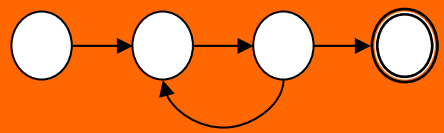
Automata e Linguagens Formais

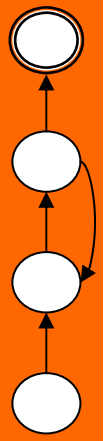
CTC 34



8

Prof. Carlos H. C. Ribeiro
carlos@ita.br

Máquinas de Turing
MTs e Funções Inteiras
Linguagens recursivas
Artifícios para o projeto de MTs
Variações de MTs
Máquinas equivalentes a MTs



A Máquina de Turing

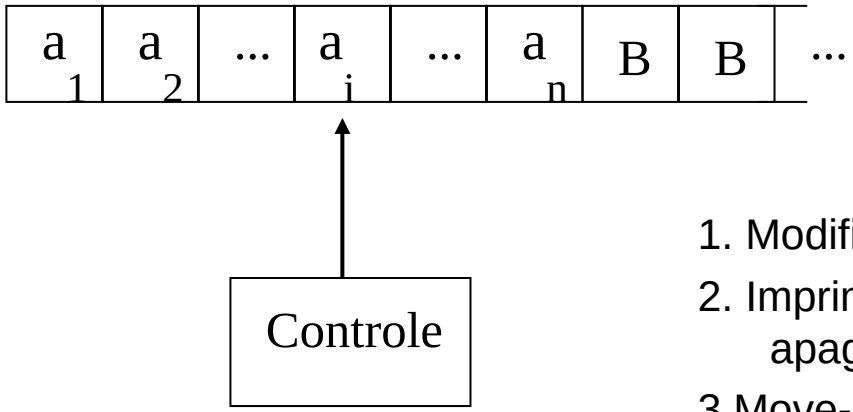
- Proposta pelo inglês Alan M. Turing, em 1936.
- Baseada em sequências de operações elementares.
- Não apresenta limitações de memória ou tempo.
- Várias possíveis realizações.
- Realização computacional de um **procedimento**.
- É a máquina computacional abstrata mais poderosa. Isto não significa ser capaz de computar qualquer função...

Alan Mathison Turing
(23/06/1912 – 07/06/1954)

Matemático, lógico, criptógrafo e herói de guerra britânico. Considerado o pai da ciência da computação. Fez previsões acerca da Inteligência Artificial e propôs o Teste de Turing, contribuindo para o debate sobre a consciência das máquinas e suas capacidades de pensar. Formalizou o conceito de algoritmo e computação com a Máquina de Turing, gerando sua versão da Tese de Church-Turing. Responsável pela quebra do código alemão *Enigma* durante a II Guerra Mundial. Depois da guerra, projetou um pioneiro computador digital programável eletronicamente. Foi processado e condenado por ser homossexual (em 1952). Morreu envenenado (provável suicídio). O Turing Award foi criado em sua homenagem.

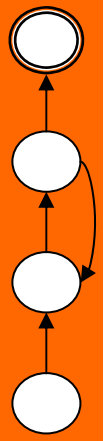
Um Modelo da Máquina de Turing

- Fita dividida em células, finita à esquerda, infinita à direita.
- Cabeça de leitura + controle da cabeça (registrador de estados).
- Cadeias escritas a partir da célula mais à esquerda.



1. Modifica-se o estado do controle
2. Imprime-se um símbolo na célula lida, apagando-se o que estava.
3. Move-se a cabeça para direita ou esquerda.

O poder computacional adicional da MT em relação a um 2-AF está em sua capacidade de escrever na fita.

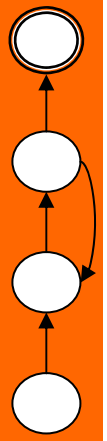


Máquina de Turing: Definições

- Uma Máquina de Turing (modelo básico) é denotada $M=(Q,\Sigma,\Gamma,\delta,q_0,B,F)$ onde:
  - Q é o conjunto finito de **estados**;
  - Γ é o alfabeto finito de **símbolos da fita**;
  - $B \in \Gamma$ é o **símbolo "em branco"** (*blank*);
  - $\Sigma \subset \Gamma$ - {B} é o conjunto de **símbolos de entrada**;
  - δ é a **função de transição** de $Q\times\Gamma$ em $Q\times\Gamma\times\{L,R\}$ (pode ser uma função não definida para alguns argumentos);
  - $q_0 \in Q$ é o **estado inicial**;
  - $F \subseteq Q$ é o **conjunto de estados finais**.

- Uma **configuração instantânea** de uma MT é um tripla $\alpha_1 q \alpha_2$ em que q é o estado e $\alpha_1\alpha_2$ é a string de Γ^* correspondente ao conteúdo da fita até o símbolo diferente de B mais à direita ou até o símbolo à esquerda da cabeça de leitura (o que estiver mais à direita). A cabeça está sobre o símbolo mais à esquerda de α_2 (se $\alpha_2 = \varepsilon$ a cabeça está sobre o símbolo B).

Máquina de Turing: Mais Definições

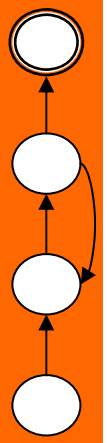


- Seja $X_1X_2 \ldots X_{i-1}\ q\ X_i \ldots X_n$ uma configuração instantânea. Assume-se que se i=n+1, entao X_i = B. Então:
  - Caso 1: **movimento para a esquerda** $\delta(q, X_i)=(p,Y,L)$.
    - Se i=1, não existe uma próxima C.I.
    - Se i>1, $X_1X_2 \ldots X_{i-1}\ q\ X_i \ldots X_n \vdash X_1X_2 \ldots X_{i-2}\ p\ X_{i-1}\ Y \ldots X_n$.
  - Caso 2: **movimento para a direita** $\delta(q, X_i)=(p,Y,R)$.
    - $X_1X_2 \ldots X_{i-1}\ q\ X_i \ldots X_n \vdash X_1X_2 \ldots X_{i-1}\ Y\ p\ X_{i+1} \ldots X_n$.

- A notação $\vdash^*$ indica que uma configuração instantânea pode ser obtida a partir de outra por zero ou mais movimentos da máquina M.

- **Def.** A **linguagem L(M)** aceita por uma MT M é o subconjunto de Σ^* que leva M a um estado final a partir da configuração inicial definida por:
  - Cadeia posicionada a partir da posição mais à esquerda da fita;
  - Cabeça de leitura posicionada sobre célula mais à esquerda da fita.
  - M no estado q_0 ;

  Formalmente: $L(M)=\{w \mid w \in \Sigma^* \text{ e } q_0w \vdash^* \alpha_1 p \alpha_2 \text{ para algum } p \in F,\ \alpha_1,\alpha_2 \in \Gamma^*\}$



MTs como Decisores

- Ao iniciar uma MT com uma entrada, pode-se:
  - Aceitar a entrada;
  - Rejeitar a entrada;
  - Não parar (MT em *loop*).
- Uma MT pode não aceitar uma cadeia de entrada: (i) ao entrar em $q \notin F$ e parar (ou seja, **rejeitar** a cadeia) ou (ii) ao entrar em *loop* (não parar).
  - Pode ser muito difícil distinguir uma MT que entrou em *loop* de outra que está demorando para computar.
- Uma MT **decide uma linguagem** quando pára para todas as entradas:
  - se $w \notin L(M)$, MT pára em $q \notin F$;
  - se $w \in L(M)$, MT pára em $q \in F$.
- A classe das linguagens aceitas por MTs decisoras é a classe das **linguagens recursivamente enumeráveis**.

