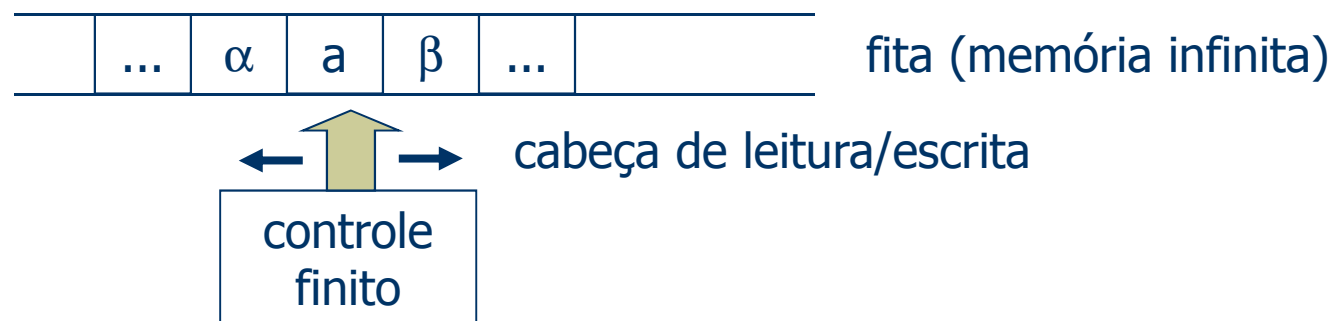


## 1.4 - Máquinas de Turing

---

- ♦ Esquemáticamente:



- ♦ Num movimento, dependendo do símbolo lido e do estado do controle finito, a máquina de Turing:
  - troca de estado
  - escreve um símbolo na fita
  - move sua cabeça de leitura/escrita uma posição para a direita ou uma posição para a esquerda ou a mantém na mesma posição.
- ♦ Definição: Uma máquina de Turing simples é uma 6-tupla  $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ , onde:
  - $Q$  - conjunto finito não vazio de estados
  - $\Sigma$  - alfabeto de entrada
  - $\Gamma$  - alfabeto da fita ( $\Sigma \subset \Gamma$ ;  $B \in \Gamma$ ;  $B \notin \Sigma$ )
  - $q_0 \in Q$  - estado inicial
  - $F \subseteq Q$  - conjunto de estados finais

# Máquinas de Turing

---

$\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{ \leftarrow, \downarrow, \rightarrow \}$  - função de transição

(  $\delta(q, a)$  é indefinida se  $q \in F, \forall a \in \Gamma$  )

- ♦ Definição: Uma configuração de  $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$  é um par  $(q, \alpha[a]\beta)$  onde  $q \in Q$ ;  $\alpha, \beta \in \Gamma^*$ ;  $a \in \Gamma$ , e os símbolos  $[ ]$  indicam a posição da cabeça de leitura/escrita.
- ♦ Definição: Transições da máquina de Turing:
  - a) se  $\delta(q, a) = (p, b, \downarrow)$  então  $(q, \alpha[a]\beta) \mapsto (p, \alpha[b]\beta)$
  - b) se  $\delta(q, a) = (p, b, \rightarrow)$  e  $\beta = c\beta'$  então  $(q, \alpha[a]\beta) \mapsto (p, \alpha b[c]\beta')$
  - c) se  $\delta(q, a) = (p, b, \leftarrow)$  e  $\alpha = \alpha'c$  então  $(q, \alpha[a]\beta) \mapsto (p, \alpha'[c]b\beta)$onde  $p, q \in Q$ ;  $\alpha, \beta, \alpha', \beta' \in \Gamma^*$ ;  $a, b, c \in \Gamma$ .
- ♦ Definição: Uma configuração  $(q, \alpha[a]\beta)$  é terminal se  $\delta(q, a)$  é indefinida. Se  $q \in F$ , a configuração terminal é uma configuração final.
- ♦ Definição: O processamento da máquina  $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$  com entrada  $w = c_1 \dots c_n$  é uma sequência de configurações:  $(q_0, \alpha_0[a_0]\beta_0), (q_1, \alpha_1[a_1]\beta_1), \dots$  tal que:
  - a)  $(q_0, \alpha_0[a_0]\beta_0) = (q_0, [c_1] \dots c_n)$
  - b)  $(q_i, \alpha_i[a_i]\beta_i) \mapsto (q_{i+1}, \alpha_{i+1}[a_{i+1}]\beta_{i+1})$
  - c) se a sequência for finita, então a última configuração é terminalonde:  $\alpha_i, \beta_i \in \Gamma^*$  ( $i = 0, 1, \dots$ );  $c_1 \dots c_n \in \Sigma^*$ ;  $a_0, a_1, \dots \in \Gamma$

# Máquinas de Turing

- ♦ Definição: Seja  $w \in \Sigma^*$  com  $w = aw'$ ,  $a \in \Sigma$ . Seja  $\beta \in \Gamma^+$  com  $\beta = b\beta'$ ,  $b \in \Gamma$ . A linguagem aceita por  $M$  (ou processamento aceitável) é definida por:  

$$L(M) = \{ w \in \Sigma^* \mid (q_0, [a]w') \mapsto^* (q, [b]\beta') \text{ com } q \in F \}$$
- ♦ Exemplo: Máquina de Turing que aceita  $L = \{ 0^n 1 0^n \mid n \geq 0 \}$   
 $M = (\{q_0, \dots, q_6\}, \{0, 1\}, \{0, 1, B\}, \delta, q_0, \{q_6\})$  com  $\delta$  dada por:

$\delta(q_0, 0) = (q_1, B, \rightarrow)$	<b>apaga o 0 mais à esquerda</b>
$\delta(q_0, 1) = (q_5, 1, \rightarrow)$	<b>não existe mais 0 à esquerda</b>
$\delta(q_1, 0) = (q_1, 0, \rightarrow)$	<b>avança na esperança de encontrar um 1</b>
$\delta(q_1, 1) = (q_2, 1, \rightarrow)$	<b>foi encontrado 1; apagar o 0 mais à direita</b>
$\delta(q_2, 0) = (q_2, 0, \rightarrow)$ $\delta(q_2, b) = (q_3, B, \leftarrow)$	<b>avança para o símbolo mais à direita</b>
$\delta(q_3, 0) = (q_4, B, \leftarrow)$	<b>apaga o 0 mais à direita</b>
$\delta(q_4, 0) = (q_4, 0, \leftarrow)$ $\delta(q_4, 1) = (q_4, 1, \leftarrow)$ $\delta(q_4, b) = (q_0, B, \rightarrow)$	<b>volta para o símbolo mais à esquerda</b>
$\delta(q_5, b) = (q_6, B, \downarrow)$	<b>aceita a cadeia</b>

# Máquinas de Turing

- ♦ Definição: Seja  $L$  uma linguagem.  $L$  é linguagem recursivamente enumerável se  $L = L(M)$  para alguma máquina de Turing  $M$ .

