda \\ B &\rightarrow bb \mid b \end{aligned}$$

A regra B gera um ou dois b's para cada a. Um PDA M que aceita L usa os a's para registrar um número aceitável de b's correspondentes na pilha. Ao processar um a, a computação de M empilha de forma não-determinística um ou dois A's na pilha. (Lembre-se que um PDA é uma máquina não-determinística.) O diagrama de M é mostrado abaixo.



5 Seja G uma gramática sensível ao contexto definida pelas regras abaixo.

$$\begin{aligned} S &\rightarrow SBA \mid a \\ BA &\rightarrow AB \\ aA &\rightarrow aaB \\ B &\rightarrow b \end{aligned}$$

a. Apresente uma derivação de $aabb$.
b. Qual é $\mathsf{L}(\mathsf{G})$?

a. A derivação pedida é mostrada abaixo.

$$\begin{aligned} S &\Rightarrow SBA \\ &\Rightarrow aBA \\ &\Rightarrow aAB \\ &\Rightarrow aaBB \\ &\Rightarrow aabB \\ &\Rightarrow aabb \end{aligned}$$

b. $\mathsf{L}(\mathsf{G}) = \{a^{i+1}b^{2i} \mid i \geq 0\}$.

6 Projete um LBA M que aceita a linguagem $\mathsf{L} = \{a^i b^{2i} a^i \mid i > 0\}$.

Devemos lembrar que a única diferença entre um LBA e uma TM é que a fita de um LBA é limitada pelo tamanho da entrada. Assim, a entrada do LBA M é $\langle w \rangle$, onde $w \in \{a, b\}^*$. Os símbolos $\langle$ e $\rangle$ servem para sinalizar os limites da fita.

O funcionamento de M é muito similar às TM já estudadas para reconhecimento de linguagens. A máquina utiliza um símbolo especial X para marcar a entrada já analisada. Assim, começando da posição 0, a máquina busca um a do prefixo (sequência inicial de a's) para marcar. A seguir, M caminha para a direita até encontrar e marcar dois b's. Depois, M continua para a direita até marcar um a do sufixo (sequência final de a's). Se a qualquer momento desta busca para a direita M encontra o fim da fita, indicado por $\rangle$, a máquina para e rejeita a entrada. Caso contrário, a cabeça é rebobinada até $\langle$ e o *loop* se repete para o próximo a do prefixo. Só devemos tomar cuidado para garantir que o prefixo e o sufixo têm o mesmo número de a's. É necessário um caso especial para rejeitar as *strings* que possuem um prefixo menor que o sufixo (veja a resolução do Exercício 1a da Lista 02).

7 Seja T uma árvore binária completa (isto é, todos os nós internos sempre possuem dois filhos). Um caminho por T é uma sequência de movimentos pelos nós que passa: pelo filho da esquerda (L), pelo filho da direita (R) ou pelo pai (U). Portanto, caminhos podem ser descritos como *strings* sobre o alfabeto $\Sigma = \{L, R, U\}$. Considere a linguagem $\mathsf{L} = \{w \in \Sigma^* \mid w$ descreve um caminho que começa na raiz e retorna à raiz$\}$. Por exemplo, $\lambda, LU, LRUULU \in \mathsf{L}$, e $U, LRU \notin \mathsf{L}$. Determine a localização de L na Hierarquia de Chomsky (isto é, determine o tipo de L: 0, 1, 2, ou 3). Considere que a altura h da árvore T pode variar de 0 a um valor fixo $n \in \mathbf{N}$, isto é, $0 \leq h \leq n$.

Vamos começar com o caso mais simples, quando a altura da árvore T é zero. Se $h = 0$, então a linguagem é formada somente pela *string* nula, isto é, $\mathsf{L}_0 = \{\lambda\}$. Podemos ver imediatamente que L_0 é regular.

Se a árvore tem altura 1, a linguagem L_1 correspondente é dada pela expressão regular $(\boldsymbol{LU} \cup \boldsymbol{RU})^*$. Essa expressão é fácil de ser compreendida pois qualquer passo para baixo na árvore (L ou R) deve necessariamente ser seguido por um passo para cima, pois a árvore tem altura 1. Esse processo pode ser repetido quantas vezes quisermos, o que explica o fecho de Kleene $(*)$.

De forma geral, uma *string* é aceita se ela satisfaz as duas condições abaixo.

1. O número de passos para baixo na árvore (L ou R) é igual ao número de passos para cima (U).
2. Os passos não indicam um movimento para baixo de uma folha ou para cima da raiz.

Para uma árvore de altura h, um DFA com $2h + 2$ estados pode ser construído para aceitar a linguagem. O estado q_0 é o estado de aceite e representa uma *string* que satisfaz a condição 1 acima. Os estados q_i, $i = 1, \ldots, h$, são usados para registrar que a *string* lida até o momento tem i U's a mais que L's ou R's. Já os estados $q_j, j = h + 1, \ldots, 2h$, representam o contrário, quando a *string* lida até o momento tem j L's ou R's a mais do que U's. Se a *string* representa um caminho que sairia da árvore (viola a condição 2 acima), o DFA entra em um estado de erro q_e e rejeita a entrada.

Pelo exposto acima, para qualquer valor de h, temos que L_h é regular, pois ela é aceita por um autômato finito. (O limite superior n para h garante que o DFA sempre terá uma quantidade finita de estados.) Finalmente, temos que

$$\mathsf{L} = \bigcup_{h=0}^{n} L_h$$

e como as linguagens regulares são fechadas sob a operação de união, concluímos que L também é regular, e, portanto, do tipo 3 na Hierarquia de Chomsky.

8 Considere a linguagem $\mathsf{L} = \{a^i b^i c^j d^j \mid i, j > 0\}$ definida sobre o alfabeto $\Sigma = \{a, b, c, d\}$. Qual é o tipo de L segundo a Hierarquia de Chomsky (HC)? Justifique sua resposta projetando uma máquina abstrata

M que reconhece L, aonde M deve possuir o mínimo poder de computação necessário para reconhecer a linguagem. (Isto é, M deve estar no mesmo nível/linha que L na HC.)

A linguagem L é do tipo 2, isto é, uma linguagem livre de contexto. É fácil perceber que não é possível construir um autômato finito para reconhecer L pois os valores de i e j são arbitrários e, portanto, o reconhecimento exige algum tipo de memória para contagem. Vamos então construir um PDA que é a máquina associada às linguagens do tipo 2.

O PDA M tem como alfabeto de entrada $\Sigma = \{a, b, c, d\}$ e alfabeto da pilha $\Gamma = \{A, C\}$, aonde A e C são utilizados respectivamente para indicar o número de a's e c's já lidos. O diagrama de M é apresentado abaixo.