Exemplo: MT para reconhecer $L=\{0^n1^n \mid n\geq 1\}$

- *Inicialmente na fita: 0^n1^n*
- *Operação: substitui 0 mais à esquerda por X, a seguir move-se para direita e substitui 1 mais à esquerda por Y. Move-se então para a esquerda, até se encontrar o X mais à direita. Move-se então uma célula para a direita (onde deve estar agora o 0 mais à esquerda) e repete-se o ciclo.*
- *Aceitação: a) Se ao se procurar um 1 encontra-se um B, realiza-se uma parada sem aceitação. b) Se, após mudar um 1 para Y, M não encontrar mais 0's, tenta-se achar 1's. Se não se encontra nenhum, ocorre aceitação.*

$Q=\{q_0,q_1,q_2,q_3,q_4\}$ $\Sigma=\{0,1\}$ $\Gamma=\{0,1,X,Y,B\}$ $F=\{q_4\}$

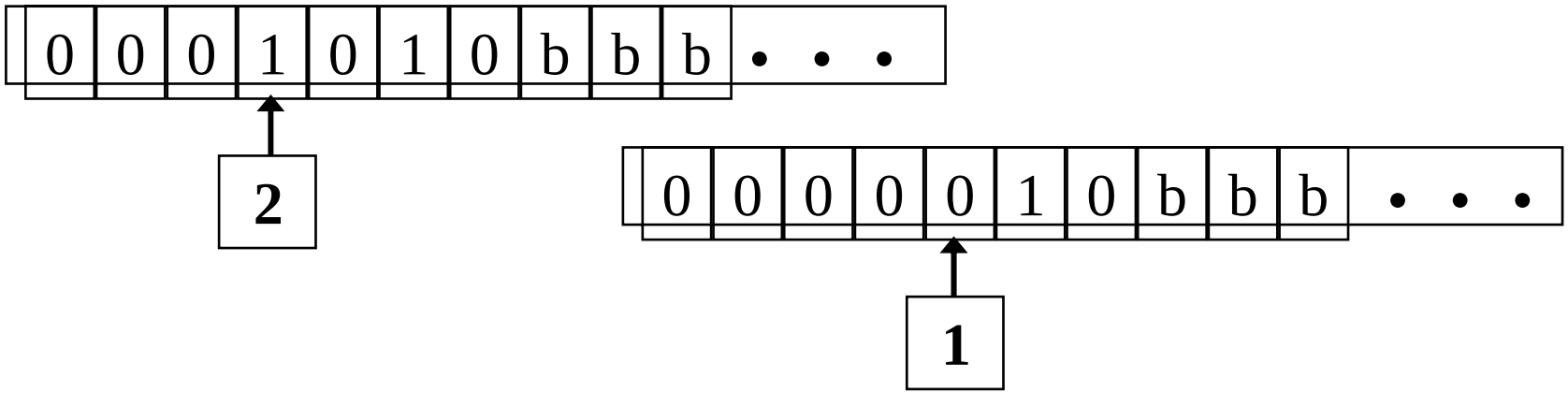
- q_0: Estado inicial; revisitado imediatamente antes da substituição do 0 mais à esquerda por um X. Ao ler um Y, muda para estado q_3.
- q_1: Usado para se buscar para a direita, ignorando-se 0's e Y's até se encontrar o 1 mais à esquerda. Ao se encontrar o 1, muda-se para Y e entra-se no estado q_2. Caso se encontre um B ou X antes do 1, a string é rejeitada.
- q_2: Usado para se fazer uma busca para a esquerda por um X, muda então para estado q_0 e move cabeça para direita, até se encontrar o 0 mais à esquerda.
- q_3: Varre sequencia de Y's, e checa se não existem 1's remanescentes. Se os Y's são seguidos por um B, entra-se estado q_4 e aceita, caso contrário rejeita.

MT para reconhecer $L=\{0^n1^n \mid n\geq 1\}$

	Símbolo				
Estado	**0**	**1**	**X**	**Y**	**B**
q_0	(q_1,X,R)	-	-	(q_3,Y,R)	-
q_1	$(q_1,0,R)$	(q_2,Y,L)	-	(q_1,Y,R)	-
q_2	$(q_2,0,L)$	-	(q_0,X,R)	(q_2,Y,L)	-
q_3	-	-	-	(q_3,Y,R)	(q_4,B,R)
q_4	-	-	-	-	-

Forma alternativa de descrever uma MT

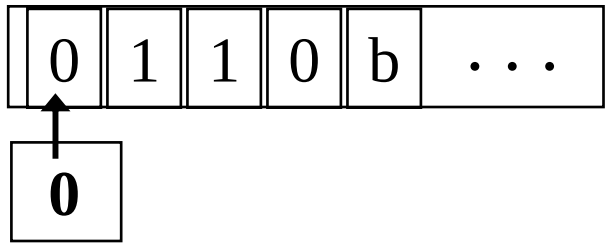
- Uma MT também pode ser descrita por um conjunto de ações.
- **Ações**: **(s, i, i', s', d)** sendo:
  - s: estado atual da MT (s $\in$ S)
  - i: símbolo que está sendo lido na fita (i $\in \Gamma$)
  - i': símbolo que é impresso na fita (i'$\in \Gamma$)
  - s': novo estado da MT (s'$\in$ S)
  - d$\in${D,E}.
- Exemplo: Execução da ação (2,1,0,1,D).



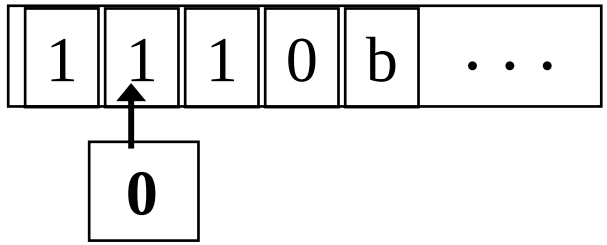
Ações da Máquina de Turing

- Ex: uma MT é definida pelo conjunto de quíntuplas: (0,0,1,0,D), (0,1,0,0,D), (0,b,1,1,E), (1,0,0,1,D), (1,1,0,1,D), (1,b,b,2,D). O estado de aceitação é 2. Verificar sua computação:

Início:

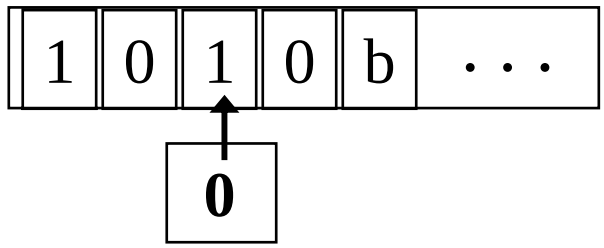


(0,0,1,0,D)

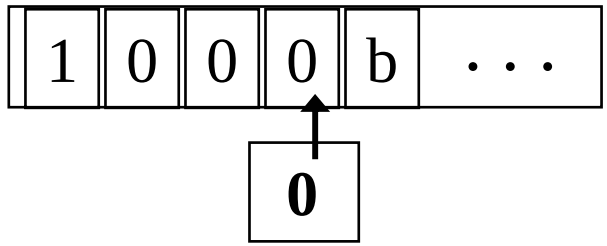


(0,1,0,0,D)

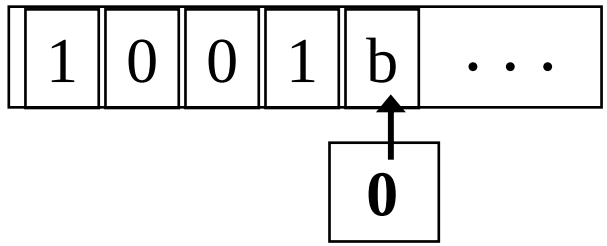
Ações da Máquina de Turing



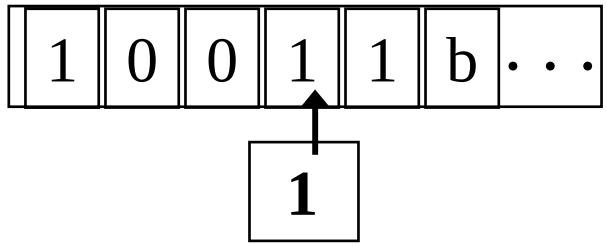
(0,1,0,0,D)



(0,0,1,0,D)

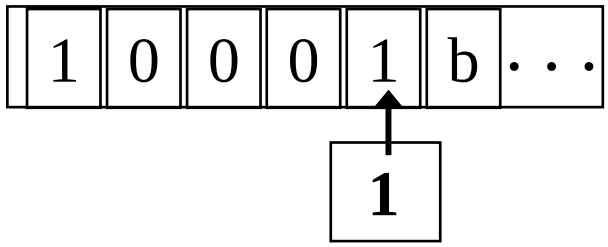


(0,b,1,1,E)

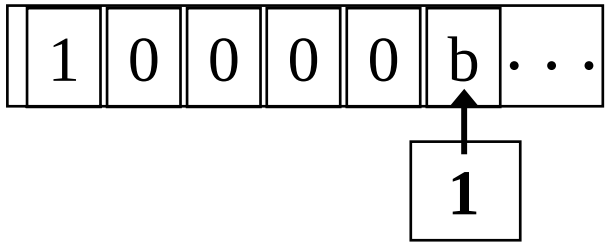


(1,1,0,1,D)