## Técnicas para construção de máquinas de Turing simples (MTS)

### 1. Fita com várias trilhas

Pode-se imaginar que a fita da MTS é constituída por  $k$  trilhas, para algum inteiro  $k$ . Isso é equivalente a imaginar que os símbolos que aparecem na fita são  $k$ -tuplas, com um componente colocado em cada trilha.

Exemplo: Máquina de Turing que calcula  $n-m$ , onde  $n$  e  $m$  são inteiros escritos em binário e  $n \geq m$ .

Podemos imaginar uma MT com uma fita de 3 trilhas, onde  $n$  é colocado na trilha 1 e  $m$ , na trilha 2. A trilha 3 está inicialmente em branco. Ao final, a trilha 3 irá conter o resultado  $n-m$  e as outras duas trilhas estarão em branco. Por exemplo, seja  $n = 12$  (1100) e  $m = 5$  (101):

configuração inicial				1	1	0	0				trilha 1
					1	0	1				trilha 2
											trilha 3

cabeça  
leitura/escrita

# Máquinas de Turing

configuração inicial

			1	1	0	0				trilha 1
				1	0	1				trilha 2
										trilha 3

cabeça leitura/escrita

- Então:  $M = (\{q_0, q_1\}, \{0, 1\}, \{0, 1, B\}, \delta, q_0, \{q_0\})$  com  $\delta$  dada por:

	(00B)	(01B)	(10B)	(11B)	(0BB)	(1BB)
q0	q0, BB0, ←	q1, BB1, ←	q0, BB1, ←	q0, BB0, ←	q0, BB0, ←	q0, BB1, ←
q1	q1, BB1, ←	q1, BB0, ←	q0, BB0, ←	q1, BB1, ←	q1, BB1, ←	q0, BB0, ←

- Para a configuração acima teremos:

$(q_0, (1BB)(11B)(00B)[(01B)]) \mapsto (q_1, (1BB)(11B)[(00B)](BB1)) \mapsto$   
 $(q_1, (1BB)[(11B)](BB1)(BB1)) \mapsto (q_1, [(1BB)](BB1)(BB1)(BB1)) \mapsto$   
 $(q_0, [ ](BB0)(BB1)(BB1)(BB1))$

configuração final

										trilha 1
										trilha 2
			0	1	1	1				trilha 3

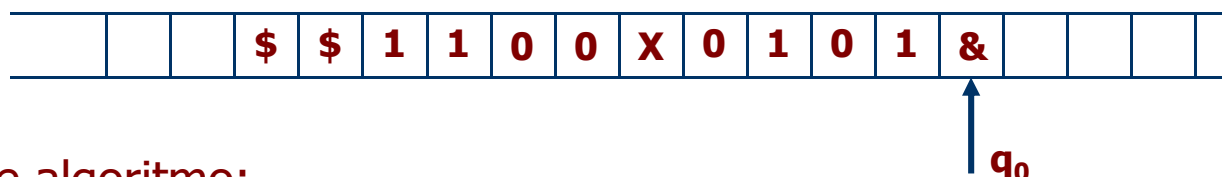
# Máquinas de Turing

## 2. Subrotinas

Pode-se considerar, sob determinadas condições, que uma máquina de Turing  $M_1$  é uma subrotina de outra máquina de Turing  $M_2$ . A idéia é a seguinte: Para “chamar”  $M_1$  a máquina  $M_2$  entra no estado inicial de  $M_1$ . Quando  $M_1$  entra num estado final (estado de “retorno”),  $M_2$  “recupera o controle”.

Exemplo: Máquina de Turing para calcular  $n+m$ , onde  $n$  e  $m$  são inteiros escritos em binário. Sejam  $n$  e  $m$  do mesmo tamanho (se os tamanhos forem diferentes é sempre possível acrescentar zeros à esquerda do menor). Por exemplo, seja  $n = 1100$  e  $m = 0101$ .

**configuração inicial**



Seja o seguinte algoritmo:

- a) se  $m = 0$ , então  $n$  é a soma;
- b)  $m = m-1$ ;  $n = n+1$ ; voltar para (a).

**Subrotina subtrai-um:**

$\delta(t, 0) = (t, 1, \leftarrow)$   
 $\delta(t, 1) = (r_1, 0, \leftarrow)$   
 $\delta(t, X) = \text{FIM}$

**Subrotina soma-um:**

$\delta(s, 0) = (r_2, 1, \rightarrow)$   
 $\delta(s, 1) = (s, 0, \leftarrow)$   
 $\delta(s, \$) = (r_2, 1, \rightarrow)$

$t, s$  - estados iniciais  
 $r_1, r_2$  - estados de retorno

# Máquinas de Turing

**configuração inicial**

			\$	\$	1	1	0	0	X	0	1	0	1	&				
--	--	--	----	----	---	---	---	---	---	---	---	---	---	---	--	--	--	--

Logo, o programa da máquina de Turing será:



$\delta(q, \&) = (t, \&, \leftarrow)$	<b>chama a subrotina subtrai-um</b>
$\delta(r_1, a) = (r_1, a, \leftarrow)$ , se $a \neq X$ $\delta(r_1, a) = (s, a, \leftarrow)$ , se $a = X$	<b>prepara-se para chamar soma-um chama soma-um</b>
$\delta(r_2, a) = (r_2, a, \rightarrow)$ , se $a \neq \&$ $\delta(r_2, a) = (q, a, \downarrow)$ , se $a = \&$	<b>prepara-se para voltar volta ao início</b>

## 3. Memória em estados

Pode-se imaginar que os estados do controle finito armazenam uma quantidade finita de informação. Isso pode ajudar na compreensão do programa da máquina de Turing.

