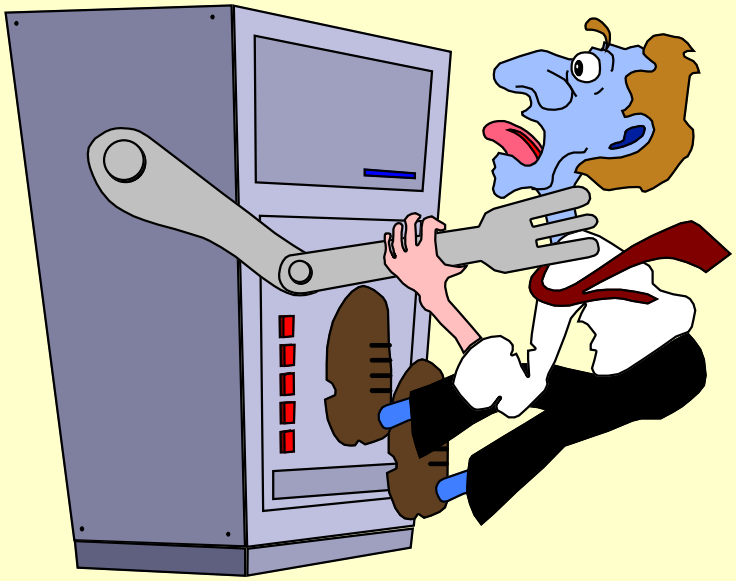
Máquinas de Turing



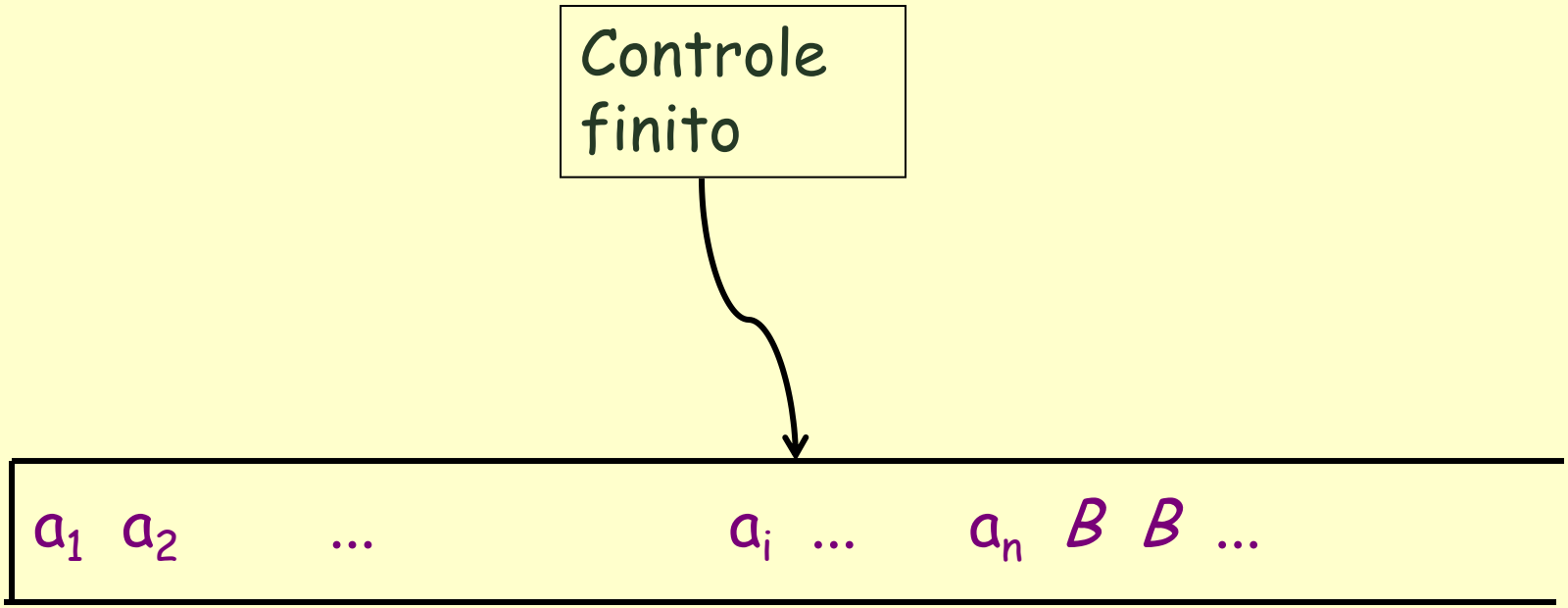
Máquinas de Turing podem fazer tudo o que um computador real faz.

