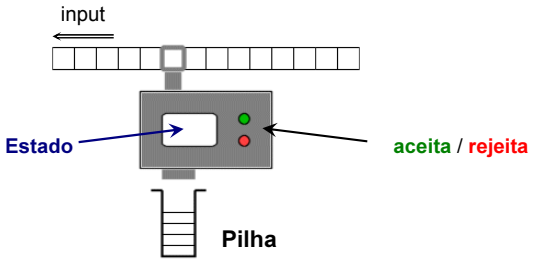
Capítulo 6

Autómatos de Pilha

Um autómato de pilha determinista (APD) é essencialmente um autómato finito determinista ao qual se acrescenta uma pilha que permite guardar informação:

$$\text{APD} = \text{AFD} + \text{Pilha}.$$



O funcionamento de um APD consiste em ler o símbolo na posição actual de leitura e o símbolo situado no topo da pilha e, com base nesses dois símbolos e no estado actual:

1. remove o símbolo do topo da pilha;
2. opcionalmente coloca um ou mais itens na pilha;
3. transita para outro estado;
4. avança a posição de leitura.

Num autómato de pilha não determinista (AP), associado a cada estado e símbolo no topo da pilha podem existir várias opões para transição de estado (em número finito) bem como para o conteúdo que é colocado na pilha:

$$\text{AP} = \text{AFND}\varepsilon + \text{Pilha}.$$

Assim como mostrámos no Capítulo 4, p. 85 que os autómatos finitos são as máquinas que permitem reconhecer as linguagens regulares, vamos ver neste capítulo

que os autómatos de uma pilha são máquinas que permitem reconhecer as linguagens independentes do contexto.

6.1 Linguagens dos autómatos de pilha

Enquanto a classe dos autómatos finitos não deterministas é equivalente à classe dos autómatos finitos deterministas, veremos na Secção 6.3, p. 176 que a classe dos autómatos de pilha deterministas é uma subclasse própria da classe dos autómatos de pilha (não deterministas).

Começamos com uma definição geral de autómato de pilha de natureza não determinista.

Definição 6.1.1 Autómato de pilha.

Um **autómato de uma pilha** (AP) é um séptuplo ordenado

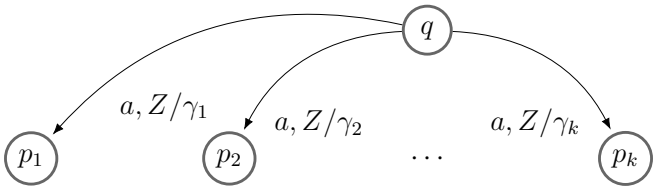
$$A = (Q, \Sigma, \Gamma, \delta, s, Z_0, F)$$

onde:

- Q é um conjunto finito, não vazio, de *estados internos*;
- Σ é um alfabeto finito de *símbolos de entrada*;
- Γ é um alfabeto finito de *símbolos de pilha*;
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \mathcal{P}_{\text{fin}}(Q \times \Gamma^\star)$, é a *função (parcial) de transição* ou de mudança de estado;
- $s \in Q$ é o *estado inicial*;
- $Z_0 \in \Gamma \setminus \Sigma$ é o *símbolo inicial da pilha*;
- $F \subseteq Q$ é o conjunto dos *estados de aceitação*.

6.1.1 Transições

Separamos a interpretação da função de transição num autómato de pilha em dois casos. No primeiro, a transição é por leitura de um símbolo de entrada. No segundo, a transição é feita por ε, não ocorrendo a leitura do símbolo de entrada nem o avanço na posição de leitura.



Para $a \in \Sigma$, a transição

$$\delta(q, a, Z) = \{(p_1, \gamma_1), (p_2, \gamma_2), \ldots, (p_k, \gamma_k)\}$$

significa que a partir do estado q, por leitura do símbolo a, com o símbolo Z no topo da pilha, ocorre uma escolha (não determinista) de um $i \in \{1, \ldots, k\}$ e o autómato

1. transita para o estado p_i;
2. retira o símbolo Z da pilha;
3. coloca a sequência de símbolos γ_i na pilha;
4. avança para o próximo símbolo de entrada.

Para $a = \varepsilon$, a transição

$$\delta(q, \varepsilon, Z) = \{(p_1, \gamma_1), (p_2, \gamma_2), \ldots, (p_k, \gamma_k)\}$$

significa que a partir do estado q, com o símbolo Z no topo da pilha, qualquer que seja o símbolo de entrada (e mesmo que já estejam todos lidos), ocorre uma escolha (não determinista) de um $i \in \{1, \ldots, k\}$ e o autómato

1. transita para o estado p_i;
2. retira o símbolo Z do topo da pilha;
3. coloca a sequência de símbolos γ_i na pilha;
4. ***não avança*** no símbolo de entrada.

Em cada transição, o elemento no topo da pilha é substituído por uma sequência finita de novos símbolos (eventualmente o mesmo símbolo) ou então é simplesmente retirado da pilha (no caso em que $\gamma_i = \varepsilon$), conforme ilustrado na figura seguinte.

6.1.2 Configurações

A configuração de um autómato de pilha é em cada momento caracterizada pelo seu estado actual, pela parte da palavra de entrada que ainda não foi lida e pelo conteúdo da pilha. Usando esta informação, podemos prever todas as possíveis histórias de transições futuras do autómato.

Definição 6.1.2 Configuração de um autómato de pilha.

A *configuração* ou *descrição instantânea* de um autómato de pilha é um triplo

$$(q, w, \gamma)$$

onde:

1. q é o estado actual do autómato;
2. $w \in \Sigma^\star$ é a sequência de símbolos de entrada que falta ler;
3. $\gamma \in \Gamma^\star$ é o conteúdo actual da pilha.

Cada transição de um autómato de pilha modifica a sua descrição instantânea e corresponde a um dos passos, de uma sequência de passos, que começa com uma configuração inicial e termina numa configuração final. As configurações finais são atingidas sempre que o autómato não pode mudar de configuração: quer porque a palavra de entrada foi completamente lida, quer porque a pilha fica vazia.

Definição 6.1.3 Passo e sequência de passos.

Um *passo* de um autómato de pilha é representado por

$$(q, aw, Z\alpha) \vdash (p, w, \beta\alpha),$$

sempre que $(p, \beta) \in \delta(q, a, Z)$.

O fecho simétrico e transitivo de $\vdash$ denota-se por $\vdash^\star$ e representa uma *sequência de zero ou mais passos.*

6.1.3 Linguagem Reconhecida por um Autómato de Pilha

Após processar todos os símbolos de uma palavra, um autómato de pilha atinge uma configuração em que apenas poderá avançar por transições ε (desde que a pilha não esteja vazia). Por outro lado, a partir do momento em que a pilha fica vazia, o autómato não pode mudar de configuração.

Podemos pensar em duas formas de reconhecimento de palavras: uma em que o autómato processa uma palavra e atinge uma configuração de aceitação (reconhecimento) que corresponde a estar num estado de aceitação (o que pressupõe que

a pilha não fica vazia entretanto) e outra em que o autómato processa a palavra e atinge uma configuração de aceitação (reconhecimento) que corresponde ao momento em que a pilha ficar vazia (independentemente do estado ser de aceitação ou de rejeição).