Exemplo: Máquina de Turing para deslocar a cadeia de entrada, uma posição para a direita.

**configuração inicial**

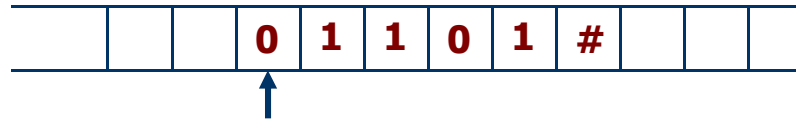
			0	1	1	0	1	#			
--	--	--	---	---	---	---	---	---	--	--	--

**configuração final**

				0	1	1	0	1	#		
--	--	--	--	---	---	---	---	---	---	--	--

# Máquinas de Turing

configuração inicial



configuração final

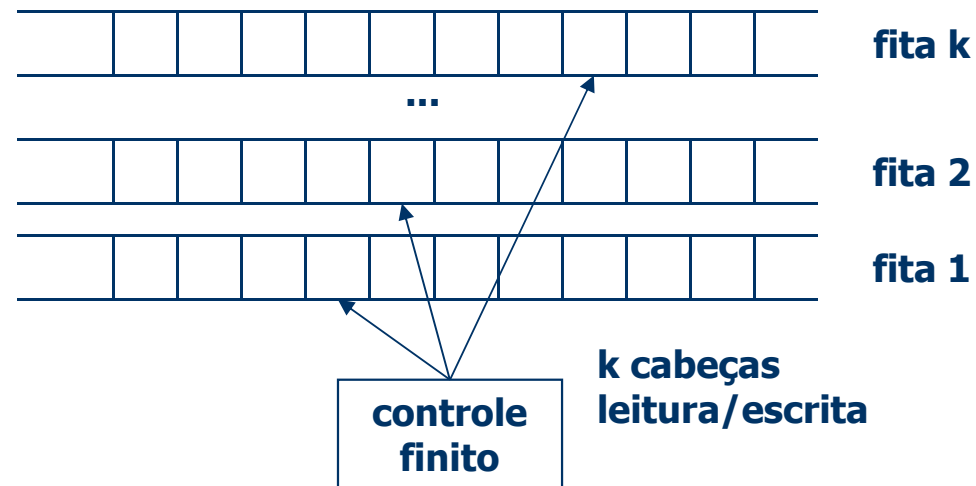


O programa para essa máquina pode ser:

$$\delta([q], x) = ([qx], B, \rightarrow) \quad x \in \{0, 1\}$$

$$\delta([qx], y) = ([qy], x, \rightarrow) \quad y \in \{0, 1, \#\}$$

## Máquina de Turing com várias fitas

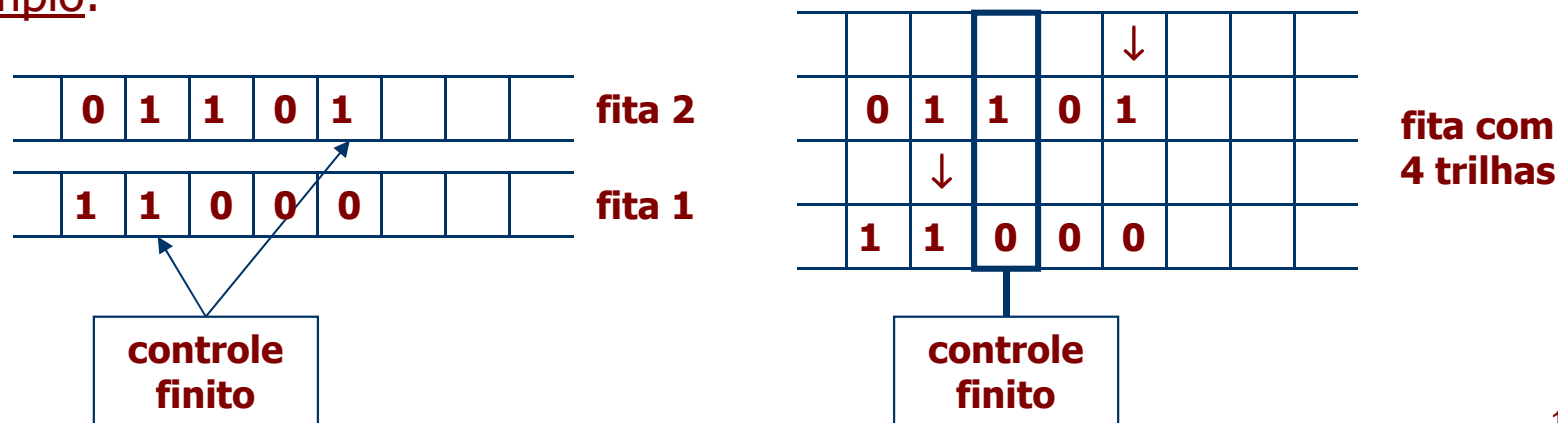




# Máquinas de Turing

- Num único movimento, a máquina de Turing com k-fitas, dependendo do estado do controle finito e dos k símbolos lidos pelas cabeças de leitura/escrita:
  - troca de estado
  - escreve um novo símbolo em cada uma das posições das cabeças
  - move cada uma das cabeças, independentemente, uma posição para a direita ou uma posição para a esquerda ou a mantém estacionária.
- Teorema: Seja L uma linguagem. Se L é aceita por uma máquina de Turing com k-fitas, então existe uma máquina de Turing simples M tal que  $L = L(M)$ .
- Prova: Seja  $M_k$  a máquina de Turing com k-fitas que aceita L. Construir a máquina de Turing simples M com uma fita de 2k trilhas. Cada fita de  $M_k$  irá corresponder a um par de trilhas em M: uma das trilhas armazena o conteúdo da fita de  $M_k$  e a outra trilha é deixada toda em branco exceto por um marcador que armazena a posição da cabeça correspondente de  $M_k$ .