Porém, mesmo uma Máquina de Turing não pode resolver certos problemas. Estes problemas estão além dos limites teóricos da computação

História

- Turing (1936): Máquinas de Turing como modelo de função computável.
- Tese de Church-Turing: qualquer modelo geral de computação permite calcular as mesmas funções (ou, tudo o que se pode computar coincide com as linguagens reconhecidas pelas Máquinas de Turing).

Máquina de Turing



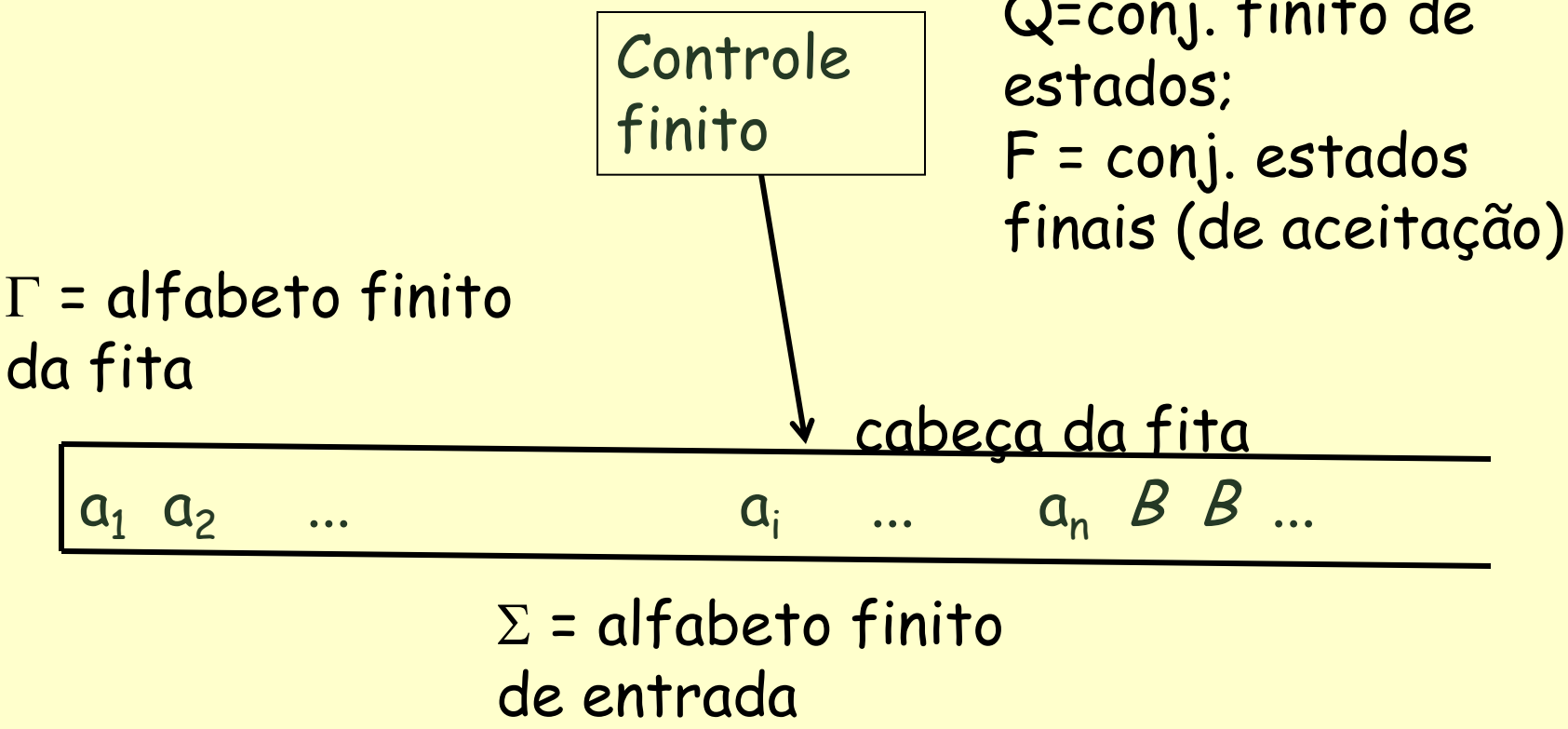
Inicialmente, a entrada é colocada na fita. Todas as outras células (infinitamente à esquerda e à direita) têm um símbolo especial da fita, *B* (*branco*).
A cabeça da fita fica posicionada em uma das células. No início, a cabeça está posicionada na célula mais à esquerda que contém a entrada.