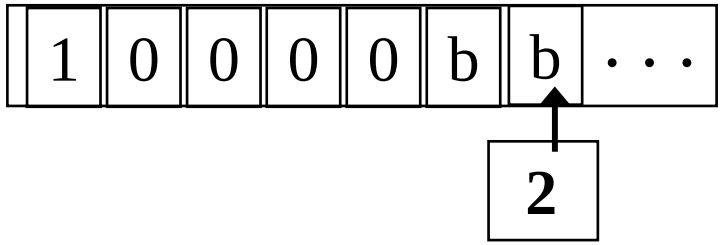
Ações da Máquina de Turing



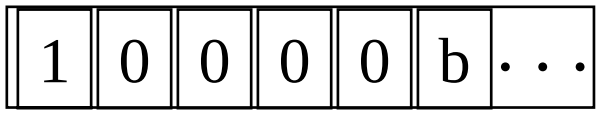
(1,1,0,1,D)



(1,b,b,2,D)

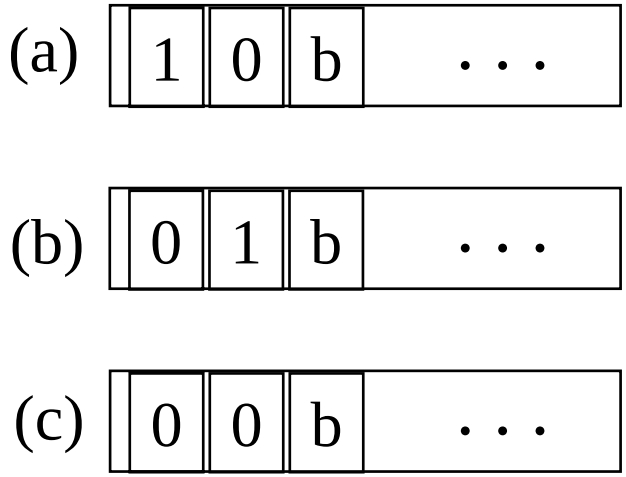


Como a máquina pára no estado 2 (estado de aceitação), a fita terá:



Ações da Máquina de Turing

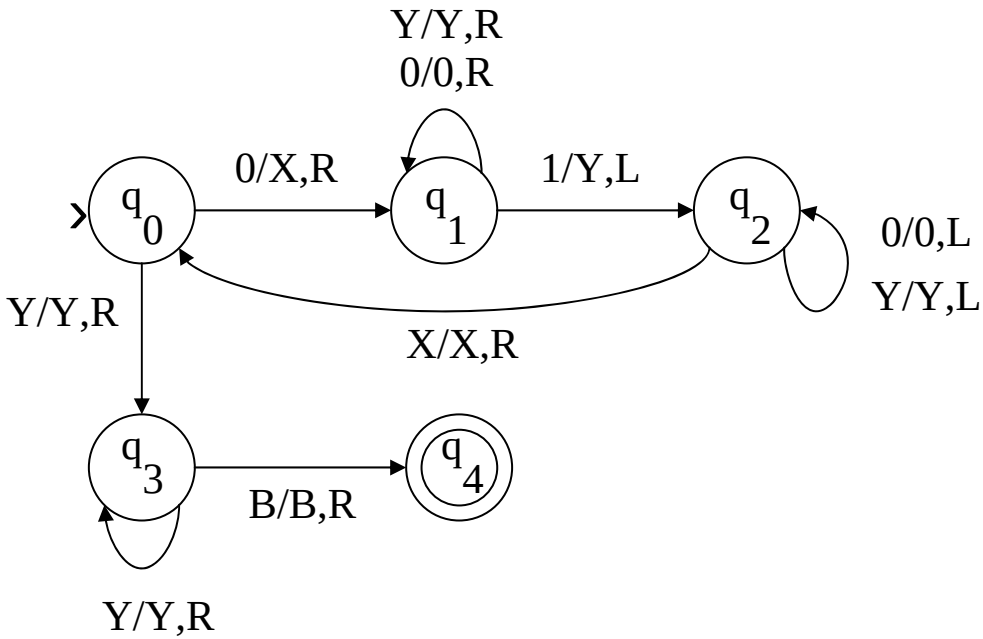
- Exercício: Considere a seguinte MT: (0,0,0,1,D), (0,1,0,0,D), (0,b,b,0,D), (1,0,1,0,D), (1,1,1,0,E). Qual o comportamento da MT quando iniciada com:



Diagramas de Transição para MTs

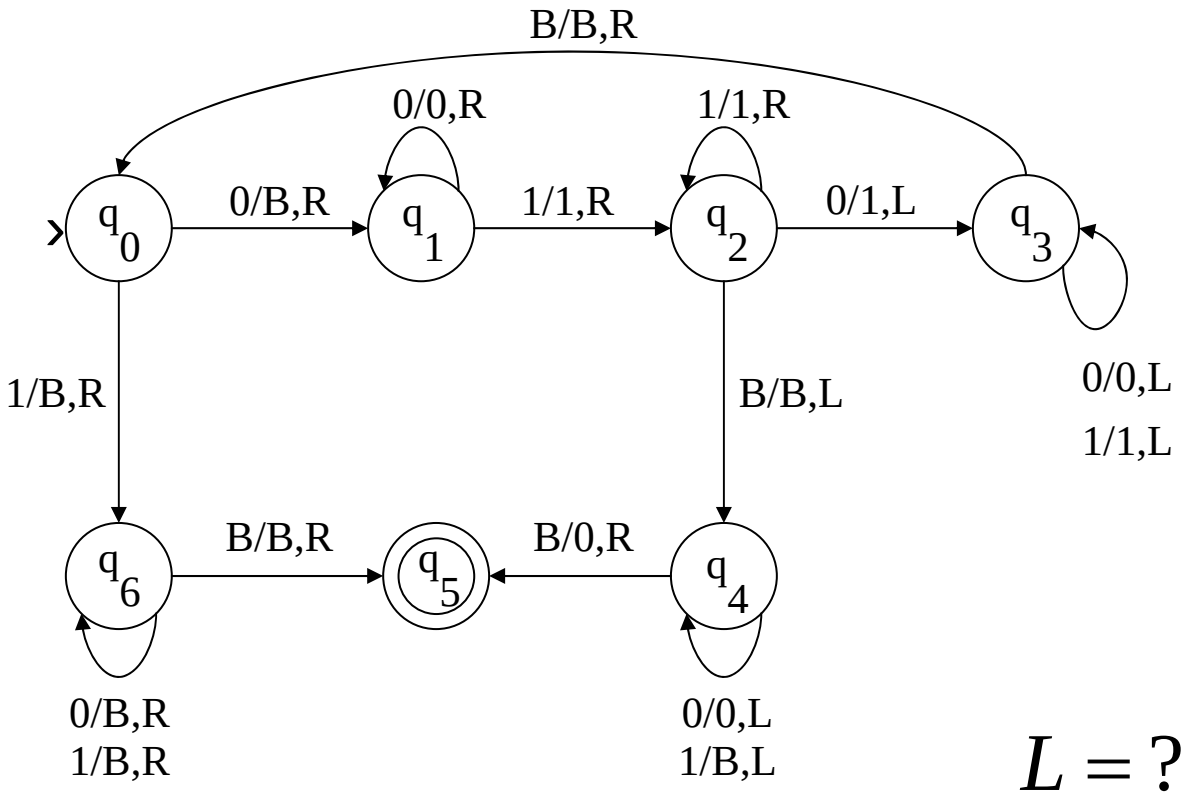
- Máquina de Turing $M=(Q,\Sigma,\Gamma,\delta,q_0,B,F)$:
  - Cada nó = um estado
  - Arco X/Y R ligando q a p $\leftrightarrow \delta(q,X)=(p,Y,R)$
  - Arco X/Y L ligando q a p $\leftrightarrow \delta(q,X)=(p,Y,L)$
  - Estado inicial e estados de aceitação: como em AFDs.

Exemplo 1:

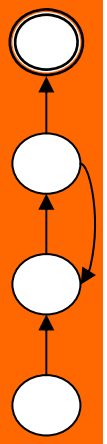


$$L = 0^n 1^n$$

Exemplo 2


B/B,R
0/0,R
1/1,R
q0
0/B,R
q1
1/1,R
q2
0/1,L
q3
1/B,R
B/B,L
0/0,L
1/1,L
B/B,R
B/0,R
q6
q5
q4
0/B,R
1/B,R
0/0,L
1/B,L

$L = ?$



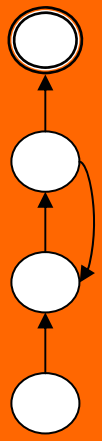
Computação de Funções Inteiras

Uma MT pode também ser vista como um computador para funções inteiras.

