Exercícios de Teoria da Computação

Departamento de Informática
Universidade Nova de Lisboa

Ano Lectivo de 2008/09

Esta compilação reúne problemas de diversas fontes, algumas das quais indicadas junto aos números dos exercícios. Os problemas de [LP81] e [Rév85] serão referidos, respectivamente, por **LP** e **Rév**. Por exemplo, **LP 1.8.5** indica que se trata do exercício número 1.8.5 de [LP81]. Os marcados com **TE** são problemas de testes e exames desta disciplina. Os restantes, sem indicação explícita da fonte, são da autoria dos docentes que nos últimos anos têm leccionado as aulas práticas — Luís Caires, Margarida Mamede e Michel Wermelinger. Quase sempre os enunciados foram refraseados, tendo-se mantido, no entanto, na essência, o problema original. Nos casos em que o próprio exercício foi ligeiramente alterado, usa-se a designação **adaptação de**. Por vezes, foram agrupados exercícios de diversas origens num só.

A notação adoptada é a das aulas teóricas.

Referências

[LP81] Harry R. Lewis and Jeffrey D. Ullman. *Principles of Compiler Design*. Prentice-Hall Software Series. Prentice-Hall International, Inc., 1981.

[Mam89] Margarida Mamede. *Expressões Regulares Deterministas com Um Símbolo de Avanço*. Relatório de uma Aula Prática, UNL DI 38/89, Universidade Nova de Lisboa, Outubro de 1989.

[Rév85] György E. Révész. *Introduction to Formal Languages*. McGraw-Hill Computer Science Series. McGraw-Hill, Inc., 1985.

Exercícios previstos por aula prática em 2008/09

Aula	Tópicos	Exercícios
1	Linguagens regulares	1, 2, 3, 4, 8
2	Sistemas de equações Linguagens racionais	11 14
3	Autómatos finitos (I)	16, 18, 19, 20
4	Autómatos finitos (II)	22a, 24, 32, 33, 34
5	Gram. indep. contexto	36, 37, 38, 39
6	Autómatos de pilha	48, 50a, 53, 54
7	Gramáticas LL(1)	56, 57
8	Gramáticas LR(0)	59, 60b, 61
9	Gramáticas LR(1)	62
10	Máquinas de Turing	64b, 65c, 66
11	Máquinas de estados e programas	69, 70
12	Verificação pelo método de Floyd (I)	71, 72, 73
13	Verificação pelo método de Floyd (II)	74, 75

1 Linguagens regulares

Exercício 1 (LP 1.8.5.) Dê exemplos de palavras que pertencem e de palavras que não pertencem às seguintes linguagens sobre $T = \{a, b\}$.

a) $\{w : \exists u \in TT \text{ tq } w = uu^{-1}u\}$

b) $\{w : ww = www\}$

c) $\{w : \exists u, v \in T^* \text{ tq } uvw = wvu\}$

d) $\{w : \exists u \in T^* \text{ tq } www = uu\}$

Exercício 2 (inclui TE 86/87) Descreva as seguintes linguagens sobre $T = \{a, b, c\}$.

a) $(a + b + c)^*(a + b + c)$

b) $c^*(a + b)(a + b + c)^*$

c) $((b + c) + a(b + c)^*a)^*$

d) $(ab + ba)^*(aa + bb)$
Assumindo que esta linguagem é a de todas as sequências possíveis de partidas de um dado jogo, em que a representa uma vitória parcial do primeiro jogador e b uma vitória do segundo jogador numa partida, determine a regra do jogo.

Exercício 3 (inclui LP 1.9.3.) Escreva uma expressão regular que represente a linguagem de todas as palavras sobre $T = \{a, b\}$ com:

a) no máximo três ocorrências de a;

b) um número de ocorrências de a divisível por três;

c) exactamente uma ocorrência da subpalavra aaa;

d) exactamente três símbolos;

e) no máximo dois símbolos.

Exercício 4 (LP 1.9.2.) Simplifique as seguintes expressões regulares.

a) $\emptyset^* + a^* + b^* + (a+b)^*$

b) $((a^*b^*)^*(b^*a^*)^*)^*$

c) $(a^*b)^* + (b^*a)^*$

d) $(a+b)^*a(a+b)^*$

Exercício 5 Diga, justificando, se as seguintes expressões regulares são iguais.

a) $(a+b)^*a^*$

b) $(a+b)^*$

c) $((a+b)a)^*$

Exercício 6 (LP 1.8.6.) Em que circunstâncias se tem $L^+ = L^* - \{\lambda\}$?

Exercício 7 (LP 1.8.4.) Demonstre as seguintes proposições.

a) $\{\lambda\}^* = \{\lambda\}$.

b) Para qualquer T e qualquer $L \subseteq T^*$ tem-se $(L^*)^* = L^*$.

c) Se a e b forem símbolos distintos, então $\{a,b\}^* = \{a\}^*\{\{b\}\{a\}^*\}^*$.

d) Se T for um alfabeto qualquer, $\lambda \in L_1 \subseteq T^*$, e $\lambda \in L_2 \subseteq T^*$, então $(L_1T^*L_2)^* = T^*$.

e) Para qualquer linguagem L, $\emptyset L = L\emptyset = \emptyset$.

Exercício 8 (inclui LP 1.9.5.) Verifique se as seguintes afirmações são verdadeiras.

a) $baa \in \mathcal{L}(a^*b^*a^*b^*)$

b) $\mathcal{L}(b^*a^*) \cap \mathcal{L}(a^*b^*) = \mathcal{L}(a^* + b^*)$

c) $\mathcal{L}(a^*b^*) \cap \mathcal{L}(c^*d^*) = \emptyset$

d) $abcd \in \mathcal{L}((a(cd)^*b)^*)$

e) $\lambda \in \mathcal{L}(((\emptyset + a)^* + \emptyset ab^*)(a+b)^*)$

f) $L_1(L_2 \cap L_3) = L_1L_2 \cap L_1L_3$

Exercício 9 Intuitivamente, uma expressão regular diz-se ambígua se existir pelo menos uma palavra da linguagem que pode ser obtida de (pelo menos) duas maneiras distintas. Por exemplo, b^*bb^* é uma expressão regular ambígua. Indique, justificando, se as seguintes expressões regulares são ambíguas.

a) $a((ab^*)cb)^* + a(ababcb^*)^*a^*$

b) $aaab^*(ab)^* + ab^* + a^*bba^*$

c) $aaba^* + aaaba + aabba^* + a$

Exercício 10 ([Mam89]) Seja T um alfabeto e L uma linguagem sobre T, denotada pela expressão regular R, que assumiremos sempre que verifica a propriedade: "em toda a subexpressão de R da forma PQ, P não é um produto de expressões regulares". Designemos por $Prim(R)$ o conjunto de símbolos que ocorrem como primeiro símbolo de uma palavra de L, i.e., $Prim(R) = \{a : a \in T \wedge \exists x \in L \;\; \exists y \in T^* \;\; x = ay\}$.