Atendendo à natureza não determinista do funcionamento de um autómato de pilha, uma palavra é reconhecida desde que exista pelo menos uma sequência de passos que conduza a uma configuração de aceitação (entre outras sequências que eventualmente conduzam a configurações de rejeição).

Definição 6.1.4 Reconhecimento por estados de aceitação.

Seja $A = \big(Q, \Sigma, \Gamma, \delta, s, Z_0, F\big)$ um AP. A linguagem de A, *reconhecida por estados de aceitação* é

$$\mathcal{L}(A) = \{w \in \Sigma^\star : \ (s, w, Z_0) \vdash^\star (p, \varepsilon, \alpha) \text{ para algum } p \in F \text{ e algum } \alpha \in \Gamma^\star\}\,.$$

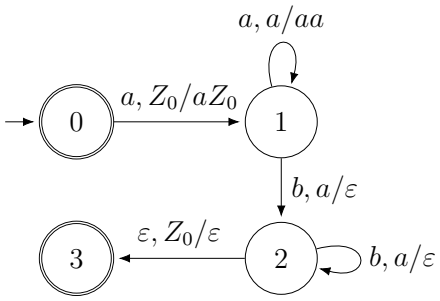
Exemplo 6.1.5

Vamos construir um autómato de pilha que reconheça por estados de aceitação a linguagem $L = \{a^n b^n : \ n \geq 0\}$.

A estratégia consiste em guardar na pilha todos os a´s lidos até aparecer o primeiro b. A partir daí, o autómato retira da pilha um a por cada b que aparece.

Na figura seguinte, ilustramos o diagrama de transições do autómato

$$A = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \{a, b, Z_0\}, \delta, q_0, Z_0, \{q_0, q_3\})\,.$$



Indicamos agora as diferentes interpretações das transições presentes no diagrama anterior.

- $\delta(q_0, a, Z_0) = \{(q_1, aZ_0)\}$: se na configuração actual o estado for q_0, o símbolo actual de entrada for a e se o símbolo no topo da pilha for Z_0 então juntamos a à pilha e o estado passa a ser q_1.
- $\delta(q_1, a, a) = \{(q_1, aa)\}$: se na configuração actual o estado for q_1, o símbolo actual de entrada for a e se o símbolo no topo da pilha for a então

juntamos a à pilha e o estado continua a ser q_1.

- $\delta(q_1, b, a) = \{(q_2, \varepsilon)\}$: se na configuração actual o estado for q_1, o símbolo actual de entrada for b e se o símbolo no topo da pilha for a então retiramos a do topo da pilha e o estado passa a ser q_2.
- $\delta(q_2, b, a) = \{(q_2, \varepsilon)\}$: se na configuração actual o estado for q_2, o símbolo actual de entrada for b e se o símbolo no topo da pilha for a então retiramos a do topo da pilha e o estado continua a ser q_2.
- $\delta(q_2, \varepsilon, Z_0) = \{(q_3, \varepsilon)\}$: se na configuração actual o estado for q_2 e se o símbolo no topo da pilha for Z_0, independentemente do símbolo de entrada, retiramos Z_0 do topo da pilha e o estado passa a ser q_3.

Na forma de reconhecimento por pilha vazia, podemos assumir que $F = \emptyset$. Assim, representamos os autómatos em que o reconhecimento é por pilha vazia apenas por um sêxtuplo $A = \big(Q, \Sigma, \Gamma, \delta, s, Z_0\big)$.

Definição 6.1.6 Reconhecimento por pilha vazia.

Seja $A = \big(Q, \Sigma, \Gamma, \delta, s, Z_0\big)$ um AP. A linguagem de A, *reconhecida por pilha vazia*, é

$$\mathcal{N}(A) = \{w \in \Sigma^\star : \ (s, w, Z_0) \vdash^\star (p, \varepsilon, \varepsilon) \text{ para algum } p \in Q\}.$$

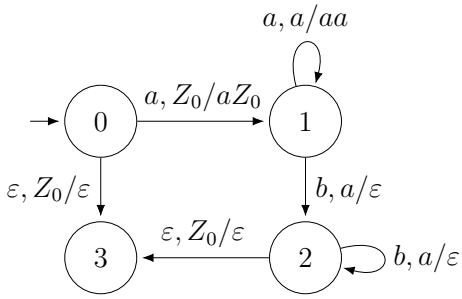
Exemplo 6.1.7

No estado de aceitação q_3 do autómato do Exemplo 6.1.5, p. 167 a pilha fica vazia e todas as palavras da forma $a^n b^n$, com $n \geq 1$, são reconhecidas quando o autómato atinge esse estado. Assim, para que a palavra vazia seja também reconhecida, basta incluir a possibilidade de o autómato esvaziar a sua pilha a partir do estado inicial.

A solução para o problema de reconhecer a linguagem $L = \{a^n b^n : \ n \geq 0\}$ por pilha vazia é dada pelo autómato

$$A = \big(\{q_0, q_1, q_2, q_3\}, \{a, b\}, \{Z_0, a, b\}, \delta, q_0, Z_0\big)$$

definido pelo seguinte diagrama de transições:



Exemplo 6.1.8

Consideremos a LIC $L = \{ww^{-1} : \ w \in \{0,1\}^\star\}$, constituída pelos palíndromos binários de tamanho par.

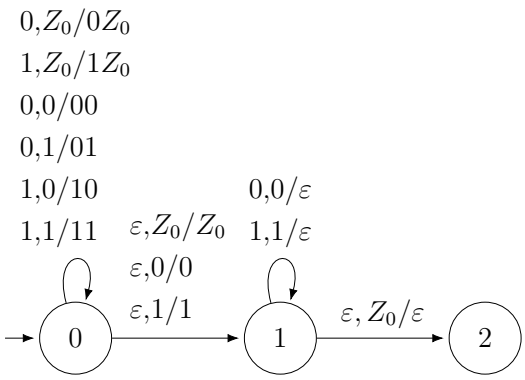
Esta linguagem é gerada pela GIC $G = (\{S\}, \{0,1\}, P, S)$ com as seguintes produções:

$$S \to \varepsilon \mid 0S0 \mid 1S1$$

A estratégia para o reconhecimento por pilha vazia por um AP é a seguinte:

1. Começando no estado inicial q_0, ler cada um dos símbolos de entrada e colocá-los na pilha;
2. A certa altura (transição ε) transitar para o estado q_1;
3. Continuar a ler os símbolos de entrada e, ao mesmo tempo, retirar itens da pilha verificando se são iguais;
4. Se a pilha ficar vazia após ter sido lido o último símbolo então a palavra é reconhecida.

Assim, um AP que reconhece esta linguagem por pilha vazia é dado pelo seguinte diagrama de transições:



Vamos agora mostrar que existe uma equivalência entre as duas noções de reconhecimento.

Teorema 6.1.9

Para todo o autómato de pilha A existe um autómato de pilha B tal que $\mathcal{L}(A) = \mathcal{N}(B)$.

Demonstração.