Exemplo:



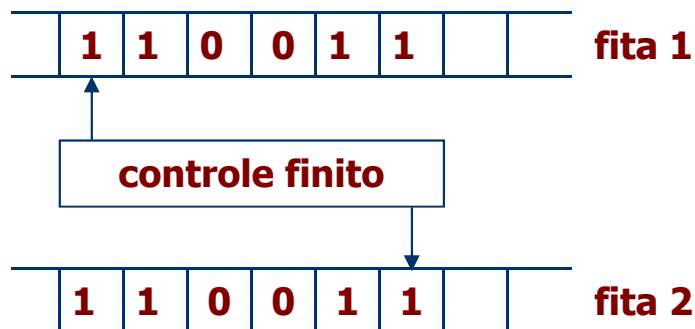
# Máquinas de Turing

---

- ♦ Um movimento de  $M_k$  é simulado por  $M$  da seguinte maneira (considere que a cabeça de leitura/escrita de  $M$  ao iniciar a simulação de um movimento de  $M_k$  encontra-se sobre o símbolo mais à esquerda de sua fita):
  - a)  $M$  desloca sua cabeça para a direita até encontrar os  $k$  marcadores  $\downarrow$ . Desse modo,  $M$  descobre os  $k$  símbolos lidos por  $M_k$  neste passo e os armazena em estado.
  - b) uma vez descobertos esses símbolos,  $M$  desloca a cabeça para a esquerda escrevendo os símbolos que  $M_k$  escreveria nesse passo, nas posições correspondentes.
  - c) deslocando sua cabeça para a direita,  $M$  atualiza os marcadores  $\downarrow$  que correspondem às cabeças de  $M_k$  que se deslocam para a direita neste passo.
  - d) finalmente, deslocando sua cabeça para a esquerda,  $M$  atualiza os marcadores  $\downarrow$  que correspondem às cabeças de  $M_k$  que se deslocam para a esquerda neste passo, posicionando também sua cabeça para simular o próximo passo de  $M_k$  e refletindo no estado do controle finito a mudança de estado de  $M_k$ .
- ♦ Exemplo: MT que aceita a linguagem  $L = \{ ww^R \mid w \in (0+1)^* \}$   
Podemos construir facilmente um reconhecedor para  $L$ , considerando-se uma MT com 2 fitas: (a) a entrada é colocada em ambas as fitas; (b) as duas fitas são lidas em sentidos contrários; (c) o comprimento da entrada é verificado ser par.

# Máquinas de Turing

- Seja, por exemplo,  $w = 110011 \in L$ .



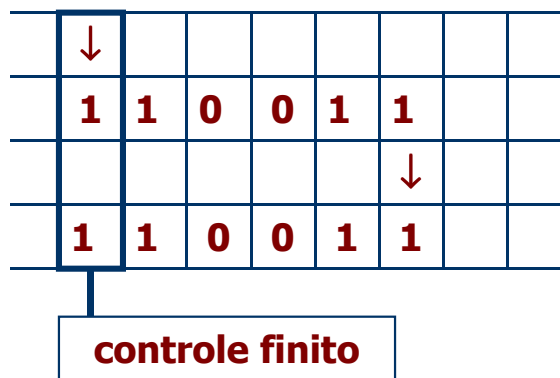
Esta máquina pode ser definida por:

$$\delta(q_0, a, a) = (q_1, a, \rightarrow, a, \leftarrow)$$

$$\delta(q_1, a, a) = (q_0, a, \rightarrow, a, \leftarrow)$$

$$\delta(q_0, b, b) = \text{aceita}$$

- Para simular essa MT podemos construir uma MTS com uma fita de 4 trilhas:



# Máquinas de Turing

- Seja a MTS definida por  $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$  onde:  
 $Q = \{ q_0, q_1, q_2, q_3, [0], [1], [2], [3] \}$ ;  $\Sigma = \{ 0, 1 \}$ ;  $F = \{ q_0 \}$   
 $\Gamma =$  conjunto de quádruplas do tipo  $(m_1 a_1 m_2 a_2)$ ;  $m_1, m_2 \in \{ \square, \downarrow \}$ ;  $a_1, a_2 \in \Sigma$   
(o símbolo  $\square$  representa uma célula em branco)  
e  $\delta$  dada por ( considere que  $x, y \in \Sigma$ ;  $\beta \in \{ \square, \downarrow \}$  ):

<b>1</b>	$\delta(q_0, (\square x \square y)) = (q_0, (\square x \square y), \rightarrow)$ $\delta(q_0, (\downarrow x \square y)) = ([x], (\downarrow x \square y), \rightarrow)$ $\delta([x], (\square x \square y)) = ([x], (\square x \square y), \rightarrow)$ $\delta([x], (\square y \downarrow x)) = (q_1, (\square y \downarrow x), \downarrow)$
<b>2</b>	$\delta(q_1, \alpha) = (q_1, \alpha, \leftarrow) \quad \alpha \neq (\square \square \square \square)$ $\delta(q_1, (\square \square \square \square)) = (q_2, (\square \square \square \square), \rightarrow)$
<b>3</b>	$\delta(q_2, (\square x \beta y)) = (q_2, (\square x \beta y), \rightarrow)$ $\delta(q_2, (\downarrow x \square y)) = ([2], (\square x \square y), \rightarrow)$ $\delta([2], (\square x \beta y)) = (q_2, (\downarrow x \beta y), \rightarrow)$ $\delta(q_2, (\square \square \square \square)) = (q_3, (\square \square \square \square), \leftarrow)$
<b>4</b>	$\delta(q_3, (\square x \square y)) = (q_3, (\square x \square y), \leftarrow)$ $\delta(q_3, (\beta x \downarrow y)) = ([3], (\beta x \square y), \leftarrow)$ $\delta([3], (\square x \square y)) = (q_3, (\square x \downarrow y), \leftarrow)$ $\delta(q_3, (\square \square \square \square)) = (q_0, (\square \square \square \square), \rightarrow)$

# Máquinas de Turing

- ♦ Seja  $w = 11$

$$\begin{aligned}
 & (q_0, [(\downarrow 1 \square 1)] (\square 1 \downarrow 1)) \mapsto ([1], (\downarrow 1 \square 1) [(\square 1 \downarrow 1)]) \\
 & \mapsto (q_1, (\downarrow 1 \square 1) [(\square 1 \downarrow 1)]) \mapsto^3 (q_2, [(\downarrow 1 \square 1)] (\square 1 \downarrow 1)) \\
 & \mapsto ([2], (\square 1 \square 1) [(\square 1 \downarrow 1)]) \mapsto (q_2, (\square 1 \square 1) (\downarrow 1 \downarrow 1) [ ]) \\
 & \mapsto (q_3, (\square 1 \square 1) [(\downarrow 1 \downarrow 1)]) \mapsto ([3], [(\square 1 \square 1)] (\downarrow 1 \square 1)) \\
 & \mapsto (q_3, [ ] (\square 1 \downarrow 1) (\downarrow 1 \square 1)) \mapsto (q_0, [(\square 1 \downarrow 1)] (\downarrow 1 \square 1))
 \end{aligned}$$

- ♦ Nesse caso, como não existe transição definida para esta configuração e  $q_0 \in F$ , então  $w = 11 \in L$ .