Representação:

- Inteiro $i \geq 0$: string 0^i
- Separação entre inteiros como argumentos de uma função:

  $(i_1,i_2,...,i_k) \rightarrow 0^{i}_{1},1, 0^{i}_{2},1,...,1, 0^{i}_{k}$
- Parada em uma fita com conteúdo 0^m: $f(i_1,i_2,...,i_k) = m$, onde f e a função calculada (computada) pela MT.
- **OBS1:** Uma mesma MT pode computar uma função de um argumento, uma outra função de dois argumentos, etc.
- **OBS2:** Domínio da função pode ser um **subconjunto** de $\mathbb{Z}^k$.



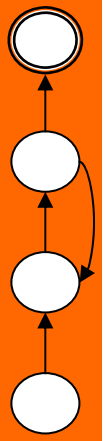
Computação de Funções Inteiras: Exemplo

Uma MT para realizar subtração de inteiros de acordo com a seguinte definição: m-n=m-n se m≥n, e m-n=0 se m<n.

Considere a MT $M=(\{q_0,q_1,q_2,q_3,q_4,q_5,q_6\},\{0,1\},\{0,1,B\},\delta,q_0,B,\varnothing)$, definida da seguinte maneira:

- Inicialmente na fita: 0^m10^n
- Computação: parada com 0^{m-n} na fita.
- Operação: substitui 0 inicial por B, a seguir inicia busca de 1 seguido de 0, substitui o 0 por 1. M entao volta para esquerda até encontrar um B, e recomeça ciclo.
- Fim da operação:
  - Ao procurar um 0 quando avançando a cabeca, M encontra um B. Nesse caso, os n 0's em 0^m10^n foram mudados para 1, e n+1 dos m 0's foram mudados para B. M entao substitui os n+1 1's por B's e um B por 0, deixando m-n 0's na fita.
  - Ao iniciar um ciclo, M não consegue encontrar um 0 a ser transformado em B, pois os primeiros m 0's já foram modificados. Neste caso, $n \geq m$ e portanto m-n = 0. M então substitui todos os 0's e 1's restantes por B's.

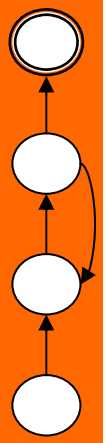
Um exemplo de computação.



Artifícios para o Projeto de MTs

Não existem “receitas” simples para o projeto de uma MT. Alguns artifícios, porém, podem ser úteis:

- Armazenamento da informação nos estados.
- Utilização de múltiplas fitas.
- Checagem de símbolos.
- Deslocamento de símbolos.
- Programação estruturada (utilização de subrotinas).



Armazenamento da Informação

- Estado pode ser escrito como um par de elementos:
  - Primeiro elemento: controle
  - Segundo elemento: informação

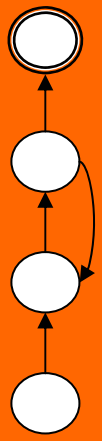
Não há modificação sobre o modelo básico: um par de elementos é um “estado”...

Exemplo: Uma MT que aceita a linguagem regular $(01^* + 10^*)$:

- Lê o primeiro símbolo da fita (0 ou 1).
- Registra o símbolo como parte do “estado”
- Verifica se o símbolo não aparece mais na fita.

A idéia é:

- Primeiro componente do “estado” controla avanço da cabeça.
- Segundo componente do “estado” lembra-se do primeiro símbolo lido.



$M=(Q, \{0,1\}, \{0,1,B\}, \delta, [q_0, B], B, F\}$

$Q=\{[q_0, 0], [q_0, 1], [q_0, B], [q_1, 0], [q_1, 1], [q_1, B]\}$

1) A partir do estado inicial ($[q_0, B]$): avança para estado seguinte e armazena símbolo lido como segundo componente do estado:

$$\delta([q_0,B],0) = ([q_1,0],0,R) \qquad \delta([q_0,B],1) = ([q_1,1],1,R)$$

2) A partir do estado ($[q_1, X]$): se o símbolo armazenado no estado for diferente do símbolo da fita, simplesmente avança:

$$\delta([q_1,0],1) = ([q_1,0],1,R) \qquad \delta([q_1,1],0) = ([q_1,1],0,R)$$

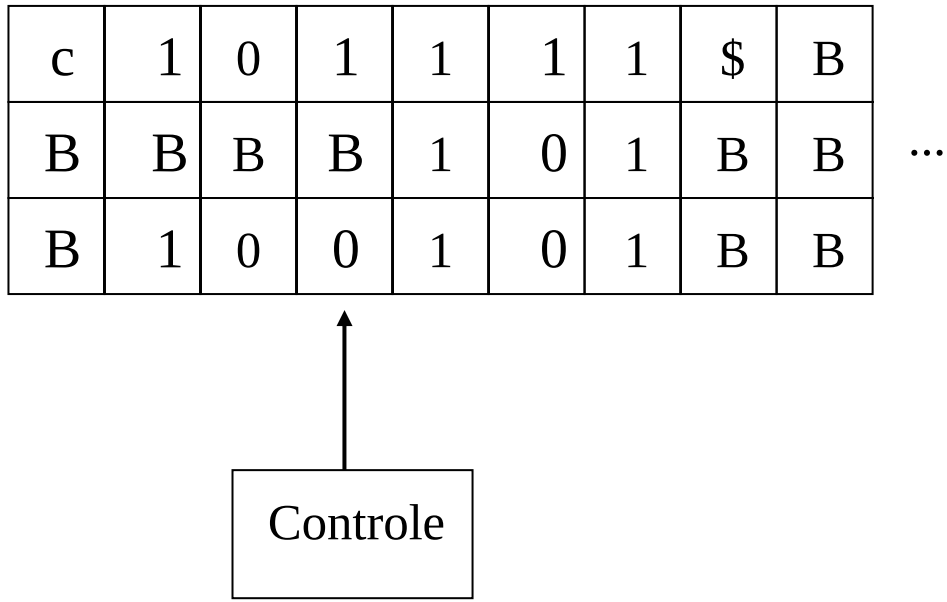
3) M entra no estado final ($[q_1, B]$) se chegar a um B sem ter antes encontrado um símbolo igual àquele armazenado no estado. Escreve o símbolo repetido no lugar do B, para referência, e volta a cabeça:

$$\delta([q_1,0],B) = ([q_1,B],0,L) \qquad \delta([q_1,1],B) = ([q_1,B],1,L)$$

4) Parada sem aceitação: $[q_1,0]$ e símbolo 0 ou $[q_1,1]$ e símbolo 1 (transições não definidas).

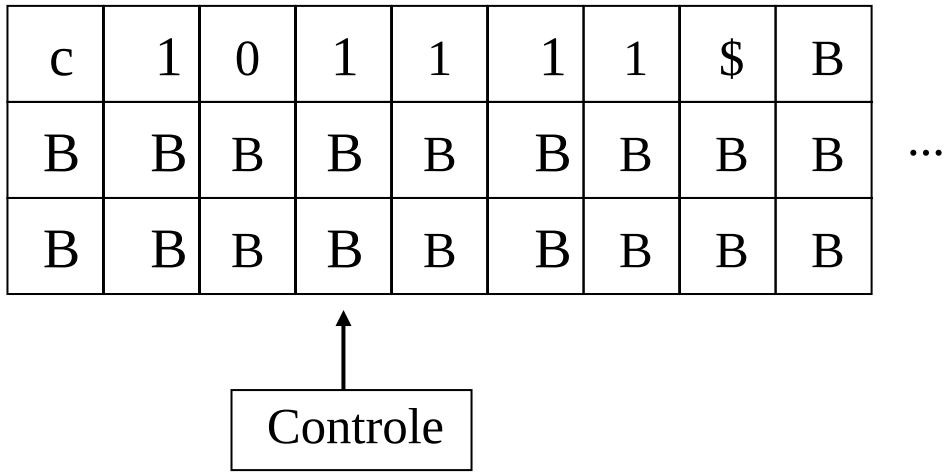
Armazenamento de Informação em Múltiplas Fitas

- Estado é escrito como uma n-upla de elementos:
  - Primeiro elemento: controle
  - Elementos restantes: informação (várias)



Exemplo: Verificador de Números Primos (> 2)

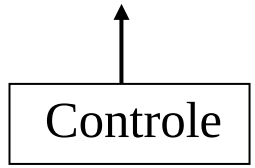
1. Número (base 2) escrito na primeira fita (cercado por c e $).