Diremos que a expressão regular R é $D(1)$, isto é, determinista com um símbolo de avanço se, para toda a subexpressão de R da forma $P+Q$, ou da forma PQ com $\lambda \in \mathcal{L}(P)$, se tiver $Prim(P) \cap Prim(Q) = \emptyset$.

a) Defina, por indução na estrutura de R, a função $\Lambda(R)$, que vale λ ou $\emptyset$, consoante λ pertence, ou não, à linguagem denotada por R.

b) Defina, por indução na estrutura de R, o conjunto $Prim(R)$.

c) Justifique se as seguintes expressões regulares são ou não $D(1)$:

   i) $(ab)^*ab$

   ii) $ab(ab)^*$

   iii) $a^*b + ca^* + b^*c$

   iv) $((a + b^*c)^* + d)^*e^*c$

   v) $b^*a(bb^*a)^*a(a + b)^*$

d) Pode ser importante programar-se o reconhecimento de uma linguagem regular directamente a partir de uma expressão regular $D(1)$ que a represente. Por exemplo, o reconhecimento de $(ab)^*\$$ ($\$$ é um símbolo terminador não pertencente a T) pode ser programado na linguagem Pascal como se segue (certas simplificações não foram feitas para haver uma correspondência directa entre o código e a expressão regular):

```
read(x);                {leitura do simbolo de avanco}
if x in ['a'] then
    begin
        while x in ['a'] do
            begin
                if x = 'a' then read(x) else ERRO;
                if x = 'b' then read(x) else ERRO
            end
    end;
if x = '$' then ACEITA else ERRO
```

Considere que R é uma expressão regular na qual não ocorre o símbolo $\emptyset$. Escreva um pseudo-código $cod(R)$ que a reconheça, considerando separadamente os seguintes casos possíveis para estrutura de R, e supondo que `x` é uma variável que contém o carácter de avanço:

i) λ

ii) a (símbolo do alfabeto)

iii) PQ (supõe-se que P não é um produto de expressões regulares)

iv) $P + Q$

v) P^*

e) Faça um programa que, dada uma expressão regular R que não contém o símbolo $\emptyset$, produz um programa em Pascal que reconhece a linguagem denotada por R, se esta for $D(1)$. O programa Pascal deve aceitar uma sequência de caracteres passíveis de impressão terminada por `<RETURN>`, afixando a mensagem `ACEITA` ou `REJEITA`, consoante a palavra pertença, ou não, à linguagem denotada por R.

Exercício 11 Apresente um sistema de equações para caracterizar cada uma das seguintes linguagens. Resolva os sistemas de equações e compare os resultados com os obtidos no Exercício 2.

a) Todas as palavras sobre $\{a, b, c, d\}$ com pelo menos um a ou um b.

b) Todas as palavras sobre $\{a, b, c\}$ em que cada a é seguido de um b.

c) Todas as palavras sobre $\{a, b, c\}$ com um número par de símbolos b.
(Admita que zero é um número par.)

d) $\{a^m b^n \mid m, n \geq 0\}$.

Exercício 12 (TE 96/97) O seguinte sistema de equações descreve de forma muito simplificada as listas de parâmetros em Pascal:

$$\begin{array}{ll} L = P \ + \ P; L & \% \text{ Listas de parâmetros} \\ P = vI : t \ + \ pI & \% \text{ Parâmetros} \\ I = i \ + \ i, I & \% \text{ Listas de identificadores} \end{array}$$

a) Linearize o sistema transformando-o num sistema linear direito equivalente.

b) Resolva o sistema dado e o sistema obtido na alínea anterior.

Exercício 13 (TE 97/98) Considere o seguinte sistema:

$$\begin{array}{ll} S = I \ + \ I; S & \% \text{ Sequência de instruções} \\ I = v? \ + \ v! \ + \ v := E & \% \text{ Instrução} \\ E = A \ + \ A \Box E & \% \text{ Expressão} \\ A = v \ + \ n & \% \text{ Átomo} \end{array}$$

Para simplificar, representou-se por v, n e $\Box$ respectivamente uma variável, um número e uma operação arbitrários. As instruções $v?$ e $v!$ são respectivamente de leitura e escrita de v.

a) Resolva o sistema directamente.

b) Linearize o sistema e resolva o sistema linear direito encontrado.

Exercício 14 Represente a árvore de cada uma das seguintes linguagens e classifique-as quanto à racionalidade. Sempre que possível, apresente um autómato finito determinista que reconheça a linguagem.

a) a^*b^*

b) $(abc)^*ab$

c) $c^*(a+b)c^*$

d) $\{a^n b^{2n} \mid n \geq 0\}$

e) $\{a^m b^n c^{m+n} \mid m, n \geq 0\}$

f) $\{pp \mid p \in \mathcal{L}(a^*ba^*)\}$

Exercício 15 (TE 86/87) Sejam T um alfabeto, L uma linguagem sobre T e $x \in T^*$. A *linguagem derivada de L em ordem a x* define-se por $\partial_x L = \{y \mid xy \in L\}$.

a) Interprete a linguagem $\partial_x L$ em termos da representação de L em árvore.

b) Mostre que se L for regular, o conjunto $\{\partial_x L \mid x \in T^*\}$ é finito.

c) Mostre que:

i) $\partial_\lambda L = L$

ii) $\lambda \in \partial_x L \Leftrightarrow x \in L$

iii) $\partial_y(\partial_x L) = \partial_{xy} L$

d) Suponha que L está representada por uma expressão regular P. Defina, por indução na estrutura de P, a função $\partial'_a P$, para $a \in T$, que devolve uma expressão regular que denota $\partial_a L$.

2 Autómatos finitos

Exercício 16 (LP 2.1.3) Construa AFDs que aceitem cada uma das seguintes linguagens.

a) $\{w \in \{a,b\}^* :$ cada a em w é imediatamente precedido e seguido por um $b\}$

b) $\{w \in \{a,b\}^* : w$ contém $abab$ como subpalavra $\}$

c) $\{w \in \{a,b\}^* : w$ não contém nem aa nem bb como subpalavras $\}$

d) $\{w \in \{a,b\}^* : w$ tem um número ímpar de a's e um número par de b's $\}$

e) $\{w \in \{a,b\}^* : w$ contém as subpalavras ab e $ba\}$

Exercício 17 Considere o alfabeto $Act = \{input, send, deliver, ack, loss, output, idle\}$, em que cada símbolo representa uma acção realizada por um sistema constituído por um receptor R, um emissor E e um meio de comunicação M, exibindo o seguinte comportamento. R recebe uma mensagem do exterior (*input*) e entrega-a a M (*send*). M tenta então entregá-la a E. Caso consiga (*deliver*), deve confirmá-lo a R (*ack*). Caso contrário, existe uma perca de informação (*loss*), pelo que R deve tentar de novo (*send*). Quanto a E, sempre que recebe uma mensagem de M, deve enviá-la para o exterior (*output*) e avisar R (*idle*) de que este pode enviar uma próxima mensagem. Especifique um AFD que aceite a linguagem $L \subseteq Act^*$ das palavras que constituem sequências válidas de trabalho do sistema descrito.