- ♦ Seja  $w = 101$

$$\begin{aligned}
 & (q_0, [(\downarrow 1 \square 1)] (\square 0 \square 0) (\square 1 \downarrow 1)) \mapsto^2 ([1], (\downarrow 1 \square 1) (\square 0 \square 0) [(\square 1 \downarrow 1)]) \\
 & \mapsto (q_1, (\downarrow 1 \square 1) (\square 0 \square 0) [(\square 1 \downarrow 1)]) \mapsto^4 (q_2, [(\downarrow 1 \square 1)] (\square 0 \square 0) (\square 1 \downarrow 1)) \\
 & \mapsto ([2], (\square 1 \square 1) [(\square 0 \square 0)] (\square 1 \downarrow 1)) \mapsto (q_2, (\square 1 \square 1) (\downarrow 0 \square 0) [(\square 1 \downarrow 1)]) \\
 & \mapsto (q_3, (\square 1 \square 1) (\downarrow 0 \square 0) [(\square 1 \downarrow 1)]) \mapsto ([3], (\square 1 \square 1) [(\downarrow 0 \square 0)] (\square 1 \square 1))
 \end{aligned}$$

- ♦ Nesse caso, como não existe transição definida para esta configuração e  $[3] \notin F$ , então  $w = 101 \notin L$ .

# Máquinas de Turing

---

## Tese de Church-Turing

“Qualquer procedimento efetivo pode ser realizado por uma máquina de Turing”

- ♦ A noção de procedimento efetivo é informal e assim, a tese de Church-Turing afirma a equivalência entre um conceito informal e um conceito formal, não sendo portanto passível de demonstração.
  - ♦ Existe, entretanto, considerável evidência que sustenta essa proposição. Por exemplo: diversas tentativas já foram feitas para definir e caracterizar a classe dos procedimentos efetivos, incluindo:
    - a máquina de Turing,
    - as funções recursivas gerais,
    - o  $\lambda$ -cálculo de Church,
    - os sistemas de produção de Post,
    - o algoritmo de Markov,
    - os predicados de Smullyan,
    - as funções  $\mu$ -recursivas de Gödel, Herbrand e Kleene
- Já foi mostrado, sem exceção, que essas formulações, apesar de pouco similares, são equivalentes duas a duas.
- ♦ O fato de tal variedade de formulações definir a mesma classe de funções computáveis é uma evidência muito forte para a generalidade de qualquer uma delas.

**Para mais detalhes sobre a tese de Church-Turing e os vários formalismos propostos para caracterizar a classe de funções computáveis, ver:**

**JONES, N.D. “Computability theory - An introduction”**

# Máquinas de Turing

## Máquina de Turing não-determinística

- ♦ Definição: Uma MTND é uma 6-tupla  $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$  onde  $Q, \Sigma, \Gamma, q_0$  e  $F$  são definidos como no caso da MTS e  $\delta : Q \times \Gamma \rightarrow P(Q \times \Gamma \times \{ \leftarrow, \downarrow, \rightarrow \})$  tal que:

$$(q, \alpha[a]\beta) \mapsto (p, \alpha[b]\beta) \Leftrightarrow (p, b, \downarrow) \in \delta(q, a)$$

$$(q, \alpha[a]c\beta) \mapsto (p, \alpha b[c]\beta) \Leftrightarrow (p, b, \rightarrow) \in \delta(q, a)$$

$$(q, \alpha c[a]\beta) \mapsto (p, \alpha[c]b\beta) \Leftrightarrow (p, b, \leftarrow) \in \delta(q, a)$$

onde  $p, q \in Q$ ;  $\alpha, \beta \in \Gamma^*$ ;  $a, b, c \in \Gamma$ .

Além disso, se  $\delta(q, a) = \emptyset$  então  $(q, \alpha[a]\beta)$  é configuração terminal. Se  $q \in F$  então a configuração terminal é configuração final.

- ♦ Teorema: Se  $L$  é uma linguagem reconhecida por uma MTND  $M$ , então existe uma MTS  $M'$  tal que  $L = L(M')$ .
- ♦ Prova: Seja  $C$  uma lista de configurações da MTND  $M$ . Para uma entrada  $w$  qualquer, a máquina  $M'$  simula  $M$  da seguinte maneira:
  - a) fazer  $C = ( (q_0, [a]w') )$ , onde  $q_0$  é o estado inicial de  $M$  e  $w = aw'$ .
  - b) enquanto  $C$  não contém uma configuração final, fazer:
    - i. para toda configuração  $c \in C$ , substituí-la por todas as configurações  $c'$  tais que  $c \mapsto^M c'$ .
    - ii. se não existe  $c'$  tal que  $c \mapsto^M c'$ , então retirar  $c$  de  $C$ .
    - iii. se  $C = \emptyset$ , então parar rejeitando  $w$ .
  - c) parar aceitando  $w$ .

# Máquinas de Turing

- ♦ Obviamente, se  $w \in L(M)$  então  $C$  irá conter uma configuração final e portanto  $w \in L(M')$ . Contudo, se  $w \notin L(M)$ ,  $M$  pode não parar, o mesmo acontecendo com  $M'$ .

## Máquina de Turing Universal

- ♦ Seja uma máquina de Turing  $M_1 = (Q, \Sigma, \Gamma, \delta, q_0, F)$  tal que  $Q = \{ 1, \dots, n \}$ ,  $\Sigma = \{ 0, 1 \}$  e  $\Gamma = \{ B, 0, 1 \}$ .  $M_1$  pode ser completamente especificada por uma tabela.