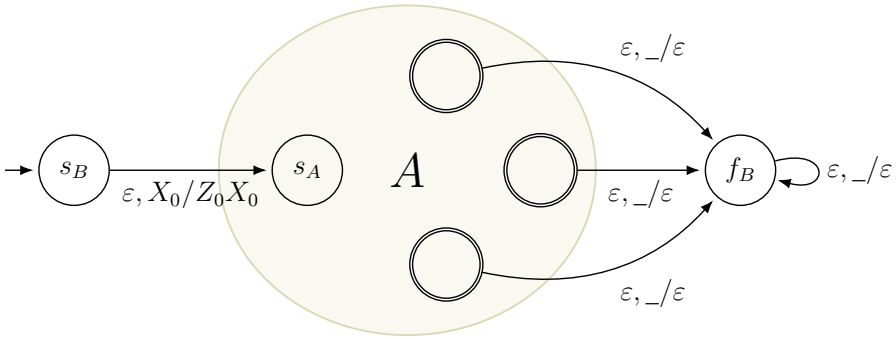
Seja $A = (Q_A, \Sigma, \Gamma, \delta_A, s_A, Z_0, F_A)$, para o qual $L = \mathcal{L}(A)$. Vamos construir um autómato B, que reconhece L por pilha vazia.

O autómato B simula o funcionamento do autómato A de forma a que quando A atinge um estado de aceitação, B esvazia a sua pilha.

A principal dificuldade é a de evitar que B esvazie a pilha antes de atingir um estado de aceitação. Ultrapassamos este problema da seguinte forma:

1. Definimos um novo símbolo inicial X_0 para a pilha de B e um novo estado inicial s_B para o autómato B. Estes servem para colocar o símbolo inicial da pilha de A na pilha de B e para que o autómato B possa transitar para o estado inicial de A.
2. O autómato B simula todas as transições do autómato A, desde o estado inicial até aos estados de aceitação.
3. Definimos um novo estado f_B, onde a pilha de B é esvaziada.
4. Se um estado de aceitação de A for atingido, a pilha de B é esvaziada sem que seja lido mais nenhum símbolo da palavra de entrada.

A figura seguinte ilustra esta construção.



Deixamos como exercício os detalhes da definição do autómato B e a prova de que de facto $L = \mathcal{N}(B)$.

Teorema 6.1.10

Para todo o autómato de pilha A existe um autómato de pilha B tal que $\mathcal{N}(A) = \mathcal{L}(B)$.

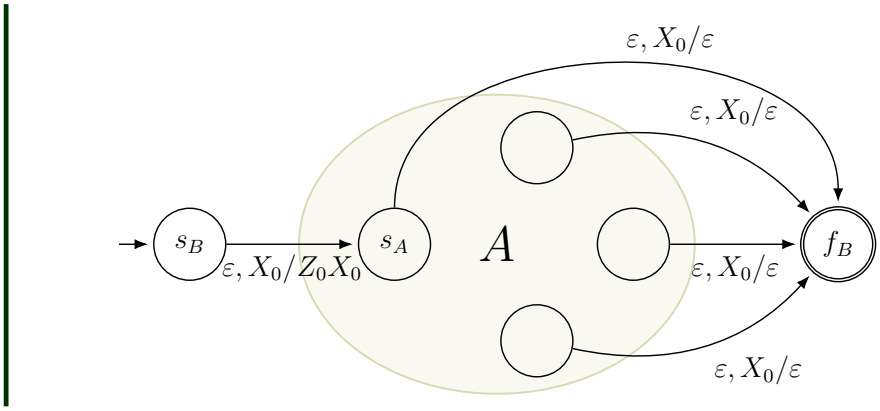
Demonstração.

Seja $A = (Q_A, \Sigma, \Gamma, \delta_A, s_A, Z_0)$, para o qual $L = \mathcal{N}(A)$. Vamos construir um autómato B que reconhece L por estados de aceitação.

O autómato B simula o funcionamento do autómato A e, sempre que A esvazia a pilha, o autómato B passa a um estado de aceitação. Tal é conseguido da seguinte forma:

1. Definimos um novo símbolo inicial X_0 para a pilha do autómato B e um novo estado inicial s_B para o autómato B.
2. O autómato B simula todas as transições de A, desde o estado inicial até aos estados em que A esvazia a pilha.
3. Definimos um novo estado f_B como o único estado de aceitação de B.
4. Sempre que a pilha de A é detectada vazia, ocorre uma transição para o estado de aceitação de B.

Esta construção é ilustrada na figura seguinte. Deixamos os detalhes da definição do autómato B, bem como a prova de que $L = \mathcal{L}(B)$, como exercícios.



6.2 Gramáticas e autómatos de pilha

Já demonstrámos na secção anterior a equivalência entre as noções de reconhecimento por estados de aceitação e por pilha vazia. Vamos agora mostrar que:

1. $L = \mathcal{L}(G) \implies L = \mathcal{N}(A)$: dada uma linguagem gerada por alguma gramática independente do contexto é possível construir um autómato de pilha que a reconhece por pilha vazia.
2. $L = \mathcal{N}(A) \implies L = \mathcal{L}(G)$: dada uma linguagem reconhecida por pilha vazia por um autómato de pilha é possível definir uma gramática independente do contexto que a gera.

6.2.1 Conversão de uma GIC num AP

A estratégia para construir um AP a partir de uma GIC $G = (V, \Sigma, P, S)$ consiste em simular todas as derivações possíveis no reconhecimento das palavras de L:

- se um símbolo terminal $a \in \Sigma$ é lido e está no topo da pilha então é removido da pilha;
- se uma variável $X \in V$ estiver no topo da pilha então pode ser substituída pelo lado direito de uma das produções que tenham X no lado esquerdo.

Seguindo esta estratégia, obtemos o autómato de pilha $A = \big(\{s\}, \Sigma, V \cup \Sigma, \delta, s, S\big)$ com as seguintes características:

- tem apenas um estado s;

- o alfabeto da pilha é o vocabulário completo de G;
- S funciona como o símbolo inicial da pilha;
- a função de transição é definida por:
  1. se $a \in \Sigma$ então $\delta(s, a, a) = \{(s, \varepsilon)\}$;
  2. se $X \in V$ então $\delta(s, \varepsilon, X) = \{(s, \alpha) : \ (X \to \alpha) \in P\}$.

Exemplo 6.2.1

Para a gramática geradora da linguagem $L = \{a^n b^n : \ n \geq 0\}$,

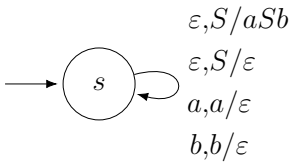
$$G = (\{S\}, \{a, b\}, \{S \to aSb \mid \varepsilon\}, S)$$

o autómato de pilha A tal que $\mathcal{L}(G) = \mathcal{N}(A)$ é

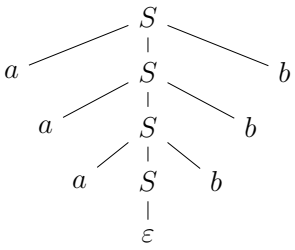
$$A = (\{s\}, \{a, b\}, \{S, a, b\}, \delta, s, S)$$

com δ definida por:

$$\begin{aligned} \delta(s, \varepsilon, S) &= \{(s, aSb), (s, \varepsilon)\} \\ \delta(s, a, a) &= \{(s, \varepsilon)\} \\ \delta(s, b, b) &= \{(s, \varepsilon)\} \end{aligned}$$



Em seguida, ilustramos o reconhecimento da palavra $aaabbb$ pelo autómato de pilha.