Exercício 18 (TE 84/85) Interprete os símbolos do alfabeto $T = \{12, 21, 13, 31, 23, 32\}$ como peças de um dominó em que cada metade possui uma, duas, ou três pintas, e onde não existem dobles. Assuma que existe um número arbitrário de ocorrências de cada peça. Uma palavra é uma sequência de peças na qual faces justapostas têm obrigatoriamente igual número de pintas. Apresente um AFD que reconheça a linguagem de todas as palavras (possivelmente vazias) que começam e acabam com uma pinta.

Exercício 19 (TE 89/90) Sejam k e n dois números inteiros tais que $0 < k < n$. Considere o alfabeto $T = \{0, 1, \ldots, k\}$ e a linguagem L, sobre T, formada por todas as palavras $a_1 a_2 \ldots a_i$ tais que o inteiro $a_1 + a_2 + \cdots + a_i$ é um múltiplo de n. Admita que zero é múltiplo de qualquer inteiro positivo.

a) Mostre que L é regular.

b) Construa um autómato finito e uma expressão regular para a linguagem L, no caso em que $k = 1$ e $n = 3$.

Exercício 20 (TE 97/98) Seja $\mathcal{A} = (\{a, b\}, \{1, 2, 3, 4, 5, 6\}, 1, \delta, \{6\})$ um autómato finito não-determinista, em que a função de transição

$$\delta : \{1, 2, 3, 4, 5, 6\} \times \{a, b, \lambda\} \to \wp(\{1, 2, 3, 4, 5, 6\})$$

é definida pela tabela

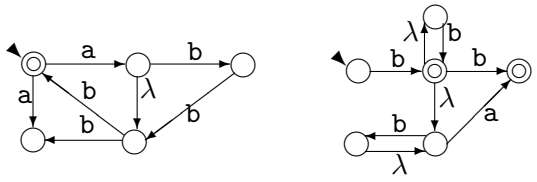
δ	a	b	λ
1	$\{4\}$	$\{5\}$	$\{2\}$
2	$\{3\}$	$\emptyset$	$\emptyset$
3	$\{1\}$	$\{6\}$	$\{5\}$
4	$\emptyset$	$\{5\}$	$\emptyset$
5	$\emptyset$	$\{2\}$	$\emptyset$
6	$\{5\}$	$\emptyset$	$\emptyset$

.

a) Defina um autómato finito determinista $\mathcal{A}'$ equivalente a $\mathcal{A}$, aplicando o algoritmo dado.

b) Determine uma expressão regular para a linguagem reconhecida por $\mathcal{A}$.

Exercício 21 Para cada um dos seguintes AFND:



a) Indique quais das palavras são aceites: `aa`, `aba`, `abb`, `ab`, `abab`, `ba`, `bb`, `b`, `bba`, λ.

b) Encontre um AFD que reconheça a mesma linguagem.

c) Caracterize um AFD reduzido equivalente.

Exercício 22 Desenhe os diagramas de transição de estado para os AFND que aceitam cada uma das seguintes linguagens.

a) $(\mathtt{ab})^*(\mathtt{ba})^* + \mathtt{aa}^*$

b) $((\mathtt{ab} + \mathtt{aab})^*\mathtt{a}^*)^*$

c) A linguagem do Exercício 16 b).

Exercício 23 Escreva um programa que tendo como entrada uma expressão regular R qualquer, produz como resultado um programa em Pascal que reconhece a linguagem $\mathcal{L}(R)$.
Sugestão: Primeiro, o programa constrói (numa tabela, por exemplo) a relação de transição de um AFND H que reconhece $\mathcal{L}(R)$. De seguida, calcula um AFD equivalente a H.

Exercício 24 (TE 88/89) Seja L uma linguagem regular. Caracterize um autómato finito que reconheça a linguagem L^{-1}.

Exercício 25 (LP 2.6.1) Seja $D = \{0, 1\}$. Considere o alfabeto $T = D \times D \times D$. Cada símbolo de T pode ser visto como uma "coluna" de uma operação de adição em notação binária. Por exemplo, a operação

$$\begin{array}{cccccc} & 0 & 1 & 1 & 0 \\ + & 1 & 0 & 1 & 0 \\ \hline 1 & 0 & 0 & 0 & 0 \end{array}$$ pode ser representada por $\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \in T^*$.

Mostre que a linguagem das palavras de T^* que representam adições correctas é regular.

Exercício 26 (adaptação de LP 2.1.4) Um *tradutor finito determinista* (TFD) é um dispositivo semelhante a um autómato finito determinista, mas tendo o objectivo de "traduzir" palavras de entrada válidas noutras palavras (de saída).

A sua operação pode ser descrita do modo seguinte. Iniciando num dado estado com uma palavra inscrita na fita (de entrada), em cada passo o TFD transita de um estado para um outro, determinado com base no símbolo corrente, tal como um AFD. No entanto, sempre que uma transição de estado ocorre, o TFD emite (para uma fita de saída) uma sequência eventualmente vazia de símbolos. Por isso, um diagrama de estados de um TFD é parecido com um diagrama de estados de um AFD em que os arcos correspondentes às transições estão etiquetados com um par símbolo/palavra.

a) Considere o alfabeto $T = \{x, y, z\}$. Desenhe um diagrama de estados para o TFD que, dada um palavra $w \in T^*$, produz uma palavra x^n, em que n é o número de ocorrências de zyz em w.

b) Elabore a especificação formal de um TFD, incluindo as noções de configuração, transição num passo e linguagem aceite. Repare que um TFD M pode ser visto como um dispositivo que calcula uma função $f_M : T^* \to T^*$. Defina pois, também, o conceito de *função computada por um TFD*.

c) Seja $T = \{I\}$. Considere a representação unária dos números naturais, isto é, $[n] = I^n$ (por exemplo $[4] = IIII$). Defina TFDs $M_{(i,j)}$ que calculem as seguintes funções: $f_{M_{(i,j)}}([n]) = [i * n + j]$, para $i, j \geq 0$ e $n > 0$.

Exercício 27 (TE 93/94) Uma *elipse* consiste na supressão de uma ou mais palavras consecutivas numa frase. Por exemplo

Uma *elipse* consiste $\cdots$ numa frase.