2. Copia conteúdo da primeira fita para terceira fita, escreve dois em binário na segunda:

c	1	0	1	1	1	1	$	B
c	1	0	$	B	B	B	B	B
c	1	0	1	1	1	1	$	B

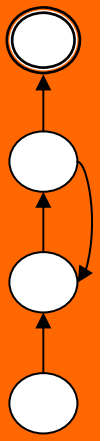
...



3. Subtrai conteúdo da segunda do conteúdo da terceira, tantas vezes quantas forem possíveis. O que sobra na terceira fita é portanto o resto da divisão.

c	1	0	1	1	1	1	$	B
c	1	0	$	B	B	B	B	B
c	1	0	1	1	0	1	$	B

...



c	1	0	1	1	1	1	$	B
c	1	0	$	B	B	B	B	B
c	1	0	1	0	1	1	$	B

...

⋮

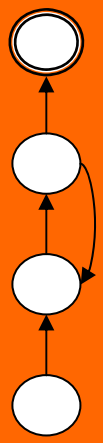
c	1	0	1	1	1	1	$	B
c	1	0	$	B	B	B	B	B
c	1	$	B	B	B	B	B	B

...

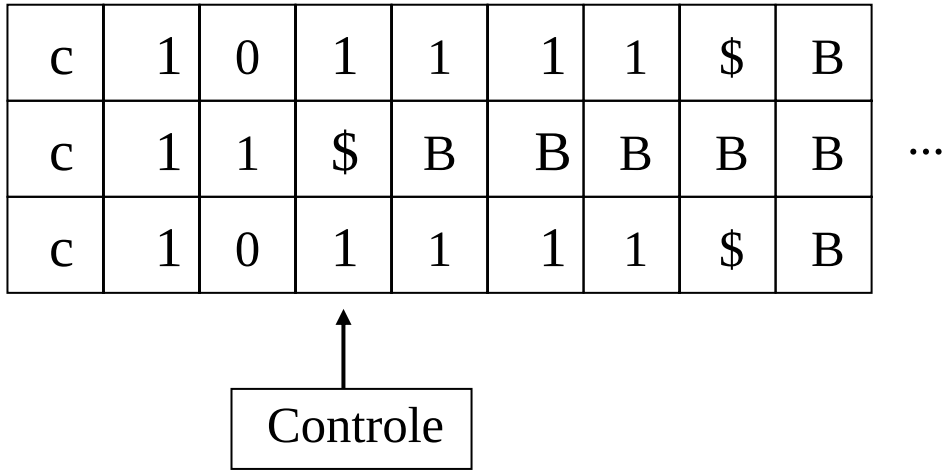
Controle

Sobrou 0 na terceira? Número não primo, parada da MT (s/ aceitação)

Senão...

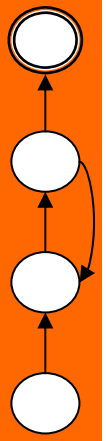


4. Copio conteúdo da primeira fita para terceira fita, incremento o conteúdo da segunda de uma unidade:



5. **Conteúdo da segunda fita igual ao da primeira?** Número primo, pára com aceitação.

   Senão volto ao passo 3.



Checagem de Símbolos

- Útil para visualizar reconhecimento de linguagens definidas por símbolos repetidos.
- Útil para comparar comprimentos de substrings em linguagens do tipo $a^i b^j c^k$.

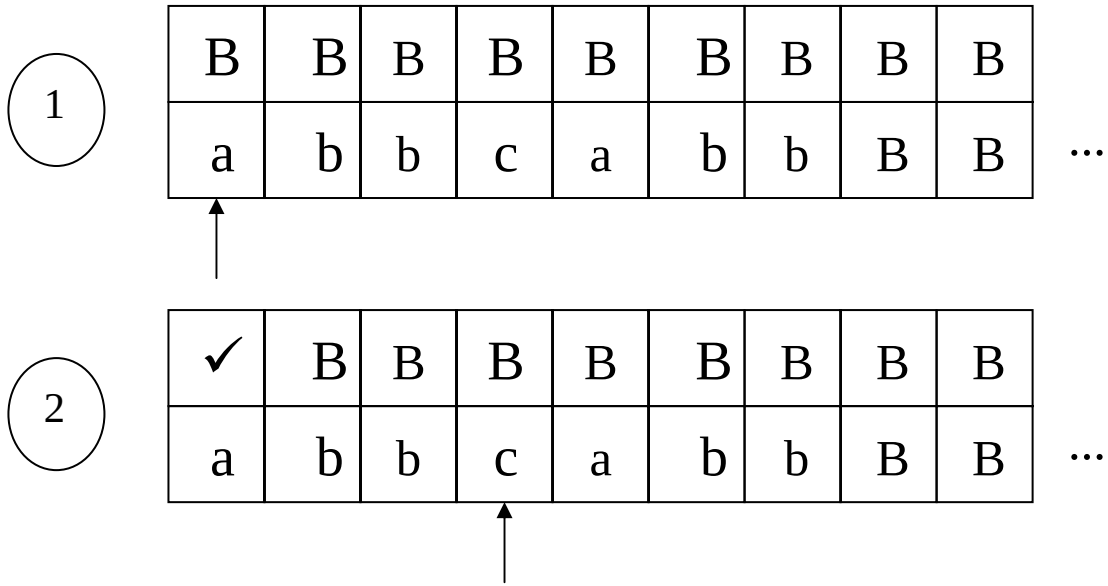
"Truque": utilizar uma fita extra que marca se o símbolo correspondente da outra fita foi (✓) ou não (B) considerado.

Exemplo

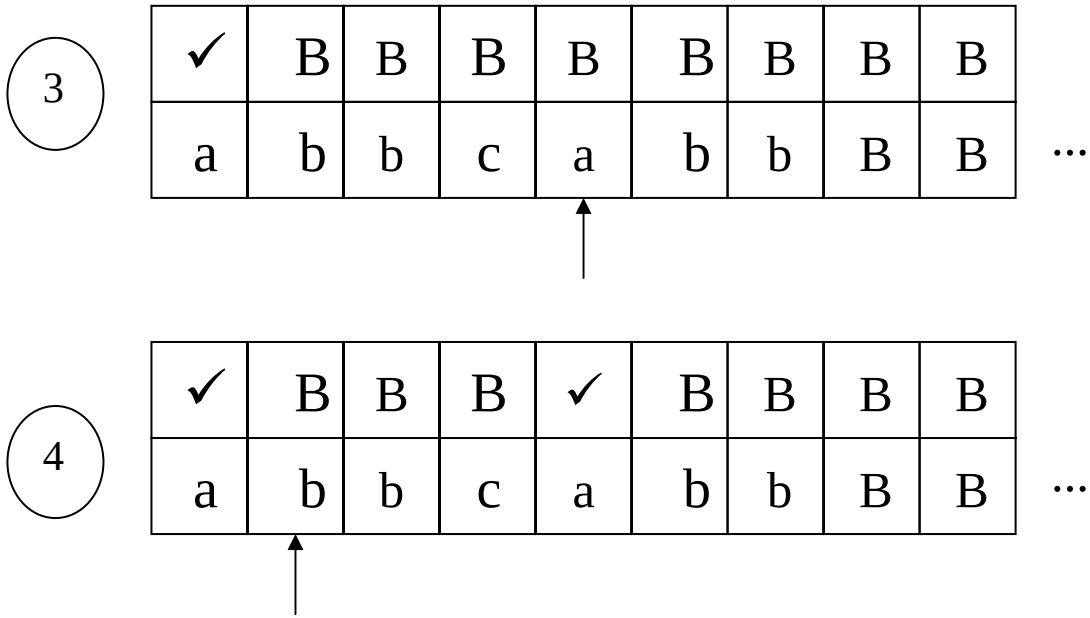
- Uma MT que reconhece {wcw | w $\in$ (a+b)*}

Idéia:

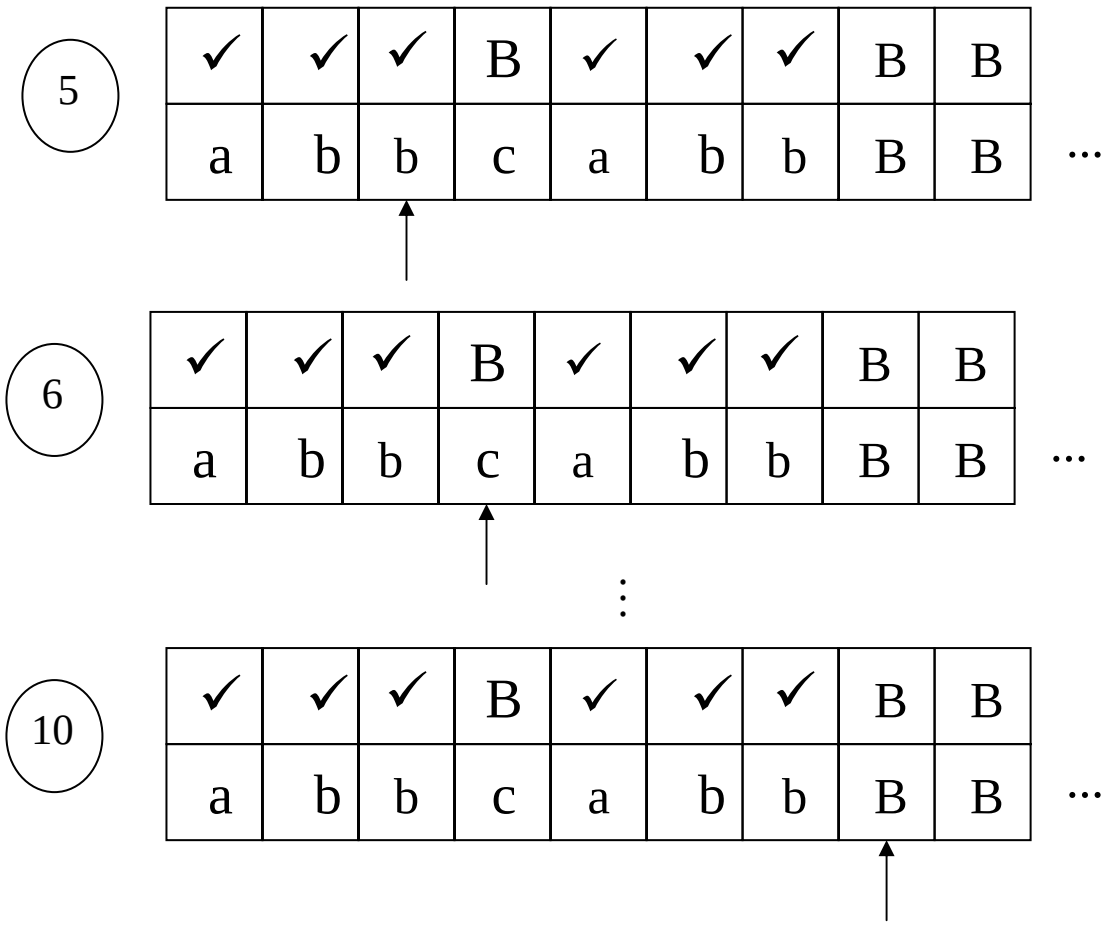
1. Marca o primeiro símbolo e registra-o no “estado”, avança sobre todos os símbolos não-marcados até achar o c.

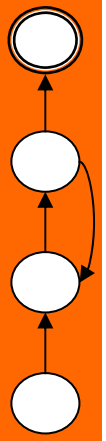


2. Após achar c, continua avançando, agora sobre símbolos marcados. Se o primeiro após c for igual ao registrado no estado, marca-o e volta para primeira posição após o símbolo marcado. Recomeça o processo a partir de 1. Caso contrário, pára sem aceitação.



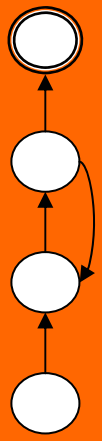
3. Quando terminar (ou seja, após marcar o último antes de c), tem que conferir se não sobrou nenhum não-marcado após o c.





Formalmente, $M=(Q, \Sigma, \Gamma, \delta, q_0, B, F\}$ onde

1. $Q=\{[q_i, x] \mid q \in \{q_1,q_2,...,q_9\}$ e x=a,b ou B}
2. $\Sigma=\{[B,y] \mid$ y=a,b ou c}, ou seja: um símbolo de entrada é sempre um símbolo "comum" acompanhado do B (que fica na primeira fita).
3. $\Gamma=\{[X,z] \mid$ X=B ou ✓ e z=a,b,c ou B}
4. $q_0=[q_1,B]$
5. $F=\{[q_9,B]\}$



Função de Transição δ (d = a ou b; e = a ou b)

1. $\delta([q_1,B],[B,d])=([q_2,d],[\checkmark,d],R)$

Lê símbolo em q_1, vai p/ q_2 guardando símbolo, marca fita e avança p/ direita.

2. $\delta([q_2,d],[B,e])=([q_2,d],[B,e],R)$

Em q_2 continua avançando p/ direita, enquanto não aparece o c.

3. $\delta([q_2,d],[B,c])=([q_3,d],[B,c],R)$

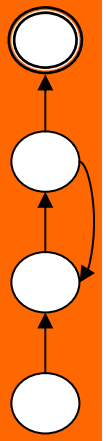
Ao achar o c, muda para estado q_3. Continua avançando p/ direita.

4. $\delta([q_3,d],[\checkmark,e])=([q_3,d],[\checkmark,e],R)$

Continua avançando p/ direita, ignorando os símbolos já marcados.

5. $\delta([q_3,d],[B,d])=([q_4,B],[\checkmark,d],L)$

Achou um símbolo não-marcado após o c. Muda p/ estado q_4, "limpa" a memória, marca a posição e inicia a volta.



6. $\delta([q_4,B],[\checkmark,d])=([q_4,B],[\checkmark,d],L)$

Em q_4 continua avançando p/ esquerda, ignorando os símbolos já marcados.

7. $\delta([q_4,B],[B,c])=([q_5,B],[B,c],L)$

Ao achar o c, muda para estado q_5. Continua avançando p/ esquerda.

8. $\delta([q_5,B],[B,d])=([q_6,B],[B,d],L)$

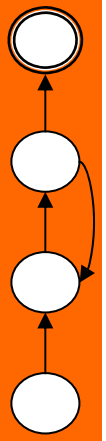
Símbolo à esquerda de c não marcado: muda estado, continua avanço p/ esquerda. Se estiver marcado, tenho que fazer teste (transição 11).

9. $\delta([q_6,B],[B,d])=([q_6,B],[B,d],L)$

Continua avançando p/ esquerda, ignorando os símbolos não marcados.

10. $\delta([q_6,B],[\checkmark,d])=([q_1,B],[\checkmark,d],R)$

Achou um símbolo marcado antes do c. Muda p/ estado q_1, "limpa" a memória, e reinicia a ida (transição 1)



11. $\delta([q_5,B],[\checkmark,d])=([q_7,B],[\checkmark,d],R)$

O estado q_5 indica que o c foi encontrado na última transição, após a qual a cabeça foi p/ esquerda (transição 7). Símbolo à esquerda de c marcado, começa o teste: muda p/ estado q_7, volta p/ direita

12. $\delta([q_7,B],[B,c])=([q_8,B],[B,c],R)$

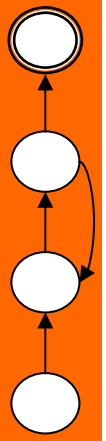
Teste em andamento: ao achar c, muda p/ q_8 e continua avançando p/ direita.

13. $\delta([q_8,B],[\checkmark,d])=([q_8,B],[\checkmark,d],R)$

Teste em andamento: símbolo à direita de c marcado, continua avançando p/ direita.

14. $\delta([q_8,B],[B,B])=([q_9,B],[\checkmark,B],L)$

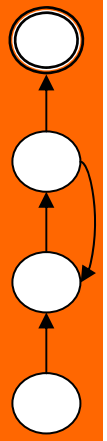
Termina o teste: achou [B,B], vai p/ estado final e pára (com aceitação). P/ qualquer outro [X,Y] na fita com estado q_8, ocorre parada sem aceitação.



Deslocamento de Símbolos

- Uma operação útil: deslocar símbolos diferentes de B para a direita.
- Idéia:
  - Lê o símbolo da fita, armazena-o no estado.
  - Substitui o símbolo da fita por aquele que havia sido lido quando da leitura realizada quando cabeça estava na posição imediatamente a esquerda.

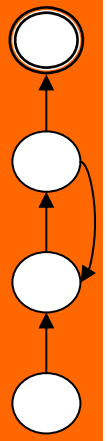
Um exemplo: Ex. 7.6, Hopcroft/Ullman



Subrotinas

- A operação de MTs pode ser decomposta em um conjunto de “subrotinas”, que correspondem a diferentes partes de um programa.
- Idéia: uso estados específicos para controlar as chamadas e retornos da subrotina.