$aaabbb$	S
$aaabbb$	aSb
$aabbb$	Sb
$aabbb$	$aSbb$
$abbb$	Sbb
$abbb$	$aSbbb$
bbb	$Sbbb$
bbb	bbb
bb	bb
b	b
ε	

6.2.2 Conversão de um AP numa GIC

A estratégia para obter uma GIC que reconheça a linguagem L de um autómato de pilha $A = (Q, \Sigma, \Gamma, \delta, s, Z_0)$ consiste em simular os passos do autómato no reconhecimento de uma palavra, os quais envolvem as seguintes acções:

1. por leitura de algum símbolo de entrada (ou mesmo sem ler) *retirar* $X \in \Gamma$ da pilha;
2. *colocar* algum $\gamma \in \Gamma$ na pilha;
3. *transitar* de um estado $p \in Q$ para um estado $q \in Q$.

Como representar as variáveis? Consideramos um símbolo inicial $S \notin \Sigma \cup \Gamma$ e cada uma das restantes variáveis é um terno $[p\, X\, q]$ que representa a acção do autómato:

estando no estado p, com o símbolo X no topo da pilha, transitar para o estado q, retirando X da pilha.

Como representar as produções? As produções são definidas de tal modo que cada variável $[p\, X\, q]$ gere as palavras $w \in \Sigma^\star$ que satisfazem

$$(p, w, X) \vdash^\star (q, \varepsilon, \varepsilon).$$

1. Começamos por definir as produções $S \to [s\, Z_0\, p]$ (uma para cada estado $p \in Q$).
2. Em seguida, se $(p_1, Y_1 Y_2 \dots Y_k) \in \delta(p, a, X)$, onde a pode ser ε e k pode ser 0, então para todos os possíveis estados $p_2, p_3, \dots, p_{k+1} \in Q$, juntamos também as produções da forma

$$[p\, X\, p_{k+1}] \to a[p_1\, Y_1\, p_2][p_2\, Y_2\, p_3] \cdots [p_k\, Y_k\, p_{k+1}].$$

Por exemplo,

- se $(q, \varepsilon) \in \delta(p, a, X)$ —, ou seja, se retiramos X da pilha por leitura de a — então juntamos a produção $[p\, X\, q] \to a$.
- se $(q, Y) \in \delta(p, a, X)$ —, ou seja, se se transita do estado p para q e se substitui o elemento X por Y no topo da pilha — então para todos os estados r, juntamos as produções $[p\, X\, r] \to a[q\, Y\, r]$.
- se $(q, YZ) \in \delta(p, a, X)$ —, ou seja, se se transita do estado p para q e se substitui o elemento X no topo da pilha por YZ — então para todos os estados r e s, juntamos a produção $[p\, X\, r] \to a[q\, Y\, s][s\, Z\, r]$.

A gramática assim construída gera a linguagem reconhecida por pilha vazia pelo autómato A.

Formalmente, a gramática é $G = (V, \Sigma, P, S)$, onde

$$V = \{[p\,X\,q] :\ p, q \in Q, X \in \Gamma\} \cup \{S\}$$

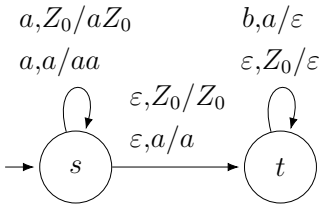
e

$$P = \big\{S \to [s\,Z_0\,p] :\ p \in Q\big\} \cup$$
$$\big\{[p\,X\,p_{k+1}] \to a[p_1\,Y_1\,p_2][p_2\,Y_2\,p_3]\cdots[p_k\,Y_k\,p_{k+1}] :$$
$$a \in \Sigma \cup \{\varepsilon\},\ p_2, p_3, \ldots, p_{k+1} \in Q, (p_1, Y_1 Y_2 \cdots Y_k) \in \delta(p, a, X)\big\}$$

A gramática pode conter (e geralmente é esse o caso) símbolos e produções inúteis.

Exemplo 6.2.2

Consideramos o autómato de pilha $A = (\{s, t\}, \{a, b\}, \{Z_0, a, b\}, \delta, s, Z_0)$ reconhecedor da linguagem $L = \{a^n b^n :\ n \geq 0\}$ e definido pelo seguinte diagrama:



Vamos seguir o método de construção de uma gramática G, geradora de L, a partir do autómato A.

- As variáveis de G são, para além de S, $[p\,X\,q]$, com $p, q \in Q = \{s, t\}$ e $X \in \Gamma = \{Z_0, a, b\}$.
- As produções a partir do axioma da gramática são:

$$S \to [s\,Z_0\,s] \mid [s\,Z_0\,t]$$

- A transição $\delta(s, a, Z_0) = \{(s, aZ_0)\}$ dá origem às produções:

$$[s\,Z_0\,s] \to a[s\,a\,s][s\,Z_0\,s] \mid a[s\,a\,t][t\,Z_0\,s]$$
$$[s\,Z_0\,t] \to a[s\,a\,s][s\,Z_0\,t] \mid a[s\,a\,t][t\,Z_0\,t]$$

- A transição $\delta(s, a, a) = \{(s, aa)\}$ dá origem às produções:

$$[s\,a\,s] \to a[s\,a\,s][s\,a\,s] \mid a[s\,a\,t][t\,a\,s]$$
$$[s\,a\,t] \to a[s\,a\,s][s\,a\,t] \mid a[s\,a\,t][t\,a\,t]$$

- A transição $\delta(s, \varepsilon, Z_0) = \{(t, Z_0)\}$ dá origem às produções:

$$[s\,Z_0\,s] \to [t\,Z_0\,s] \qquad [s\,Z_0\,t] \to [t\,Z_0\,t]$$

- A transição $\delta(s, \varepsilon, a) = \{(t, a)\}$ dá origem às produções:

$$[s\,a\,s] \to [t\,a\,s] \qquad [s\,a\,t] \to [t\,a\,t]$$

- A transição $\delta(t, b, a) = \{(t, \varepsilon)\}$ dá origem à produção $[t\,a\,t] \to b$
- A transição $\delta(t, \varepsilon, Z_0) = \{(t, \varepsilon)\}$ dá origem à produção $[t\,Z_0\,t] \to \varepsilon$

Eliminando os símbolos e produções inúteis, a gramática reduz-se às produções

$$S \to [s\,Z_0\,t]$$
$$[s\,Z_0\,t] \to a[s\,a\,t][t\,Z_0\,t] \mid [t\,Z_0\,t]$$
$$[s\,a\,t] \to a[s\,a\,t][t\,a\,t] \mid [t\,a\,t]$$
$$[t\,Z_0\,t] \to \varepsilon \qquad [t\,a\,t] \to b$$

Após simplificação de variáveis, esta gramática é equivalente à gramática com produções

$$S \to aX \mid \varepsilon \qquad X \to aXb \mid b$$

6.3 Autómatos de pilha deterministas

Um autómato de pilha é determinista sempre que, no decorrer processo de aceitação ou rejeição de uma qualquer palavra, exista uma única possibilidade de avançar de uma configuração para a seguinte.