é uma elipse da frase anterior. Seja L uma linguagem regular sobre um alfabeto T. Definimos agora a elipse de L como sendo a linguagem sobre $T \cup \{\cdots\}$ tal que

$$elipse(L) = \{v \cdots u \mid vwu \in L \text{ para algum } w \in T^+\}.$$

Note que o símbolo $\cdots$ ocorre uma e uma só vez em cada palavra de $elipse(L)$. Mostre como é que dado um autómato finito M, que aceita uma linguagem L qualquer, se pode definir um autómato finito M' que aceite $elipse(L)$.

Exercício 28 (TE 93/94) O Datalog é uma linguagem semelhante ao Prolog, mas que não pode usar símbolos de função. Eis um exemplo de uma cláusula Datalog, usando v para representar as variáveis e c para representar as constantes:

$$c(v, v, v) \text{ :- } c(v), c(c), c(v, c).$$

Cada cláusula Datalog pode ser definida pela seguinte gramática independente do contexto $G = (\{\text{:-}, v, c, (,), \_\}, \{S, A, B, N, L, T\}, S, R)$ sendo R o seguinte conjunto de produções.

$$\begin{aligned}
S &\to A \text{ :- } B \\
B &\to A \mid A, B \\
A &\to N \mid N(L) \\
L &\to T \mid T, L \\
T &\to N \mid v \\
N &\to c
\end{aligned}$$

Apresente um autómato finito determinista que aceite $\mathcal{L}(G)$.

Exercício 29 (TE 93/94) A *Prefix Extended Notation (PEN)* permite especificar gramáticas independentes do contexto de forma mais concisa. Para tal usam-se nos corpos das produções os símbolos especiais $\textcircled{?}$, $\textcircled{*}$ e $\textcircled{+}$ antes de um símbolo terminal ou não-terminal x. O significado é: $\textcircled{?}x$ denota zero ou uma, $\textcircled{*}x$ zero ou mais e $\textcircled{+}x$ uma ou mais ocorrências de x. Por exemplo, as seguintes gramáticas representam ambas a

linguagem dos termos aritméticos, sendo a gramática da direita equivalente à gramática PEN da esquerda.

$$\begin{array}{ll} T \to F\ \textcircled{*}P & T \to FX \\ P \to \times F \mid /F & X \to PX \mid \lambda \\ F \to \textcircled{?}-\ \textcircled{+}D & P \to \times F \mid /F \\ D \to 0 \mid \cdots \mid 9 & F \to YZ \\ & Y \to - \mid \lambda \\ & Z \to ZD \mid D \\ & D \to 0 \mid \cdots \mid 9 \end{array}$$

a) Usando os símbolos n e t para representar os símbolos não-terminais e os símbolos terminais, respectivamente, escreva a expressão regular que representa a linguagem de todas as sequências de produções PEN.

b) Apresente um autómato finito determinista para a mesma linguagem.

c) Prove que para qualquer gramática independente do contexto é possível obter uma gramática PEN equivalente que não usa λ.

Exercício 30 (TE 97/98) Seja $\mathcal{A} = (T, Q, q_I, \delta, F)$ um autómato finito determinista e $\mathcal{L}(\mathcal{A})$ a linguagem por ele reconhecida. Dado um alfabeto U disjunto de T, seja

$$L = \{x_0 a_1 x_1 \dots a_n x_n : n \geq 0, x_0, x_1, \dots, x_n \in U^*, a_1, \dots, a_n \in T \text{ e } a_1 \dots a_n \in \mathcal{L}(\mathcal{A})\}.$$

(L é a linguagem formada pelas palavras $a_1 \dots a_n$ de $\mathcal{L}(\mathcal{A})$ onde cada símbolo a_i está rodeado por palavras arbitrárias x_{i-1} e x_i sobre U, eventualmente vazias.) Caracterize um autómato finito determinista que reconhece L.

Exercício 31 (TE 97/98) Seja $T = \{a, b\}$ e $n > 0$. Caracterize um autómato finito determinista $\mathcal{A} = (T, Q, q_I, \delta, F)$ que reconheça a linguagem formada por todas as palavras em que nem a nem b têm mais de n ocorrências consecutivas. (Por exemplo, se $n = 2$, as palavras λ, a, aa e $abbaa$ pertencem à linguagem, mas $abbbaa$ não pertence.)

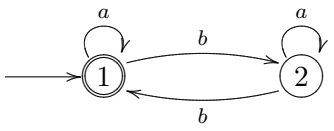
Exercício 32 Defina um AFD que aceita a linguagem sobre $\{a, b\}$ das palavras de comprimento ímpar e com um número par de a's.

Exercício 33 (TE 97/98) Dado um alfabeto T, $a \in T$ e $L \subseteq T^*$, o conjunto das palavras de L que terminam em a é a linguagem

$$Termina_a(L) = \{x \in L \mid \exists_{w \in T^*}\ x = wa\}.$$

Por exemplo, $Termina_a(\{\lambda, a, b, aa, ab, ba, bb, aaa, aab\}) = \{a, aa, ba, aaa\}$.

1. A partir do seguinte autómato finito determinista, que reconhece uma linguagem M sobre o alfabeto $T = \{a, b\}$ que não interessa precisar, crie um novo autómato finito determinista que reconheça $Termina_a(M)$:



2. Seja $\mathcal{A} = (T, Q, q_I, \delta, F)$ um autómato finito determinista e $\mathcal{L}(\mathcal{A})$ a linguagem por ele reconhecida. Caracterize um autómato finito determinista $\mathcal{A}'$ tal que $\mathcal{L}(\mathcal{A}') = Termina_a(\mathcal{L}(\mathcal{A}))$.

Exercício 34 (TE 97/98) Usando o algoritmo por aproximações sucessivas, construa o autómato reduzido do autómato finito $\mathcal{A} = (\{a, b\}, \{1, 2, 3, 4, 5, 6, 7\}, 1, \delta, \{5, 6, 7\})$ em que a função de transição δ é dada pela seguinte tabela:

δ	a	b
1	6	4
2	5	4
3	5	3
4	3	6
5	1	7
6	2	7
7	3	5

3 Gramáticas independentes do contexto

Exercício 35 (adaptação de Rév 1.4 e 3.2) Classifique as seguintes gramáticas e as respectivas linguagens geradas, segundo a hierarquia de Chomsky.

a) $G = (\{a, b\}, \{S, A, B\}, S, P)$, com P o seguinte conjunto de produções:

$$\begin{aligned} S &\to aAB \\ Ab &\to SBb \\ Aa &\to SaB \\ bB &\to a \\ B &\to SA \mid ab \end{aligned}$$

b) $G = (\{a, b, c\}, \{S, A, B, C, D\}, S, P)$, com P o seguinte conjunto de produções:

$$\begin{aligned} S &\to aBc \mid AbC \mid \lambda \\ A &\to aBC \mid ABc \mid AbS \mid SbC \\ B &\to Abc \mid abC \mid SBc \\ C &\to BSC \mid Abc \\ D &\to SaB \mid ab \end{aligned}$$