- ♦ **Exemplo:**

	B	0	1
1	-	-	(2, 0, $\rightarrow$ )
2	(3, 1, $\leftarrow$ )	(3, 1, $\leftarrow$ )	(2, 1, $\rightarrow$ )
3	(4, 0, $\rightarrow$ )	(4, 0, $\rightarrow$ )	(3, 1, $\leftarrow$ )
4	-	-	-

- ♦ Uma tabela como essa, entretanto, pode ser codificada da seguinte maneira:
  - os estados podem ser codificados como 1, 11, 111, ...
  - para cada estado pode-se construir um bloco (correspondente a uma linha da tabela) dividido em 3 sub-blocos (correspondente às colunas da tabela, ou seja, aos símbolos b, 0 e 1).



# Máquinas de Turing

## Codificação de um sub-bloco:

- ♦ se  $\delta(i, a) = (j, x, d)$  onde  $d \in \{ \leftarrow, \downarrow, \rightarrow \}$ , então o sub-bloco será codificado como:

$\underbrace{\$1 \dots 1}_{j \text{ vezes}} dx$

- ♦ se  $\delta(i, a) = \emptyset$ , então sua codificação será:  $\$0$
- ♦ Notar que os sub-blocos podem ser separados por  $\$$ ; os blocos podem ser separados por  $\$ \$$  e a codificação toda pode ser delimitada por  $\$ \$ \$$ .
- ♦ **Exemplo: a codificação da máquina de Turing especificada pela tabela:**

	B	0	1
1	-	-	(2, 0, $\rightarrow$ )
2	(3, 1, $\leftarrow$ )	(3, 1, $\leftarrow$ )	(2, 1, $\rightarrow$ )
3	(4, 0, $\rightarrow$ )	(4, 0, $\rightarrow$ )	(3, 1, $\leftarrow$ )
4	-	-	-

será:

$\$ \$ \$ 0 \$ 0 \$ 1 1 \rightarrow 0 \$ \$ 1 1 1 \leftarrow 1 \$ 1 1 1 \leftarrow 1 \$ 1 1 \rightarrow 1 \$ \$ 1 1 1 1 \rightarrow 0 \$ 1 1 1 1 \rightarrow 0 \$ 1 1 1 \leftarrow 1 \$ \$ 0 \$ 0 \$ 0 \$ \$ \$$

- ♦ Uma máquina de Turing universal (MTU) é uma máquina de Turing que quando apresentada a uma codificação de uma máquina de Turing M e a uma cadeia w, simula o comportamento de M com entrada w.

# Máquinas de Turing

---

Intuitivamente (em termos de linguagem de programação):

Podemos imaginar uma codificação para a MT como:

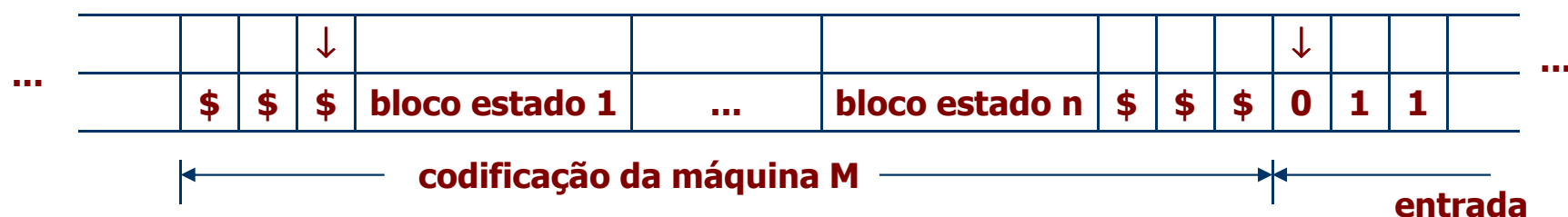
- ♦ existem 3 matrizes NOVOESTADO, ESCRIVE e MOVIMENTO para armazenar as informações da tabela que especifica uma MT qualquer.
- ♦ se  $\delta(i, a) = (j, x, d)$  onde  $d = \{-1, 0, 1\}$ , então:  
    NOVOESTADO[i, a] = j  
    ESCREVE[i, a] = x  
    MOVIMENTO[i, a] = d
- ♦ com essa codificação, o programa para a MTU pode ser:

```
estado := 1;  
cabeça := 1;  
símbolo := fita[cabeça];  
while transição_definida do  
begin  
    estado := novoestado[estado,símbolo];  
    fita[cabeça] := escreve[estado,símbolo];  
    cabeça := cabeça + movimento[estado,símbolo];  
end;
```

# Máquinas de Turing

- ♦ Mais formalmente: Usando a codificação com símbolos  $\$, 0, 1, \leftarrow, \downarrow, \rightarrow, B$  podemos definir a MTU como:
  - a) a MTU tem uma fita com 2 trilhas;
  - b) a trilha de baixo irá usar os símbolos  $\$, 0, 1, \leftarrow, \downarrow, \rightarrow, b$  e será usada para armazenar a codificação de uma MT  $M$  qualquer e também para armazenar a codificação de uma entrada qualquer de  $(0+1)^*$ ;
  - c) a trilha de cima irá usar os símbolos  $\downarrow$  e  $B$ . O símbolo  $\downarrow$  será usado para armazenar o estado atual e a posição da cabeça de  $M$ .

Inicialmente:



Funcionamento da MTU:

- 1) a MTU movimenta sua cabeça para a direita até encontrar o símbolo  $\downarrow$  sobre um símbolo  $x$  da entrada, armazenando-o em seu controle finito. A MTU movimenta então sua cabeça para a esquerda até encontrar o símbolo  $\downarrow$  que armazena o estado de  $M$ . A MTU então apaga esse símbolo e movimenta sua cabeça para a direita para o sub-bloco correspondente a  $x$  e escreve  $\downarrow$  sobre o primeiro símbolo desse sub-bloco (que deve ser 1; do contrário a MTU pára, pois não existe próximo movimento de  $M$ ). Seja  $m_1$  esse último marcador.