Assim, para que seja determinista, a definição do autómato não pode permitir que haja escolha entre dois estados, nem entre duas actualizações distintas do topo da pilha, nem entre transitar pela leitura de um símbolo ou pela palavra vazia.

Definição 6.3.1 Autómato de pilha determinista.

Um autómato de pilha, $A = (Q, \Sigma, \Gamma, \delta, s, Z_0, F)$, é *determinista* se para $q \in Q$ e $Z \in \Gamma$:

1. Para qualquer $a \in \Sigma \cup \{\varepsilon\}$, o conjunto $\delta(q, a, Z)$ tem, quando muito, um elemento;
2. Se $\delta(q, \varepsilon, Z) \neq \emptyset$ então $\delta(q, a, Z) = \emptyset$, para qualquer $a \in \Sigma$.

Ao contrário dos autómatos finitos, em que é possível converter qualquer AFND ou AFNDε num AFD, existem linguagens independentes do contexto para as quais não existe um autómato de pilha determinista que as reconheça. Um exemplo é dado pela linguagem $\{ww^{-1} : \ w \in \{a, b\}^\star\}$.

Atendendo a que todas as linguagens regulares são reconhecíveis por autómatos finitos deterministas, é natural que também sejam reconhecíveis por autómatos de pilha deterministas.

Teorema 6.3.2

Se L é uma linguagem regular então existe um autómato de pilha determinista, A, tal que $L = \mathcal{L}(A)$ (i.e., o autómato A reconhece L por estados de aceitação).

Demonstração.

Seja $A = (Q, \Sigma, \delta_A, s, F)$ um AFD que reconhece L. A partir do autómato A construímos o APD $A_P = (Q, \Sigma, \{Z_0\}, \delta_P, s, Z_0, F)$ que "ignora a sua pilha", isto é,

$$\delta_P(q, a, Z_0) = \begin{cases} (\delta_A(q, a), Z_0) & \text{se } a \in \Sigma \\ \emptyset & \text{se } a = \varepsilon \end{cases}$$

Salientamos que a modalidade de reconhecimento presente no teorema anterior é por estados de aceitação.

Embora para os autómatos de pilha não deterministas seja indiferente considerar as formas de reconhecimento por estados de aceitação ou por pilha vazia, existem linguagens regulares para as quais não existe um APD que as reconheça por pilha vazia.

Exemplo 6.3.3

Consideremos a linguagem associada à expressão regular 0^* e vamos admitir que existe um autómato de pilha determinista que reconhece esta linguagem por pilha vazia.

Para a palavra 00 existirá um único caminho desde a configuração inicial até à configuração de aceitação em que a pilha fica vazia. Mas nesse caso o reconhecimento da palavra 000 seguirá o mesmo caminho para os dois primeiros 0´s e nessa altura a pilha fica vazia pelo que o autómato não pode prosseguir para o terceiro 0.

É assim evidente que a característica desta linguagem que a impede de ser reconhecida por um APD na modalidade de pilha vazia é o facto de ela conter palavras que são prefixos de outras palavras da linguagem. Dito de outra forma, a linguagem não é livre de prefixos (confirmar com a Definição 1.2.19, p. 15).

Teorema 6.3.4

Uma linguagem L é reconhecida por pilha vazia por um APD *se e só se L é livre de prefixos e é reconhecida por estados de aceitação por um* APD.

Demonstração.
A prova passa por usar as mesmas construções definidas no caso geral para converter a modalidade de reconhecimento por pilha vazia na modalidade de reconhecimento por estados de aceitação e vice-versa. Deixamos os detalhes ao cuidado do leitor.

Qualquer linguagem L pode ser transformada numa linguagem semelhante e livre de prefixos. Basta considerar um novo símbolo, \$, para marcar o fim das palavras e definir $L_{\$} = \{w\$: \ w \in L\}$.

6.4 Propriedades das LIC

Conforme prometido no capítulo anterior, vamos agora destacar mais algumas propriedades da classe das linguagens independentes do contexto, colocando em evidência as diferenças relativas à classe das linguagens regulares.

Teorema 6.4.1

Se L_1 e L_2 são linguagens independentes do contexto então a união $L_1 \cup L_2$ *é uma linguagem independente do contexto.*

Demonstração.
Sejam $G_1 = (V_1, \Sigma, P_1, S_1)$ e $G_2 = (V_2, \Sigma, P_2, S_2)$ GIC geradoras das linguagens L_1 e L_2, respectivamente.

Vamos admitir que os conjuntos das variáveis das duas gramáticas são disjuntos, ou seja, $V_1 \cap V_2 = \emptyset$, renomeando as variáveis de uma das gramáticas quando tal não se verifique.

Deixamos como exercício provar que a gramática

$$G_3 = (V_1 \cup V_2, \Sigma, P_1 \cup P_2 \cup \{S \to S_1 \mid S_2\}, S)$$

gera a linguagem $L_1 \cup L_2$.

Teorema 6.4.2

Se L_1 e L_2 são linguagens independentes do contexto então a concatenação $L_1 L_2$ *é uma linguagem independente do contexto.*

Demonstração.
De forma análoga à demonstração do teorema anterior, basta juntar a produção $S \to S_1 S_2$ e assegurar que $V_1 \cap V_2 = \emptyset$ (se necessário, mudando o nome de algumas variáveis).

Teorema 6.4.3

Se L é uma linguagem independentes do contexto então o fecho de Kleene $L^\star$ *é uma linguagem independente do contexto.*

Demonstração.

Seja $G = (V, \Sigma, P, S)$ uma GIC geradora de L. Seja $S_0 \notin V$ um novo símbolo. A gramática

$$G^\star = (V \cup \{S_0\}, \Sigma, P \cup \{S_0 \to S_0 S \mid \varepsilon\}, S_0)$$

gera a linguagem $L^\star$.

Sejam $L_1 = \{a^n b^n c^m : \ m, n \geq 0\}$ e $L_2 = \{a^m b^n c^n : \ m, n \geq 0\}$. Tanto L_1 como L_2 são independentes do contexto e a sua intersecção é a linguagem $L_1 \cap L_2 = \{a^n b^n c^n : \ n \geq 0\}$.

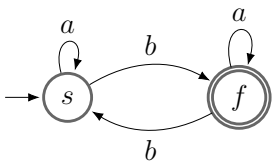
Conforme mostrámos no Exemplo 5.5.3, p. 156, a linguagem $L_1 \cap L_2$ não é independente do contexto. Concluímos assim que a *intersecção* de linguagens independentes do contexto não é necessariamente independente do contexto.

Embora a classe das LIC não seja fechada para a intersecção de linguagens, a intersecção de uma linguagem independente do contexto com uma linguagem regular é uma linguagem independente do contexto.

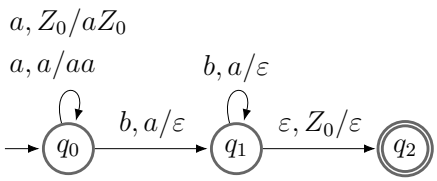
A demonstração usual deste facto utiliza a noção de autómato de pilha e passa por construir o autómato produto de um autómato de pilha com um autómato finito determinista, conforme ilustrado no exemplo seguinte.