Exercício 36 Defina gramáticas independentes do contexto que gerem cada uma das seguintes linguagens.

a) $\{wcw^{-1} \mid w \in \{a, b\}^*\}$

b) $\{ww^{-1} \mid w \in \{a, b\}^*\}$

c) $\{w \in \{a, b\}^* \mid w = w^{-1}\}$

d) $\{wc^n \mid w \in \{a, b\}^*, n = |w|\}$

e) $\{a^m b^n \mid m \geq 0, n \geq 0, m \neq n\}$

f) $\{a^n b^m c^k \mid 0 \leq n < m = n + k\}$

Exercício 37 (LP 3.1.4) Especifique uma gramática independente do contexto, sobre o alfabeto $T = \{a, b, (,), \circledast, \emptyset, \oplus, \lambda\}$, que gere a linguagem de todas as expressões regulares sobre o alfabeto $\{a, b\}$.

Exercício 38 Verifique se cada uma das seguintes gramáticas é ambígua e, em caso afirmativo, defina uma GIC não ambígua que gere a mesma linguagem.

a) $G = (\{(,)\}, \{S\}, S, \{S \to () \ , \ S \to (S) \ , \ S \to SS\})$.

b) $G = (\{a, (,), -, +, *\}, \{E\}, E, \{E \to a \ , \ E \to (-E) \ , \ E \to (E + E) \ , \ E \to E * E\})$.

c) $G = (\{a, (,), -, +, *\}, \{E\}, E, \{E \to a \ , \ E \to (-E) \ , \ E \to (E+E) \ , \ E \to (E*E)\})$.

d) $G = (\{a, b\}, \{A\}, A, \{A \to AA \ , \ A \to aAb \ , \ A \to \lambda\})$.

Exercício 39 (TE 02/03) Uma escrita mais simétrica da instrução if C then S_1 else S_2 é S_1 if C else S_2. A seguinte gramática descreve esta variante da instrução condicional: $\mathcal{G} = (\{b_1, \ldots, b_n, c_1, \ldots, c_k, \text{if}, \text{else}\}, \{S, B, C\}, S, \mathcal{R})$, com $\mathcal{R}$ o conjunto de produções

$$\begin{array}{ll} S \longrightarrow B \mid S \text{ if } C \text{ else } S & \% \text{ Instrução condicional} \\ B \longrightarrow b_1 \mid \cdots \mid b_n & \% \text{ Instrução básica} \\ C \longrightarrow c_1 \mid \cdots \mid c_k & \% \text{ Condição booleana} \end{array}$$

Mostre que a gramática é ambígua e defina uma gramática equivalente não ambígua.

Exercício 40 (TE 89/90) A linguagem de acesso a uma base de dados é descrita pela seguinte GIC $G = (\{\text{abrir}, \text{fechar}, \text{ler}, \text{act}\}, \{S, C, E, L, A\}, S, P)$, com P o seguinte conjunto de produções:

$$\begin{array}{ll} S \to \text{abrir } C \text{ fechar} & \text{(Sessão)} \\ C \to \lambda \mid EC & \text{(Ciclo)} \\ E \to L \mid A & \text{(Execução)} \\ L \to \lambda \mid \text{ler } L & \text{(Leitura)} \\ A \to \lambda \mid \text{act ler } A \mid \text{act } A \text{ ler} & \text{(Actualização)} \end{array}$$

Prove que G é ambígua e apresente uma gramática independente do contexto, não ambígua, que gere a mesma linguagem.

4 Gramáticas de atributos e de acções

Exercício 41 (TE 87/88) Uma árvore finita e não vazia, de nós rotulados por elementos de um conjunto T, pode ser representada por uma expressão $a(A_1, \ldots, A_n)$, em que $a \in T$ é o rótulo da raiz, e $A_1, \ldots, A_n$ são as subárvores da raiz. Se a árvore tiver um único nó rotulado por a, será representada pela expressão a. Considere a seguinte GIC destas expressões, $G = (T, N, S, R)$, com $T = \{(,), , , a, b, \$\}$, $N = \{S, A, L, T\}$ e R o seguinte conjunto de produções:

$$\begin{array}{l} S \to A\$ \\ A \to T \mid T(L) \\ L \to A \mid L, L \\ T \to a \mid b \end{array}$$

Especifique uma gramática de atributos e uma gramática de acções que associe, a cada palavra p, a altura da árvore que p representa. Por exemplo, a altura de $a(b, a(b))\$$ é três.

Exercício 42 Considere a seguinte gramática independente do contexto $G = (T, N, S, R)$ com $T = \{\wedge, \Rightarrow, \forall, a, x, i, (,), \$\}$, $N = \{E, S, P, A, V, I\}$ e R o seguinte conjunto de produções:

$$\begin{array}{l} S \to E\ \$ \\ E \to E \wedge E \mid E \Rightarrow E \mid \forall\ V\ E \mid (E) \mid P(A) \\ A \to \lambda \mid V A \\ P \to aI \\ V \to xI \\ I \to i \mid Ii \end{array}$$

Especifique uma gramática de atributos e uma gramática de acções que associe, a cada derivação de uma palavra $p \in \mathcal{L}(G')$, o conjunto das variáveis livres da fórmula lógica que p representa.

Exercício 43 Pretende-se a especificação duma gramática de atributos para uma linguagem de "programação" muito simples, permitindo apenas declarações de variáveis e instruções de afectação. Tendo em vista este objectivo, resolva cada uma das seguintes alíneas.

a) Especifique uma GIC para expressões booleanas admitindo os habituais operadores $\wedge, \vee, \neg$, as constantes `true` e `false`, e variáveis. A gramática deve reflectir as associatividades e as precedências usuais dos operadores booleanos.

b) Especifique uma GIC para expressões aritméticas admitindo os operadores $+, -, *, /$, constantes inteiras, e variáveis. Integre-a com a gramática anterior usando os operadores relacionais $<$ e $=$. Tenha de novo em conta a associatividade e precedência usual de todos os operadores envolvidos.

c) Defina atributos e regras semânticas que permitam associar à raiz de cada árvore de derivação de expressões aritméticas ou booleanas o seu valor.
Sugestão: Associe ao símbolo inicial da gramática um atributo herdado que "represente" o valor das variáveis que aparecem nas expressões.

d) Estenda a gramática de atributos anterior para tratar programas que consistem de uma série de declarações *var* : *tipo* (sendo o tipo `int` ou `bool`) seguida de uma série de afectações *var* := *expr*. Pretende-se que tal gramática permita associar a um programa correcto o seu estado final de execução, representado pelos valores finais das variáveis nele declaradas. Todo os erros possíveis (tais como variáveis não declaradas e conflitos de tipos) deverão ser detectados.