# Máquinas de Turing

---

- 2) a MTU movimenta agora sua cabeça para a esquerda até encontrar \$\$\$, marcando com  $\downarrow$  o \$ mais à direita. Seja  $m_2$  esse novo marcador.
  - 3) a MTU movimenta então sua cabeça para a direita até encontrar o marcador  $m_1$  e chama uma subrotina que move, alternadamente,  $m_1$ , uma posição para a direita e  $m_2$ , um bloco para a direita. Quando  $m_1$  apontar para um símbolo diferente de 1,  $m_2$  estará localizado sobre o \$ imediatamente anterior ao bloco correspondente ao próximo estado de M. Nesse ponto, a MTU apaga  $m_1$  e armazena em seu controle finito o símbolo que M irá escrever e o sentido do movimento da cabeça de M. Em seguida, a MTU movimenta sua cabeça para a direita até encontrar o marcador  $\downarrow$  sobre a entrada. O símbolo sob esse marcador é então trocado e  $\downarrow$  é movimentado segundo o sentido armazenado. Neste ponto a MTU simulou um movimento de M. Em seguida, o processo é repetido.
- ♦ Se M pára com entrada w qualquer, a MTU irá parar também e a parte da fita que inicialmente contém w, ao final, vai estar como a fita de M. Quando M pára, a MTU pode descobrir se M está num estado final ou não, indo para seu estado final ou não, conforme o caso.
  - ♦ Se M não parar, então a MTU também não vai parar.
  - ♦ Portanto, a MTU simula M.

# Máquinas de Turing

---

## O problema da parada de máquinas de Turing

“Seja uma MT  $M$  e uma entrada  $w$  qualquer.  $M$  pára com entrada  $w$ ?”

- Vamos mostrar que esse problema é indecidível, ou seja, que não existe um algoritmo que responda “sim” ou “não” para todo par  $(M, w)$  (note que isso não significa que não se pode determinar se uma máquina de Turing específica sob uma entrada específica irá parar ou não).

## Enumeração de máquinas de Turing

- Como mostramos anteriormente, uma MT pode ser codificada como uma cadeia de  $\Omega^*$ , onde  $\Omega = \{ \$, 0, 1, \leftarrow, \downarrow, \rightarrow \}$ . Podemos escolher uma ordenação dos símbolos de  $\Omega$  e então enumerar as cadeias de  $\Omega^*$ . Considerando que cada uma dessas cadeias é a codificação de uma MT (eventualmente existirão cadeias mal formadas e que podem ser consideradas como codificações de MT que param com zero movimentos), faz sentido falar na  $i$ -ésima máquina de Turing  $T_i$  ( $i > 0$ ).
- Da mesma maneira as cadeias de  $\{0, 1\}^*$  podem ser ordenadas (por exemplo:  $\Lambda, 0, 1, 00, 01, 10, 11, 000, \dots$ ) e portanto, também faz sentido falar da  $j$ -ésima cadeia de  $\{0, 1\}^*$ .

# Máquinas de Turing

---

- ♦ Seja, então, a linguagem:  $L = \{ x_i \in (0+1)^* \mid x_i \text{ não é aceita por } T_i \} \quad (i > 0)$
- ♦ Vamos mostrar que não existe MT que aceita  $L$ .  
Vamos supor, por absurdo, que  $T_j$  ( $j > 0$ ) é uma MT que aceita  $L$ .  
Então:  $x_j \in L \Leftrightarrow x_j \text{ não é aceita por } T_j$   
Mas, se  $T_j$  aceita  $L$ , então:  
 $x_j \in L \Leftrightarrow x_j \text{ é aceita por } T_j$ . CONTRADIÇÃO!  
Portanto, não existe uma MT que aceita  $L$ .
- ♦ Teorema: Não existe um algoritmo (ou seja, uma MT que sempre pára) que resolve o problema da parada de máquinas de Turing.
- ♦ Prova (informal):  
Considere, por absurdo, que existe um algoritmo  $A$  que determina se uma MT qualquer, com entrada  $w$ , vai parar ou não.  
Seja  $M$  uma máquina de Turing. Temos então:
  - 1) Dada uma sentença  $x$ ,  $M$  enumera as sentenças de  $\{0, 1\}^*$  até encontrar um inteiro  $i$  tal que  $x = x_i$ .
  - 2) Em seguida,  $M$  gera a codificação da máquina  $M_i$ , por enumeração das cadeias de  $\{ \$, 0, 1, \leftarrow, \downarrow, \rightarrow \}^*$ .
  - 3) Aplicar o algoritmo  $A$  para determinar se  $M_i$  com entrada  $x_i$  pára ou não.

# Máquinas de Turing

---

- ♦ Se  $A$  determina que  $M_i$  não pára com entrada  $x_i$ , então  $M$  pára e aceita  $x_i$ ;
- ♦ Se  $A$  determina que  $M_i$  pára com entrada  $x_i$ , então transferir o controle para uma MTU que simula  $M_i$  com entrada  $x_i$ . Como  $M_i$  vai parar, a MTU pára e determina se  $M_i$  aceita  $x_i$  ou não:
  - Se  $M_i$  aceitar  $x_i$ , então  $M$  pára e não aceita  $x_i$ .
  - Se  $M_i$  não aceita  $x_i$ , então  $M$  pára e aceita  $x_i$ .

Portanto,  $M$  aceita  $L$ , o que é um ABSURDO!

Logo, o algoritmo  $A$  não existe.

## Linguagens recursivas e linguagens recursivamente enumeráveis

- ♦ Seja  $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$  uma MT. Sejam  $A(M)$ , o conjunto das cadeias aceitas por  $M$ , e  $R(M)$ , o conjunto das cadeias rejeitadas por  $M$ , definidos como:

$$A(M) = \{ x \in \Sigma^* \mid (q_0, [x]) \mapsto^* (q, [\alpha]); q \in F \}$$

$$R(M) = \{ x \in \Sigma^* \mid (q_0, [x]) \mapsto^* (q, \alpha[a]\beta) \text{ com } q \notin F \text{ e } \delta(q, a) = \emptyset \}$$

Podemos definir também:

$$C(M) = \Sigma^* - (A(M) \cup R(M))$$

ou seja, o conjunto das cadeias de  $\Sigma^*$  que levam  $M$  a não parar.