Exemplo 6.4.4

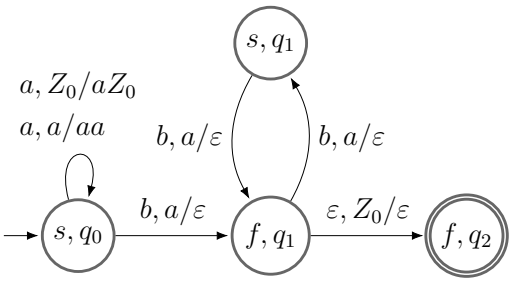
Denotamos por L_R a linguagem regular reconhecida pelo seguinte AFD:



Por outro lado, a linguagem $L_I = \{a^n b^n : \ n > 0\}$ é reconhecida por estados de aceitação pelo seguinte autómato de pilha:



Realizando o produto dos autómatos obtemos o AP reconhecedor de $L_R \cap L_I$:



Em alternativa, vamos demonstrar esta propriedade da intersecção de uma LIC com uma LR usando uma GIC na forma normal de Chomsky e um AFD, baseando-nos em [1].

Começamos por observar que qualquer linguagem regular L_R se pode escrever como uma união de um número finito de linguagens regulares, cada uma delas reconhecível por um AFD com apenas um estado de aceitação, ou seja,

$$L_R = L_1 \cup L_2 \cup \cdots \cup L_n$$

onde L_1, L_2, ..., L_n é reconhecível por um AFD com apenas um estado de aceitação.

Se L_I é uma linguagem independente do contexto então

$$L_R \cap L_I = \bigcup_{k=1}^{n} (L_k \cap L_I)$$

e como a classe das LIC é fechada para a união finita de LIC, basta provar que a intersecção de uma linguagem regular reconhecível por um AFD com apenas um estado de aceitação com uma linguagem independente do contexto é ainda independente do contexto.

Lema 6.4.5

Seja $A = \big(Q, \Sigma, \delta, s, \{f\}\big)$ *um* AFD *com um único estado de aceitação e seja* $G = \big(V, \Sigma, P, S\big)$ *uma* GIC *na forma normal de Chomsky.*

Uma gramática geradora da linguagem $\mathcal{L}(A) \cap \mathcal{L}(G)$ *é* $G_\cap = \big(V_\cap, \Sigma, P_\cap, S_\cap\big)$, *definida por:*

1. $V_\cap = \{[p, X, q] : \ p, q \in Q, \ X \in V\}$.
2. *Para cada produção em* P *da forma* $X \to YZ$, *com* $X, Y, Z \in V$, *e para quaisquer* $p, q, r \in Q$, $[p, X, q] \to [p, Y, r]\,[r, Z, q]$ *é uma produção em* $P_\cap$.
3. *Para cada produção em* P *da forma* $X \to a \in P$, *com* $X \in V$, $a \in \Sigma$, *e para quaisquer* $p, q \in Q$ *tais que* $\delta(p, a) = q$, $[p, X, q] \to a$ *é uma produção em* $P_\cap$.
4. $S_\cap = [s, S, f]$.

Demonstração.

Deixamos como exercício a prova deste resultado.

Teorema 6.4.6

Se L_R *é uma linguagem regular e* L_I *é independente do contexto então* $L_R \cap L_I$ *é independente do contexto.*

Demonstração.

Como vimos anteriormente, é suficiente considerar que a linguagem regular L_R é gerada por um AFD A com um único estado de aceitação.

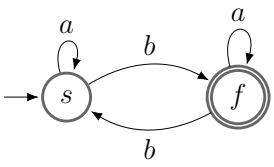
Vimos também na secção Secção 5.2, p. 130 que se L_I é uma LIC tal que que $\varepsilon \notin L_I$ então é gerada por uma gramática na forma normal de Chomsky.

Assim, a gramática G_A definida no lema anterior é uma GIC na forma normal de Chomsky geradora de $\mathcal{L}(A) \cap \mathcal{L}(G)$.

No caso em que $\varepsilon \in L_R \cap L_I$, a gramática que se obtém gera $\mathcal{L}(A) \cap \mathcal{L}(G) \setminus \{\varepsilon\}$. Basta introduzir uma nova variável inicial S_0 e adicionar as duas produções $S_0 \to \varepsilon$ e $S_0 \to [s, S, f]$ para obter uma gramática equivalente à original.

Exemplo 6.4.7

Seja L_R a linguagem regular reconhecida pelo seguinte AFD:



A GIC $G = (\{S, X, A, B\}, \{a, b\}, P, S)$ com produções

$$S \to XB \mid AB \qquad X \to AS \qquad A \to a \qquad B \to b$$

está na forma normal de Chomsky e gera a LIC $L_I = \{a^n b^n : \ n > 0\}$ e a LR.

Assim, uma gramática geradora da linguagem $L_R \cap L_I$ é

$$G_\cap = (V_\cap, \{a, b\}, P_\cap, S_\cap)$$

onde:

1. As $2^2 \times 4 = 16$ variáveis em $V_\cap$ são: $[s, S, s]$, $[s, S, f]$, $[f, S, s]$, $[f, S, f]$, $[s, X, s]$, $[s, X, f]$, $[f, X, s]$, $[f, X, f]$, $[s, A, s]$, $[s, A, f]$, $[f, A, s]$, $[f, A, f]$, $[s, B, s]$, $[s, B, f]$, $[f, B, s]$, $[f, B, f]$.

2. As produções associadas a $S \to XB$ são:

$$\begin{aligned}
[s, S, s] &\to [s, X, s][s, B, s] \mid [s, X, f][f, B, s] \\
[s, S, f] &\to [s, X, s][s, B, f] \mid [s, X, f][f, B, f] \\
[f, S, s] &\to [f, X, s][s, B, s] \mid [f, X, f][f, B, s] \\
[f, S, f] &\to [f, X, s][s, B, f] \mid [f, X, f][f, B, f]
\end{aligned}$$

   As produções associadas a $S \to AB$ são:

$$\begin{aligned}
[s, S, s] &\to [s, A, s][s, B, s] \mid [s, A, f][f, B, s] \\
[s, S, f] &\to [s, A, s][s, B, f] \mid [s, A, f][f, B, f] \\
[f, S, s] &\to [f, A, s][s, B, s] \mid [f, A, f][f, B, s] \\
[f, S, f] &\to [f, A, s][s, B, f] \mid [f, A, f][f, B, f]
\end{aligned}$$

   As produções associadas a $X \to AS$ são:

$$\begin{aligned}
[s, X, s] &\to [s, A, s][s, S, s] \mid [s, A, f][f, S, s] \\
[s, X, f] &\to [s, A, s][s, S, f] \mid [s, A, f][f, S, f] \\
[f, X, s] &\to [f, A, s][s, S, s] \mid [f, A, f][f, S, s] \\
[f, X, f] &\to [f, A, s][s, S, f] \mid [f, A, f][f, S, f]
\end{aligned}$$

3. As produções associadas a $A \to a$ são:

$$[s, A, s] \to a \qquad [s, A, f] \to a \qquad [f, A, s] \to a \qquad [f, A, f] \to a$$

As produções associadas a $B \to b$ são:

$$[s, B, s] \to b \qquad [s, B, f] \to b \qquad [f, B, s] \to b \qquad [f, B, f] \to b$$

4. O axioma da gramática é $S_\cap = [s, S, f]$.

Atendendo a que a classe das linguagens independentes do contexto não é fechada para a intersecção, é possível encontrar exemplos que provam que o mesmo ocorre para o complementar de uma LIC e para diferença de duas LIC.