Exercício 44 (TE 88/89) Considere a seguinte linguagem de programação, cujos programas são sequências de instruções, possivelmente etiquetadas. Uma instrução é uma afectação (af), um salto (goto E) ou um salto condicional (goto E if cond). A GIC $G = (T, N, S, R)$ descreve a linguagem de tais programas, onde $T = \{\text{af}, \text{goto}, \text{if}, \text{cond}, :, ;, a, \ldots, z, \$\}$, $N = \{S, P, I, C, E\}$ e R é o seguinte conjunto de produções:

$$\begin{array}{l} S \to P\$ \\ P \to P; I \mid I \\ I \to E : C \mid C \\ C \to \text{af} \mid \text{ goto } E \mid \text{ goto } E \text{ if cond} \mid \lambda \\ E \to a \mid \cdots \mid z \end{array}$$

Especifique uma gramática de atributos que associe, a cada programa p, o conjunto dos pares (e, n) em que e é uma etiqueta usada numa instrução de salto mas nenhuma instrução do programa é etiquetada por e (e é usada e não declarada), e n é o número dessa instrução de salto. Por exemplo, para o programa

$$\text{goto } z \,;\, a : \text{ af } ;\ \text{goto } c \,;\ \text{goto } z \,;\ \text{goto } a \ \$$$

obter-se-ia o conjunto $\{(z, 1), (c, 3), (z, 4)\}$.

Exercício 45 (TE 89/90) Considere uma linguagem de nomes de funções e respectivas variáveis locais. Cada função pode declarar as suas variáveis e pode possuir diversas funções locais. Um programa é uma função especial. A gramática seguinte descreve tais programas. $G = (\{\texttt{aa}, \texttt{bb}, \ldots, \texttt{zz}, \texttt{a}, \texttt{b}, \ldots, \texttt{z}, ;, \$\}, \{P, I, L, V, F\}, P, R)$, com R o seguinte conjunto:

$P \to ILF\$$	(Programa)
$I \to \texttt{aa} \mid \texttt{bb} \mid \cdots \mid \texttt{zz}$	(Identificador da função)
$L \to VL \mid \lambda$	(Lista de variáveis)
$V \to \texttt{a} \mid \texttt{b} \mid \cdots \mid \texttt{z}$	(Variável)
$F \to FF \mid ILF; \mid \lambda$	(Função)

Por exemplo, o seguinte programa identado

```
aa
{var}  a b c
  bb
  {var}        a d
  ;
  cc
  {var}        a e f
    dd
    {var}      a d g
    ;
  ;
$
```

representa a palavra `aa a b c bb a d ; cc a e f dd a d g ; ; $`.

Defina atributos que permitam calcular quais as funções onde ocorrem as diferentes variáveis do programa. Especifique para tanto uma gramática de atributos e uma gramática de acções. No exemplo acima, o resultado seria

$$\{(\texttt{a}, \{\texttt{aa}, \texttt{bb}, \texttt{cc}, \texttt{dd}\}), (\texttt{b}, \{\texttt{aa}\}), (\texttt{c}, \{\texttt{aa}\}), (\texttt{d}, \{\texttt{bb}, \texttt{dd}\}), (\texttt{e}, \{\texttt{cc}\}), (\texttt{f}, \{cs\texttt{c}\}), (\texttt{g}, \{\texttt{dd}\})\}.$$

5 Autómatos de pilha

Exercício 46 Construa autómatos de pilha que reconheçam as linguagens $\{a^n b^n : n \geq 1\}$ e $\{a^n b^n : n \geq 0\}$, pelos critérios dos estados de aceitação e da pilha vazia.

Exercício 47 (TE 93/94) Muitas linguagens de programação permitem o uso de dois tipos de parêntesis em expressões aritméticas: os rectos [] e os curvos (). Por exemplo,

a expressão `(x[(i+j) div 2] + 5) * 3` é válida em Pascal. Considere a linguagem das sequências de parêntesis bem formadas, isto é, tal que o número de parêntesis a abrir e a fechar seja igual, e estejam bem "encaixados". Por exemplo, `([()])` e `[(([]())[])]` são sequências bem formadas, mas `)(`, `([)]` e `(([))` não o são. Especifique um autómato de pilha determinista capaz de reconhecer a linguagem descrita.

Exercício 48 (Rév 6.11) Considere a linguagem L sobre $\{a, b\}$ de todas as palavras não vazias em que o número de ocorrências do símbolo a é igual ao número de ocorrências de b. Apresente um autómato de pilha determinista que reconheça L.

Exercício 49 (TE 93/94) Seja $T = \{1\}$ e considere a representação unária dos números naturais, isto é, o natural m é representado por $1^m \in T^*$. Por exemplo, 4 é representado por 1111. Uma *expressão aditiva* é uma palavra sobre $\{1, \oplus, \ominus\}$ representável pela expressão regular

$$N((\oplus + \ominus)N)^*$$

onde $N = 1^*$.

Especifique um autómato de pilha que aceite a linguagem das expressões aditivas com valor não negativo.

Exercício 50 Construa autómatos de pilha que aceitem cada uma das linguagens seguintes.

a) $\{a^m b^n \mid m \leq n \leq 2m\}$

b) $\{w \in \{a, b\}^* \mid w = w^{-1}\}$

c) $\{w \in \{a, b\}^* \mid w$ contém duas vezes mais b's que a's$\}$

Exercício 51 (TE 88/89) Seja $A = (T, Q, q_i, \delta, F)$ um autómato finito determinista e & um símbolo que verifica $\& \notin T \cup Q$. Caracterize, a partir de A, um autómato de pilha determinista A' tal que $p \in \mathcal{L}(A) \Leftrightarrow p\&p^{-1} \in \mathcal{L}(A')$.

Exercício 52 (TE 97/98) Considere o alfabeto $T = \{a, b, c\}$. Dada uma palavra x sobre T, chamemos *redução* de x a qualquer palavra que se obtenha eliminando de x uma (e uma só) sub-palavra da forma abc. Por exemplo a palavra $aabcabcc$ tem duas reduções possíveis, $aabcc$ e $aabcc$. Seja L a linguagem sobre T formada por todas as palavras que por sucessivas reduções se transformam na palavra vazia, compreendendo a própria palavra vazia. Por exemplo, a palavra $aababccbc$ pertence a L visto que

$$aab\underline{abc}cbc \gg a\underline{abc}bc \gg abc \gg \lambda,$$

onde se indicou por $x \gg y$ que y é uma redução de x. Note-se que a palavra $aabbcc$ não tem nenhuma redução, portanto não pertence a L. A palavra anteriormente mostrada, $aabcabcc$, também não pertence a L porque após duas reduções por uma ordem arbitrária se transforma em ac, que não é redutível.

a) Defina a linguagem L por uma gramática independente do contexto.

b) Defina um autómato de pilha determinista que reconheça L.