# Máquinas de Turing

---

- ♦ Definição:  $L$  é recursiva  $\Leftrightarrow$  existe máquina de Turing  $M$  tal que  $A(M) = L$  e  $R(M) = \Sigma^* - L$  (ou seja,  $M$  sempre pára).
- ♦ Definição:  $L$  é recursivamente enumerável  $\Leftrightarrow$  existe máquina de Turing  $M$  tal que  $A(M) = L$  (ou seja, se  $x \notin L$ , a máquina  $M$  com entrada  $x$  pode não parar).
- ♦ Lema: Se  $L$  é uma linguagem recursiva, então  $\sim L = \Sigma^* - L$  é recursiva.
- ♦ Prova:  
 $L$  é recursiva. Seja  $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$  tal que  $A(M) = L$  e  $R(M) = \sim L$ .  
Construir  $M' = (Q \cup \{q\}, \Sigma, \Gamma, \delta', q_0, \{q\})$  onde  $q$  é um novo estado e  $\delta'$  é dada por:
  - a) se  $\delta(p, a) = (p', x, d)$  então  $\delta'(p, a) = (p', x, d)$   
com  $p, p' \in Q$ ;  $a, x \in \Gamma$ ;  $d \in \{ \leftarrow, \downarrow, \rightarrow \}$
  - b) se  $\delta(p, a) = \emptyset$ , com  $p \notin F$ , então  $\delta'(p, a) = (q, a, \downarrow)$
  - c)  $\delta'(q, a) = \emptyset, \forall a \in \Gamma$

Logo:

$$A(M') = R(M) = \Sigma^* - L = \sim L$$

$$R(M') = A(M) = L = \Sigma^* - \sim L$$

Portanto,  $\sim L$  é recursiva.



# Máquinas de Turing

---

- ♦ Lema: Seja  $x_1, x_2, \dots$  uma enumeração das cadeias de  $\Sigma^*$  e  $T_1, T_2, \dots$  uma enumeração das máquinas de Turing sobre  $\Sigma$ . Seja  $L = \{ x_i \in \Sigma^* \mid x_i \in A(T_i) \}$ . Então:
  - a)  $L$  é recursivamente enumerável
  - b)  $\sim L$  não é recursivamente enumerável
- ♦ Prova: Exercício!
- ♦ Teorema: A classe das linguagens recursivas é um subconjunto próprio da classe das linguagens recursivamente enumeráveis.
- ♦ Prova: Consequência dos lemas anteriores.

## Máquinas de Turing e gramáticas tipo-0

- ♦ Proposição: A linguagem  $L$  é reconhecida por uma MT  $\Leftrightarrow L$  é gerada por uma gramática tipo-0.

O teorema a seguir estabelece metade da prova dessa proposição.

- ♦ Teorema: Se  $L$  é gerada por uma gramática tipo-0, então  $L$  é reconhecida por uma MT.
- ♦ Prova: Seja  $G = (N, \Sigma, P, S)$  uma gramática tipo-0 tal que  $L = L(G)$ . Construir uma MT  $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$  tal que:  
 $\Gamma = N \cup \Sigma \cup \{ B, \#, X \}$  onde  $B, \#, X \notin N \cup \Sigma$

# Máquinas de Turing

Inicialmente  $M$  tem uma entrada  $w \in \Sigma^*$  na fita.  $M$  então insere  $\#$  antes de  $w$  e  $\#S\#$  depois. Logo, o conteúdo da fita após esse passo será:

	<b>#</b>	<b>w</b>	<b>#</b>	<b>S</b>	<b>#</b>	
--	----------	----------	----------	----------	----------	--

Em seguida,  $M$  irá (não deterministicamente) simular derivações de  $G$  começando com  $S$ , substituindo os símbolos que aparecerem entre os dois últimos  $\#$  usando todas as possíveis produções de  $P$ .

Seja  $\#w\#A_1A_2\dots A_k\#$  o conteúdo da fita de  $M$  num dado passo.  $M$  movimenta sua cabeça por  $A_1A_2\dots A_k$ , escolhendo não-deterministicamente todas as possíveis subcadeias  $A_i\dots A_{i+m}$  que são lados esquerdos de produções de  $P$ , substituindo essas subcadeias pelos correspondentes lados direitos das produções (nessa substituição  $M$  pode deslocar  $A_{i+m+1}\dots A_k\#$  para a esquerda ou para a direita a fim de preencher ou conseguir espaço, caso o lado direito da produção usada tiver comprimento diferente do lado esquerdo dessa produção).

Com essa simulação de derivações de  $G$ ,  $M$  irá escrever uma cadeia  $\#w\#\alpha\#$  na sua fita, exatamente se  $S \Rightarrow^* \alpha$ . Caso  $\alpha = w$  ( $M$  pode comparar as cadeias  $w$  e  $\alpha$  a cada passo para testar essa condição),  $M$  aceita  $w$ .

Assim, se  $S \Rightarrow^* w$  então  $M$  aceita  $w$ .

Logo: se  $L = L(G)$  então  $L = A(M)$ .

# Máquinas de Turing

- Exemplo: Seja  $G = (\{A, B\}, \{0, 1\}, P, A)$  com  $P$  dado por:

$$A \rightarrow B0$$
$$A0 \rightarrow B1$$
$$A \rightarrow 0$$
$$B \rightarrow BB$$
$$B \rightarrow 1A$$
$$B \rightarrow 1$$

Considere duas subrotinas para deslocamento da cadeia da fita uma posição para a direita e uma posição para a esquerda, com estados inicial e de retorno, respectivamente,  $[di]$ ,  $[df]$  e  $[ei]$ ,  $[ef]$ .

Um trecho do programa da MT que corresponde às ações que devem ser tomadas quando é encontrada uma subcadeia  $A$  (isto é, quando se tem  $\#w\#...A...\#$  na fita) é dado por:

$$\delta(q_1, A) = \{ ([di], Y, \rightarrow),$$
$$(q_2, A, \rightarrow),$$
$$(q_3, 0, \leftarrow) \}$$
$$\delta([df], Y) = (q_4, B, \rightarrow)$$
$$\delta(q_4, X) = (q_3, 0, \leftarrow)$$
$$\delta(q_2, 0) = (q_5, 1, \leftarrow)$$
$$\delta(q_5, A) = (q_3, B, \leftarrow)$$

corresponde a  $A \rightarrow B0$

corresponde a  $A0 \rightarrow B1$

corresponde a  $A \rightarrow 0$

onde é utilizado um símbolo adicional  $Y$  para indicar chamadas a subrotinas (a subrotina `desloca_para_direita` utiliza um símbolo adicional  $X$  para marcar o espaço criado).