Consideremos as LIC $L_1 = \{a^n b^n c^m : \ m, n \geq 0\}$ e $L_2 = \{a^m b^n c^n : \ m, n \geq 0\}$. Deixamos como exercício a construção de GIC para as linguagens $\overline{L_1}$ e $\overline{L_2}$, gramáticas essas que provam que os complementares de L_1 e L_2 são ainda LIC.

Uma vez que a classe das LIC é fechada para a operação de união, a linguagem $L = \overline{L_1} \cup \overline{L_2}$ é ainda independente do contexto. O seu complementar,

$$\overline{L} = \overline{\overline{L_1} \cup \overline{L_2}} = L_1 \cap L_2 = \{a^n b^n c^n : \ n \geq 0\},$$

não é no entanto uma linguagem independente do contexto.

Este exemplo permite-nos concluir que:

> O *complementar* de uma linguagem independente do contexto não é necessariamente independente do contexto.

Sempre que uma classe de linguagens é fechada para a operação de diferença então também é fechada para a operação de intersecção. De facto, se L_1 e L_2 são duas quaisquer linguagens, então $L_1 \cap L_2 = L_1 \setminus (L_1 - L_2)$. Assim, se a classe das LIC fosse fechada para a diferença então também seria para a intersecção. Mas não é!

Este argumento permite-nos afirmar que:

> A *diferença* de duas linguagens independentes do contexto não é necessariamente independente do contexto.

Consideremos a gramática G geradora da linguagem $L = \{a^n b^n : \ n \geq 1\}$, com produções $S \to ab \mid aSb$. Se invertermos os lados direitos das produções desta gramática obtemos a gramática G^{-1} cujas produções são: $S \to ba \mid bSa$. Claramente esta gramática gera a linguagem reversa de L, $L^{-1} = \{b^n a^n : \ n \geq 1\}$.

Este exemplo ilustra o resultado geral, enunciado no teorema seguinte.

Teorema 6.4.8

Se L é uma linguagem independentes do contexto então a reversa L^{-1} *é uma linguagem independente do contexto.*

Demonstração.
Basta reverter os lados direitos das produções de uma qualquer GIC que gere L. Os detalhes da demonstração são deixados como exercício.

A classe das linguagens independentes do contexto, no que respeita a homomorfismos e substituições, tem um comportamento idêntico ao da classe das linguagens regulares:

- A *imagem directa* de uma LIC por intermédio de um homomorfismo é uma LIC.
- A *imagem inversa* de uma LIC por intermédio de um homomorfismo é uma LIC.

Teorema 6.4.9

Sejam $h\colon {\Sigma_1}^\star \to {\Sigma_2}^\star$ um homomorfismo de palavras e $L \subseteq {\Sigma_1}^\star$ uma linguagem independente do contexto. Nestas condições, a linguagem $h(L) \subseteq {\Sigma_2}^\star$ é independente do contexto.

Demonstração.
Seja $G = (V, \Sigma_1, P, S)$ uma gramática geradora de uma linguagem L. Deixamos como exercício verificar que a linguagem $h(L)$ é gerada pela gramática $G' = (V', \Sigma_2, P', S')$ especificada por:

- As variáveis de G' incluem todas as variáveis de G e ainda um conjunto de novas variáveis associadas aos símbolos terminais de G: $V' = V \cup \{X_a : a \in \Sigma_1\}$;
- O símbolo inicial de G' é o mesmo de G, ou seja, $S' = S$;
- As produções em G' incluem todas as produções em G, substituindo cada símbolo terminal $a \in \Sigma_1$ no lado direito das produções pela variável X_a. Para além disso, para cada variável X_a definimos uma nova produção $X_a \to h(a)$.

Exemplo 6.4.10

Consideremos a gramática G com produções $S \to 0S0 \mid 1S1 \mid \varepsilon$, geradora da linguagem dos palíndromos binários, e o homomorfismo $h\colon \{0,1\}^\star \to \{a,b\}^\star$ definido por $h(0) = aba$ e $h(1) = b$

A gramática $G' = (\{S, X_0, X_1\}, \{a, b\}, P', S)$ com produções

$$\begin{aligned} S &\to X_0 S X_0 \mid X_1 S X_1 \mid \varepsilon \\ X_0 &\to aba \\ X_1 &\to b \end{aligned}$$

é geradora de $h(\mathcal{L}(G))$.

Teorema 6.4.11

Sejam $h\colon {\Sigma_1}^\star \to {\Sigma_2}^\star$ um homomorfismo de palavras e $L \subseteq {\Sigma_2}^\star$ uma linguagem independente do contexto. Nestas condições, a linguagem $h^{-1}(L) \subseteq {\Sigma_1}^\star$ é independente do contexto.

Demonstração.

Seja $A = (Q, \Sigma_2, \Gamma, \delta, s, F)$ um autómato de pilha reconhecedor de L. Vamos construir um autómato de pilha que dada uma qualquer palavra $w \in {\Sigma_1}^\star$ simula o funcionamento do autómato A com a palavra $h(w)$.

A acção do homomorfismo h numa palavra $w = a_1 a_2 \cdots a_n$ é dada pela concatenação das acções em cada um dos símbolos que a constituem, ou seja,

$$h(a_1 a_2 \cdots a_n) = h(a_1)h(a_2)\cdots h(a_n).$$

Assim, cada um dos estados do autómato simulador vai funcionar como uma memória auxiliar que guarda a parte de uma palavra $h(a_i)$ que falta processar, ou seja, um sufixo de $h(a_i)$.

Seja $H = \{v \in {\Sigma_2}^\star : \ uv \in h(a), a \in \Sigma_1\}$ o conjunto finito de possíveis sufixos de palavras da forma $h(a)$, com $a \in \Sigma_1$.

Consideremos o autómato de pilha $A' = (Q', \Sigma_1, \Gamma, \delta', s', F')$ definido por:

- $Q' = \{[q, v] : \ q \in Q, v \in H\}$
- $s' = (s, \varepsilon)$
- $F' = F \times \{\varepsilon\}$
- para $[q, w] \in Q'$, a função de transição δ' é definida por:
  1. para $a \in \Sigma_1$ e $Z \in \Gamma$,
  $$\delta'([q, \varepsilon], a, Z) = \{([q, h(a)], Z)\}$$
  2. para $b \in \Sigma_2 \cup \{\varepsilon\}$ e $Z \in \Gamma$,
  $$\delta'([q, bw], \varepsilon, Z) = \{([p, w], \alpha) : \ (p, \alpha) \in \delta(q, b, Z)\}$$

3. nos restantes casos, $\delta'([q,w],a,Z) = \emptyset$.

Podemos provar, por indução sobre o número de passos, que

$$([s,\varepsilon],w,Z_0) \vdash_{A'}^{\star} ([q,x],\varepsilon,\alpha) \text{ se e só se } (s,w,Z_0) \vdash_{A}^{\star} (q,\varepsilon,\alpha),$$

onde $h(w) = yx$.