Exercício 53 (TE/97/98) Considere a linguagem

$$L = \{a^i b^j a^k b^m \mid i, j, k, m \geq 1,\ k \geq j,\ i + j = k + m\}$$

sobre o alfabeto $\{a, b\}$.

a) Mostre que L não é uma linguagem regular.

b) Defina um autómato de pilha determinista que reconheça L.

Exercício 54 (TE/97/98) Considere a seguinte linguagem

$$L = \{(ab)^i c^k a^j : j \geq i \geq 1, k \geq 1\}$$

sobre o alfabeto $\{a, b, c\}$.

a) Mostre que L não é uma linguagem regular.

b) Defina um autómato de pilha determinista que reconheça L.

6 Análise sintáctica

Exercício 55 Considere a seguinte GIC $G = (T, N, S, R)$, com $T = \{begin, d, semi, end, s\}$, $N = \{P, X, Y\}$, $S = P$ e R o seguinte conjunto de produções:

$$\begin{aligned} P &\rightarrow begin\ d\ semi\ X\ end \\ X &\rightarrow d\ semi\ X \\ X &\rightarrow s\ Y \\ Y &\rightarrow \lambda \\ Y &\rightarrow semi\ s\ Y \end{aligned}$$

Mostre que G é LL(1) e construa a tabela de transições para o respectivo analisador sintáctico determinista. Ilustre a sua operação na palavra de entrada *begin d semi s semi s end*.

Exercício 56 Considere a seguinte gramática independente do contexto $G = (T, N, S, R)$ com $T = \{\wedge, \Rightarrow, \forall, a, x, i, (,), \$\}$, $N = \{E, S, P, A, V, I\}$ e R o seguinte conjunto de produções:

$$\begin{aligned} S &\rightarrow E\ \$ \\ E &\rightarrow E \wedge E \mid E \Rightarrow E \mid \forall\ V\ E \mid (E) \mid P(A) \\ A &\rightarrow \lambda \mid V\,A \\ P &\rightarrow aI \\ V &\rightarrow xI \\ I &\rightarrow i \mid Ii \end{aligned}$$

a) Transforme G numa gramática G' LL(1).

b) Construa a tabela de análise sintáctica correspondente.

c) Apresente um reconhecedor recursivo descendente para $\mathcal{L}(G')$.

Exercício 57 (TE 87/88) Uma árvore finita e não vazia, de nós rotulados por elementos de um conjunto T, pode ser representada por uma expressão $a(A_1, \ldots, A_n)$, em que $a \in T$ é o rótulo da raiz, e $A_1, \ldots, A_n$ são as subárvores da raiz. Se a árvore tiver um único nó rotulado por a, será representada pela expressão a. Considere a seguinte GIC destas expressões, $G = (T, N, S, R)$, com $T = \{(,), , , a, b, \$\}$, $N = \{S, A, L, T\}$ e R o seguinte conjunto de produções:

$$\begin{aligned} S &\to A\$ \\ A &\to T \mid T(L) \\ L &\to A \mid L, L \\ T &\to a \mid b \end{aligned}$$

Defina uma gramática LL(1) equivalente à dada e construa a respectiva tabela de análise sintáctica.

Exercício 58 (TE 88/89) Considere a seguinte linguagem de programação, cujos programas são sequências de instruções, possivelmente etiquetadas. Uma instrução é uma afectação (af), um salto (goto E) ou um salto condicional (goto E if cond). A GIC $G = (T, N, S, R)$ descreve a linguagem de tais programas, onde $T = \{\text{af}, \text{goto}, \text{if}, \text{cond}, : , ; , a, \ldots, z, \$\}$, $N = \{S, P, I, C, E\}$ e R é o seguinte conjunto de produções:

$$\begin{aligned} S &\to P\$ \\ P &\to P; I \mid I \\ I &\to E : C \mid C \\ C &\to \text{af} \mid \text{ goto } E \mid \text{ goto } E \text{ if cond} \mid \lambda \\ E &\to a \mid \cdots \mid z \end{aligned}$$

Defina uma gramática LL(1) equivalente à dada e construa a respectiva tabela de análise sintáctica.

Exercício 59 Considere a seguinte gramática $(\{a, b\}, \{S, A, B\}, S, R)$, com R o seguinte conjunto de produções:

$$\begin{aligned} S &\to A\$ \\ A &\to AB \mid B \\ B &\to aAb \mid ab \end{aligned}$$

a) Construa o AFD dos itens válidos LR(0) e justifique que a gramática é LR(0).

b) Apresente a respectiva tabela de análise sintáctica, defina o autómato de pilha determinista associado, e ilustre o seu funcionamento com a palavra $abab\$$.

c) Usando o mesmo exemplo, verifique que a análise LR(0) corresponde à derivação direita.

Exercício 60 Repita o exercício anterior para as seguintes gramáticas:

a) Gramática $(\{\$, +, b, (,)\}, \{S, A, T\}, S, R)$, com R o conjunto de produções:

$$\begin{aligned} S &\to A\$ \\ A &\to A + T \mid T \\ T &\to b \mid (A) \end{aligned}$$

Reconheça a palavra $b + (b)\$$.

b) Gramática $(\{\$, i, v\}, \{P, L\}, P, R)$ de produções

$$P \to iL\$$$
$$L \to Lv \mid \lambda$$

Reconheça a palavra $iv\$$.

Exercício 61 Mostre que a gramática $(\{\$, +, *, b, (,)\}, \{S, A, T, F\}, S, R)$, com R o conjunto de produções:

$$S \to A\$$$
$$A \to A + T \mid T$$
$$T \to T * F \mid F$$
$$F \to b \mid (A)$$

não é LR(0).

Exercício 62 Considere a seguinte gramática $G = (\{=, *, id\}, \{S, P, L, R\}, S, Q)$, com Q o seguinte conjunto de produções:

$$S \to P$$
$$P \to L = R \mid R$$
$$L \to *R \mid id$$
$$R \to L$$

a) Construa o autómato finito determinista dos itens válidos LR(1).

b) Indique, justificando, se a gramática é LR(0), LR(1) e/ou LALR(1).

c) Apresente as tabelas de análise sintáctica LR(0), LR(1) e LALR(1), quando fizer sentido.

d) Faça o reconhecimento da sequência $*id = id$ e escreva a derivação direita desta palavra.