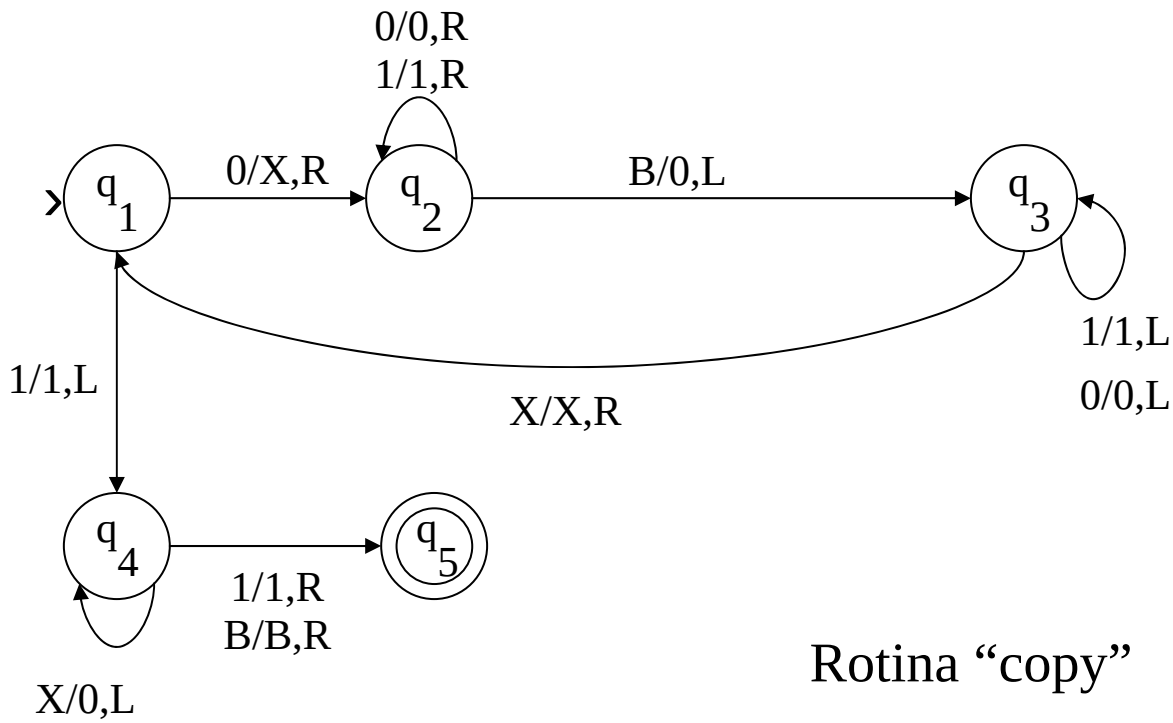
Um exemplo: Ex. 7.7, Hopcroft/Ullman



Visualização de sub-rotinas

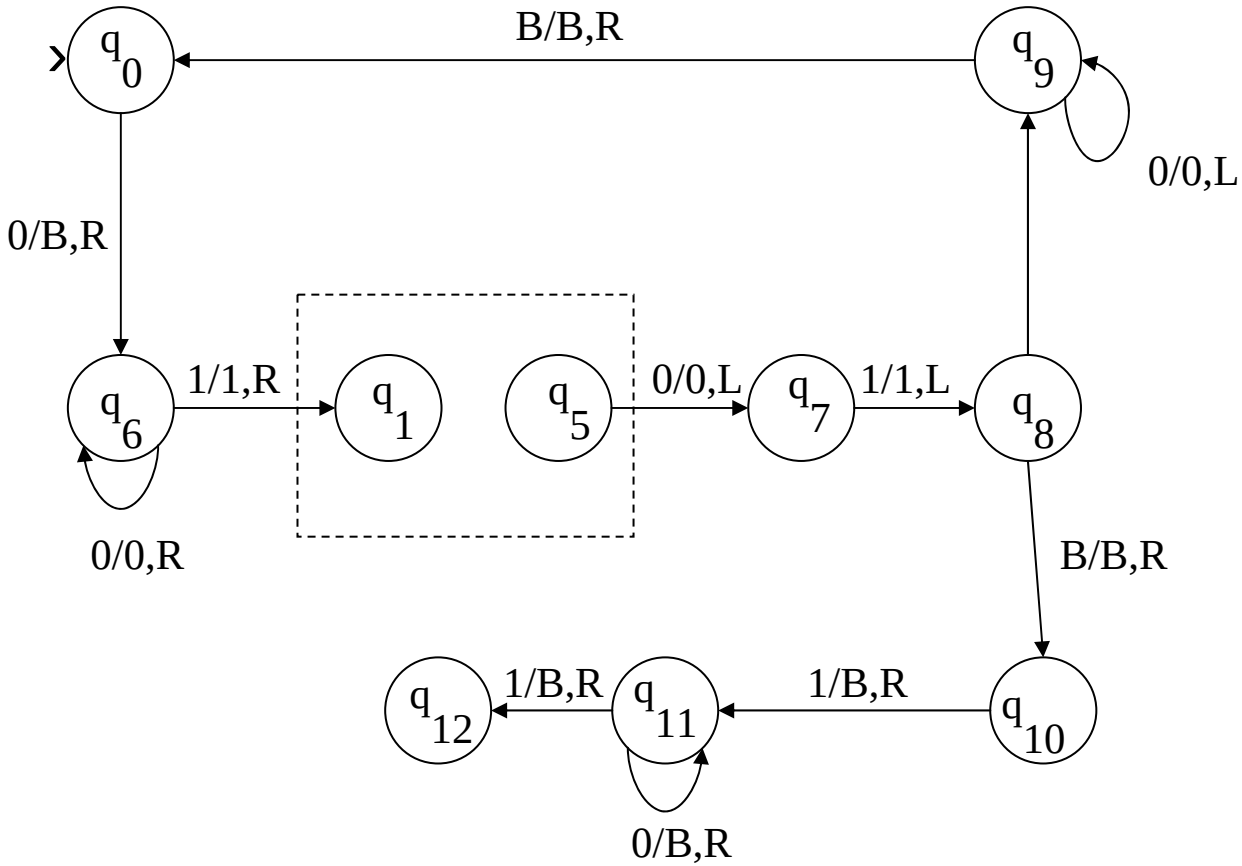
- Uma MT para executar multiplicações:
  - Início: 0^m10^n1 na fita
  - Fim: 0^{mn} na fita
  - Estratégia:
    1. Em geral, a fita conterá $0^i10^n10^{kn}$ para algum k.
    2. Mudamos um 0 no primeiro grupo para B e **adicionamos n 0´s ao último grupo.** Resultado: $0^{i-1}10^n10^{(k+1)n}$
    3. O processo acima é executado m vezes, e a cada vez um 0 no primeiro grupo é trocado por B. Quando o primeiro grupo de 0´s tiver sido completamente trocado por B´s, teremos mn 0´s no último grupo.
    4. Etapa final: trocar os 10^n por B's.

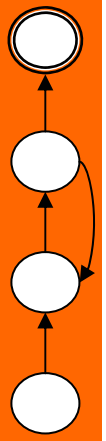
Visualização de sub-rotinas



Rotina “copy”

Programa de multiplicação usando a rotina “copy”



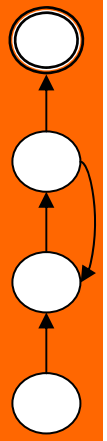


Modificações da Máquina de Turing (1)

- Fita infinita nas duas direções.
  - Similar à MT usual, mas admitindo deslocamento nas duas direções (parte à esquerda do símbolo inicial na fita é ocupado com B's).

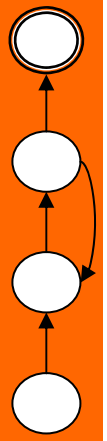
Teorema: L é reconhecida por um MT com fita infinita nas duas direções sss L é reconhecida por uma MT usual. ☹

Modificações da Máquina de Turing (2)



- MT multifita.
  - Similar à MT usual, mas admitindo várias k fitas infinitas bidirecionais e k cabeças, comandadas por um único controle.
  - Para cada estado do controle e conjunto de símbolos lidos por cada uma das cabeças:
    - muda estado;
    - imprime um novo símbolo em cada uma das fitas;
    - move ou não cada uma das cabeças independentemente, para a direita ou esquerda.

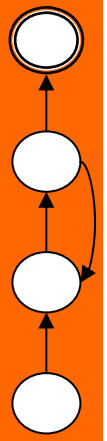
Teorema: L é reconhecida por um MT multifita sss L é reconhecida por uma MT usual. ☹



Modificações da Máquina de Turing (3)

- MT não-determinística
  - Similar à MT usual, mas admitindo escolha não-determinística do trio <novo estado, novo símbolo a ser impresso, direção de movimento da cabeça>.