Daqui concluímos que a linguagem reconhecida pelo autómato A', na modalidade de estados de aceitação, satisfaz $h(\mathcal{L}(A')) = \mathcal{L}(A) = L$ e, portanto, $\mathcal{L}(A') = h^{-1}(L)$.

A classe das linguagens independentes do contexto é ainda fechada para a operação de *substituição* (por LIC). Deixamos a demonstração ao cuidado do leitor.

Teorema 6.4.12

Seja L uma LIC *sobre um alfabeto Σ e consideremos uma substituição s que a cada símbolo $a \in \Sigma$ faz corresponder uma* LIC $s(a) = L_a$. *Nestas condições, a linguagem*

$$s(L) = \{w_1 w_2 \cdots w_n : \ w_i \in L_{a_i} \text{ e } a_1 a_2 \cdots a_n \in L, n \in \mathbb{N}_0\}$$

é também independente do contexto.

6.5 Exercícios

1. Verifique que as seguintes linguagens são Independentes do Contexto, construindo autómatos de pilha que as reconheçam, se possível deterministas:

(a) $\{0^n 1^n : \ n > 0\}$;

(b) $\{0^n 1^{2n} : \ n \geq 0\}$;

(c) $\{0^m 1^n : \ m \geq n > 0\}$;

(d) $\{wcw^{-1} : \ w \in \{a,b\}^\star\}$;

(e) $\{w \in \{a,b\}^\star : \ \#_a(w) = \#_b(w)\}$

(f) $\{w \in \{a,b\}^\star : \ \#_a(w) \neq \#_b(w)\}$

(g) $\{a^m b^n c^{m+n} : \ m,n \geq 0\}$

(h) $\{1^m 0^n 1^m : \ m,n \geq 1\}$

(i) $\{01^n 01^n 0 : \ n \geq 1\}$

(j) $\{a^n w w^{-1} b^n : \ w \in \{a,b\}^\star, n \geq 0\}$

2. Construa um AP, se possível determinista, que reconheça a linguagem das sequências de parêntesis rectos correctamente equilibradas. Por exemplo:

[[][[]]][].

3. Sejam $L_1 = \{(11)^n(00)^n : \ n \geq 0\}$, $L_2 = L_1^\star$ e $L_3 = \{\big((11)^n(00)^n\big)^m : m, n \geq 0\}$.

(a) Mostre, usando o teorema da substituição, que L_1 e L_2 são independentes do contexto.

(b) Construa GIC geradoras de L_1 e de L_2 e reduza-as à forma normal de Chomsky.

(c) Construa autómatos de pilha reconhecedores de L_1 e de L_2.

Sugestão. Basta usar o método de conversão de uma GIC num autómato de pilha (não determinista) com um único estado, indicado na secção Subsecção 6.2.1, p. 172.

(d) Mostre que L_3 não é uma LIC.

Sugestão. Seja n o inteiro indicado no lema da bombagem para LIC. Considere a palavra $z = (00)^n(11)^n \in L_3$ de tamanho $4n \geq n$. Dada uma qualquer partição de z da forma $z = uvwxy$, com $|vwx| \leq n$ e $|vx| > 0$, mostre que existe um valor de k tal que $uv^kwx^ky \notin L_3$.

4. Considere a linguagem $L = \{w \in \{a, b\}^+ : \ \#_a(w) = \#_b(w)\}$.

(a) Comente a afirmação "Uma vez que a palavra $ab \in L$ é um prefixo próprio da palavra $abba \in L$, não existe um autómato de pilha que reconheça L por pilha vazia."

(b) Construa um autómato de pilha que reconheça L por estados de aceitação.

5. Considere a linguagem L associada à expressão regular $a^+b^+c^+$.

(a) Construa um autómato finito, A^{-1}, com 4 estados, que reconheça L^{-1}.

(b) A partir do autómato A^{-1} construído na alínea anterior obtenha uma gramática Linear à Direita geradora de L^{-1}.

(c) A partir da gramática obtida na alínea anterior obtenha uma gramática Linear à Esquerda geradora de L.

(d) Construa um AP reconhecedor de $L_I = \{a^nb^ma^mc^n : \ n > 0 \text{ e } m \text{ par}\}$.

(e) Construa um AP reconhecedor de $L \cap L_I$.

Sugestão. Usar o método do autómato produto.

6. Sabe-se que a classe das LIC não é fechada para a intersecção, isto é, existem LIC L_1 e L_2 tais que $L_1 \cap L_2$ não é uma LIC. Usando este facto, mostre que a classe das LIC também não é fechada para o complementar.

7. Mostre que se $L = \mathcal{N}(A)$ para algum APD A então L é livre de prefixos.

Sugestão. Faça a prova por contradição, assumindo que L não é livre de prefixos.

8. Considere a linguagem $L = \{0^n 1^n : \ n \geq 0\}$.

(a) Indique uma GIC G geradora de L e, em seguida, prove que $\mathcal{L}(G) = L$.

(b) Obtenha um AP com um único estado reconhecedor de L. Esse autómato é determinista? Justifique.

(c) Construa um APD que reconheça L por estados de aceitação.

Sugestão. Ver o Exemplo 6.1.5, p. 167.

(d) Caso seja possível construa um APD que reconheça L por pilha vazia, senão indique a razão pela qual tal não é possível.

9. Sabendo que a linguagem $L_1 = \{a^n b^n c^n : \ n \in \mathbb{N}\}$ não é independente do contexto mostre que a linguagem $L_2 = \{w \in \{a,b,c\}^+ : \ \#_a(w) = \#_b(w) = \#_c(w)\}$ também não é independente do contexto.

10. Sabendo que a linguagem $L_1 = \{a^n b^m a^n b^m : \ n, m \geq 0\}$ não é independente do contexto, mostre que $L_2 = \{ww : \ w \in \{a,b\}^\star\}$ também não é uma LIC.

11. Sabendo que a linguagem $L_1 = \{ww^{-1} : \ w \in \{a,b\}^\star\}$ é independente do contexto, mostre que a linguagem $L = \{xx^{-1}y^{-1}y : \ x, y \in \{a,b\}^\star\}$ é também independente do contexto.

12. Seja $A = \big(Q_A, \Sigma, \Gamma, \delta_A, s_A, Z_0, F_A\big)$ um AP reconhecedor de uma LIC L_I, por estados de aceitação. Seja $B = \big(Q_B, \Sigma, \delta_B, s_B, F_B\big)$ um AFD reconhecedor de uma LR L_R.

Mostre que a linguagem $L_I \cap L_R$ é reconhecida pelo autómato de pilha $A_\cap = (Q_A \times Q_B, \Sigma, \Gamma, \delta_\cap, s_\cap, F_\cap)$, produto dos autómatos A e B, onde:

- $s_\cap = (s_A, s_B)$;
- $F_\cap = F_A \times F_B$;
- Se $(p_A, \gamma) \in \delta_A(q_A, a, Z)$ e $p_B = \delta_B(q_B, a)$, para $p_A, q_A \in Q_A, p_B, q_B \in Q_B$, $a \in \Sigma \cup \{\varepsilon\}$, $Z \in \Gamma$ e $\gamma \in \Gamma^\star$ então

$$((p_A, p_B), \gamma) \in \delta_\cap((q_A, q_B), a, Z).$$

Sugestão. Use indução estrutural sobre $w \in \Sigma^\star$.