Exercício 63 (TE 89/90) Considere uma linguagem de nomes de funções e respectivas variáveis locais. Cada função pode declarar as suas variáveis e pode possuir diversas funções locais. Um programa é uma função especial. A gramática seguinte descreve tais programas. $G = (\{\texttt{aa}, \texttt{bb}, \ldots, \texttt{zz}, \texttt{a}, \texttt{b}, \ldots, \texttt{z}, ;, \$\}, \{P, I, L, V, F\}, P, R)$, com R o seguinte conjunto:

$P \to ILF\$$	(Programa)
$I \to \texttt{aa} \mid \texttt{bb} \mid \cdots \mid \texttt{zz}$	(Identificador da função)
$L \to VL \mid \lambda$	(Lista de variáveis)
$V \to \texttt{a} \mid \texttt{b} \mid \cdots \mid \texttt{z}$	(Variável)
$F \to FF \mid ILF; \mid \lambda$	(Função)

Por exemplo, o seguinte programa identado

```
aa
{var}  a b c
  bb
  {var}        a d
  ;
  cc
```

```
{var}        a e f
  dd
  {var}      a d g
  ;
;
$
```

representa a palavra `aa a b c bb a d ; cc a e f dd a d g ; ; $`.

a) Defina uma gramática LL(1) equivalente e construa a respectiva tabela de análise sintáctica, utilizando os algoritmos de grafos para determinação dos primeiros e seguintes.

b) Considere agora a seguinte gramática $G' = (\{i, v, ;\}, \{P', L', F'\}, P', R')$, com R':

$$\begin{aligned} P' &\rightarrow i\, L'\, F' \\ L' &\rightarrow L'\, v \mid \lambda \\ F' &\rightarrow i\, L'\, F'\, ; \mid \lambda \end{aligned}$$

i) Apresente o autómato finito dos itens válidos LR(1).

ii) Justifique se a gramática é LR(0), LR(1) e/ou LALR(1).

7 Máquinas de Turing

(Adaptados de Thomas A. Sudkamp, *Languages and Machines*, Addison-Wesley, 1997.)

Exercício 64 Construa máquinas de Turing com alfabeto de entrada $\{a, b\}$ que executam as seguintes operações. Em cada caso, quando a computação terminar, a máquina deve estar num estado q_f com a cabeça de leitura e escrita posicionada na primeira casa da fita.

a) Move a entrada uma casa para a direira. Configuração de entrada $q_I BuB$, resultado $q_f BBuB$.

b) Concatena à entrada o seu inverso. Configuração de entrada $q_I BuB$, resultado $q_f Buu^{-1}B$.

c) Insere uma casa em branco entre cada um dos símbolos de entrada. Por exemplo, a configuração de entrada $q_I BabaB$ resulta em $q_f BaBbBaB$.

d) Apaga os b's da entrada. Por exemplo, a configuração de entrada $q_I BbabaababB$ resulta em $q_f BaaaaB$.

Exercício 65 Construa máquinas de Turing com alfabeto de entrada $\{a, b\}$ que aceitam as seguintes linguagens por estados de aceitação.

a) $\{a^i b^j \mid i \geq 0, j \geq i\}$.

b) $\{a^i b^j a^i b^j \mid i, j > 0\}$.

c) Palavras com o mesmo número de a's e de b's.

Exercício 66 Construa uma máquina de Turing com duas fitas que aceita palavras nas quais cada a é seguido por um número crescente de b's. Mais precisamente, as palavras aceites têm a forma

$$ab^{n_1}ab^{n_2}\cdots ab^{n_k}$$

com $k > 0$ e $n_1 < n_2 < \cdots < n_k$.

Exercício 67 Prove que toda a linguagem recursivamente enumerável é aceite por uma máquina de Turing com um único estado de aceitação.

Exercício 68 Um método alternativo de aceitação de palavras é o seguinte: Uma palavra u é aceite por uma máquina de Turing M se a computação de M com entrada u **passar** por um estado de aceitação (sem terminar obrigatoriamente nesse estado). Com esta definição, uma palavra pode ser aceite mesmo que a computação da máquina não termine. Mostre que as linguagens **aceites por passagem em estado de aceitação** são precisamente as linguagens recursivamente enumeráveis.

8 Máquinas de estados e programas

Exercício 69 Defina uma máquina PC ("pocket calculator") com as seguintes características:

- Tem dois registos, identificados como o registo x e o registo y, cada um dos quais pode guardar um número natural.
- As operações são: incrementar ou decrementar qualquer dos registos (se o conteúdo de um registo for zero, a operação de decrementar esse registo mantém o valor zero); adicionar o conteúdo dos registos e guardar o resultado no registo y; idem, mas multiplicando o conteúdo dos registos.
- As condições são: verificar se um dos registos tem o conteúdo zero; verificar se os conteúdos de ambos são iguais; condição identicamente verdade.

Escreva programas para as seguintes tarefas:

- Colocar 0 no registo x.
- Colocar 1 no registo y.
- Calcular o factorial do valor inicial no registo x e colocar o resultado no registo y.

Exercício 70 Dado um alfabeto T e um símbolo $B \notin T$, defina uma máquina de Turing MT em que:

- Os estados de memória são os conteúdos de uma fita bi-infinita cujas células estão vazias ou contêm um símbolo de T, bem como a indicação de qual a célula que está a ser inspeccionada pela cabeça de leitura e escrita. Como de costume, convenciona-se que uma célula vazia contém o símbolo B. O número de células que contêm símbolos de T é finito.
- As operações são as que resultam da movimentação da cabeça para a esquerda ou para a direita, e da escrita de um símbolo de $T \cup \{B\}$ na célula corrente.

- As condições são: verificar se o símbolo na célula corrente é diferente de um símbolo de $T \cup \{B\}$ especificado; condição identicamente verdade.

Supondo que $a, b \in T$, escreva um programa que em qualquer palavra de entrada substitui todos os símbolos a por b.

9 Verificação pelo método de Floyd

Exercício 71 Sem usar a operação de multiplicação, escreva um programa para calcular o produto de dois números naturais, forneça as respectivas asserções de entrada e de saída e demonstre a sua correcção parcial e a sua terminação.

Exercício 72 Exercício análogo ao anterior para calcular a potência por um número natural de um número real $\neq 0$ sem usar a operação de exponenciação.

Exercício 73 Dada uma tabela linear $a[1..n]$ de números reais ($n \geq 1$), escreva um programa para calcular o valor máximo guardado na tabela com base em comparações efectuadas com a relação $<$ ou $\leq$. Especifique o programa e prove a sua correcção parcial e a sua terminação.

Exercício 74 Repita o Exercício 72 mas usando um algoritmo de complexidade logarítmica em função do expoente.

Exercício 75 Dada uma tabela linear ordenda $a[1..n]$ de números reais ($n \geq 1$) e um número real b, escreva um programa que implemente a pesquisa dicotómica de b em a. Especifique o programa e prove a sua correcção parcial e a sua terminação.