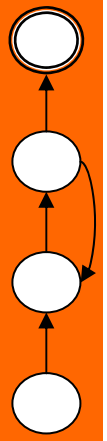
Teorema: L é reconhecida por um MT não-determinística sss L é reconhecida por uma MT usual. ☹



Modificações da Máquina de Turing (4)

- MT multidimensional
  - Similar à MT usual, mas admitindo fita como uma matriz infinita (no espaço n-dimensional). Inicialmente, a string a ser reconhecida é escrita em uma das direções, todo o resto da matriz sendo ocupado por B's.
  - Para cada estado do controle e símbolo lido:
    - muda estado;
    - imprime um novo símbolo na matriz;
    - move cabeça em uma das n direções, sentido positivo ou negativo.

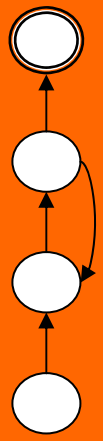
Teorema: L é reconhecida por um MT multidimensional sss L é reconhecida por uma MT usual. ☹



Modificações da Máquina de Turing (5)

- MT multicabeças
  - Similar à MT usual, mas admitindo fita lida por k cabeças independentes. Para cada estado do controle e conjunto de símbolos lidos pelas cabeças:
    - muda estado;
    - cada cabeça imprime um novo símbolo;
    - move ou não cabeças independentemente, para direita ou esquerda

Teorema: L é reconhecida por um MT multicabeças sss L é reconhecida por uma MT usual. ☹

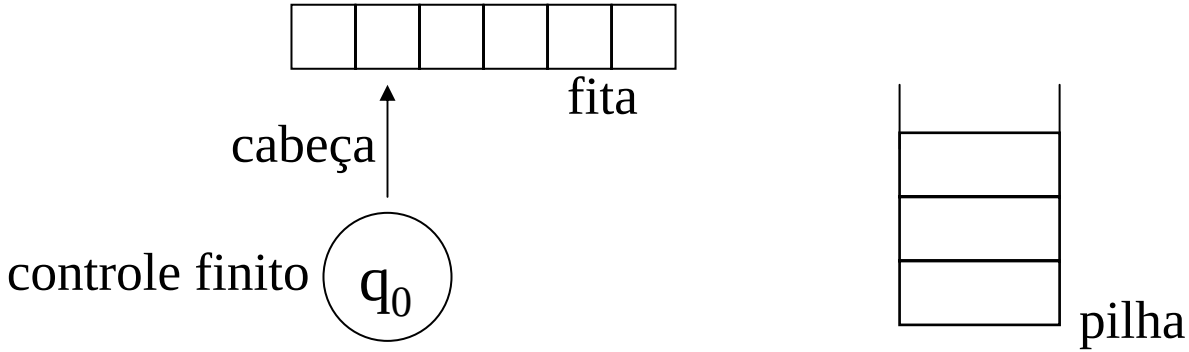


Máquinas equivalentes a MTs

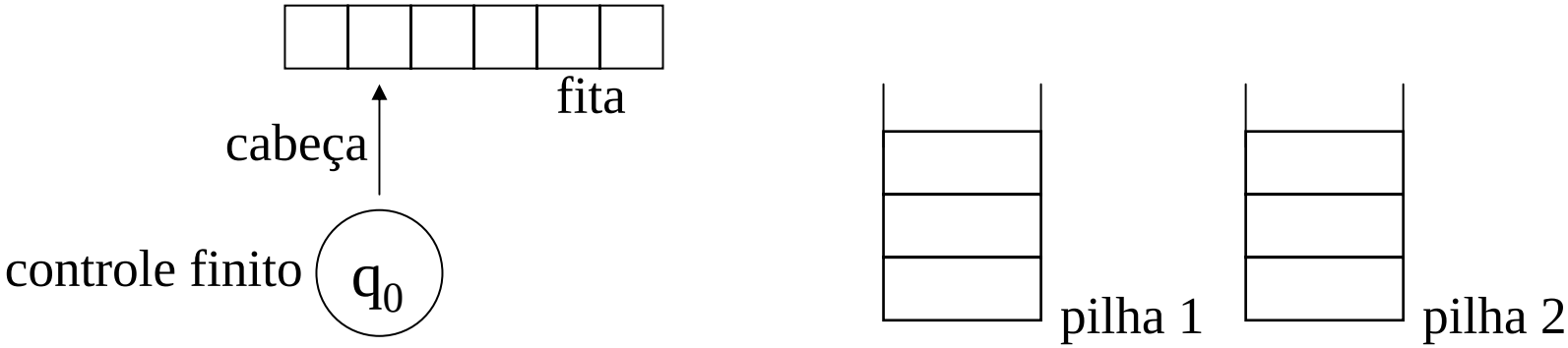
- Vimos que uma MT pode ter várias realizações equivalentes, aparentemente mais complexas:
  - várias fitas
  - memória no estado
  - não-determinismo
- Ok, mas... Existe máquinas aparentemente mais simples do que MTs com o mesmo poder computacional?

SIM!

Autômato de “Pilhas”



Reconhece LLCs ⇒ menos poderoso do que uma MT



Reconhece LREs ⇒ equivalente a MTs !!!

Autômato de “Pilhas”

Início:

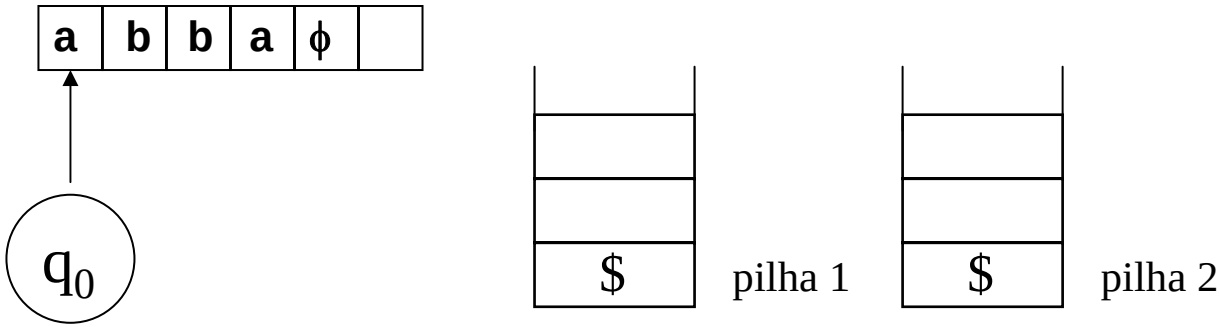
Pilhas vazias ($ no fundo)

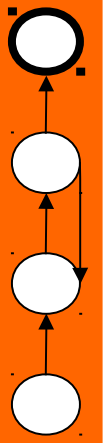
Cadeia na fita com final marcado por um símbolo ϕ.

Regra de transição generalizada: $\delta(q,a,X_1,X_2,...,X_k) = (p, \Gamma_1, \Gamma_2,..., \Gamma_k)$

X_i: símbolos

Γ_i: cadeias





A.M. Turing Award

ACM's most prestigious technical award is accompanied by a prize of **US$100.000**. It is given to an individual selected for contributions of a technical nature made to the computing community. The contributions should be of lasting and major technical importance to the computer field.

Some Former Award Recipients

1968 Richard Hamming 1969 Marvin Minsky 1971 John McCarthy 1972 E.W. Dijkstra 1974 Donald E. Knuth 1975 Allen Newell	1975 Herbert A. Simon 1983 Dennis M. Ritchie 1984 Niklaus Wirth 1986 John Hopcroft 1994 Edward Feigenbaum 2003 Alan Kay

A.M. Turing Award

2012 Shafi Goldwasser e Silvio Micali

"For transformative work that laid the complexity-theoretic foundations for the science of cryptography and in the process pioneered new methods for efficient verification of mathematical proofs in complexity theory"

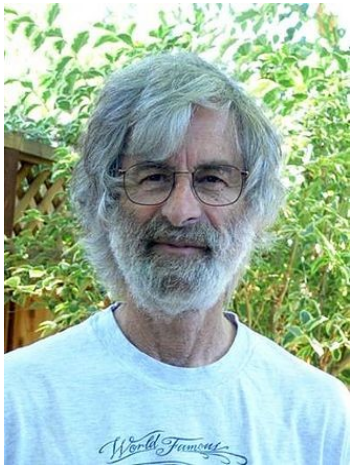
MIT CSAIL, Weizmann Institute of Science

MIT CSAIL

A.M. Turing Award

2013 Leslie Lamport

"For fundamental contributions to the theory and practice of distributed and concurrent systems, notably the invention of concepts such as causality and logical clocks, safety and liveness, replicated state machines, and sequential consistency."



Microsoft Research
Mountain View, California