Um *movimento* da MT é uma função do estado do controle finito e do símbolo atual da fita. Em um movimento, a MT:

1. Mudará de estado (opcionalmente para o mesmo).
2. Gravará um símbolo de fita na célula atual, substituindo o existente (podendo ser o mesmo).
3. Movimentará (necessariamente) a cabeça da fita uma célula à esquerda ou à direita.

MT: notação formal

M = (Q, Σ, Γ, δ, qo, F)



Máquina de Turing

Função de transição δ:

$\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\}$

Ou seja, $\delta(q,a) = (p,b,D)$ onde:

- *p* é o próximo estado em Q;
- *b* é o símbolo que substituirá *a* na fita;
- *D* é uma direção (esquerda ou direita) em que a cabeça da fita irá se mover.

A linguagem de uma MT

Linguagem é um conjunto de *cadeias* de símbolos sobre um alfabeto/vocabulário, V. É um subconjunto específico de V*. Estas cadeias são denominadas *sentenças* da linguagem, e são formadas pela justaposição de elementos individuais, os símbolos da linguagem.

- Exs: V = {a, b, c} ; L1 = {ab, bc}
- L2={ w | w consiste de um número igual de 0's e 1's }
- L3={ w | w é um número inteiro binário primo }
- L4={ w | w é um programa em C sintaticamente correto }
- L5={ $0^n 1^n$ | $n \geq 1$} = {01, 0011, 000111, ...}
- L6={$0^i 1^j$ | $0 \leq i \leq j$} = {λ, 01, 011, 011, ..., 0011, 00111, ...}

A linguagem de uma MT

- Intuitivamente: a cadeia de entrada é colocada na fita, e a cabeça da fita começa no símbolo mais à esquerda da cadeia. Se a MT parar eventualmente num estado de aceitação (de F), a entrada é dita *aceita ou reconhecida*; caso contrário, não.

- Assim, a Linguagem L(M) Aceita ou Reconhecida por uma MT, $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$, é o conjunto de cadeias w em Σ^* tais que $f(q_0, w_1)$ leva a uma sequência de transições que culminem num estado p de F (aceitação por estado final). Não importa o conteúdo final da fita neste caso!

A linguagem de uma MT

- As linguagens aceitas por MT são também chamadas de *linguagens recursivamente enumeráveis (LRE)*
- Pela Tese de Church, as funções computáveis coincidem com as LRE

Um **Problema** na Teoria dos Autômatos consiste em decidir se uma dada cadeia é elemento (sentença) de alguma linguagem específica. Assim, um problema é sinônimo de decidir a pertinência a um conjunto (uma linguagem). Mais precisamente:

Se Σ é um alfabeto e L é uma linguagem sobre Σ, então o **problema L** é:

Dada uma cadeia w em Σ^*, decidir se w está ou não em L. (Questão de Decisão)

Ex.: O problema de testar o caráter primo de um número binário pode ser expresso pela linguagem Lp que consiste em todas as cadeias binárias cujo valor como número binário é primo. Ou seja, dada uma cadeia de 0's e 1's, diremos *sim* se a cadeia for a representação binária de um primo e diremos *não* caso contrário. Lp é então o conjunto-solução do problema.

Linguagem Lp ou Problema P?

- São dois lados da mesma moeda.
- A Linguagem Lp é o conjunto das soluções de um problema P.
- Todo problema P tem um problema de decisão associado: dada uma (cadeia) candidata à solução, decidir se ela de fato é solução (se tem as características de solução). Ou seja, se pertence à Lp.
- Diferentemente, um algoritmo que **busca** uma solução tem o trabalho extra de verificar mais de uma possibilidade.

Encarar problemas como uma questão de pertinência de conjunto (pertence ou não à linguagem) tem sido útil aos estudos da teoria da complexidade.

Problemas de Decisão têm se mostrado tão difíceis quanto suas versões do tipo "resolva isso".

Ao reduzirmos um problema para sua versão de pertinência e provarmos que é difícil resolvê-la, então podemos concluir que resolver o problema inicial será igualmente difícil. **(Técnica de Prova por Redução)**

$P \equiv i \in Lp$?

Exemplo

- Seja o problema P1, de compilar um programa na linguagem de programação X. Quão difícil é resolvê-lo? (qual sua complexidade?).
- Considere agora o problema de decisão P2: dada uma cadeia, ela pertence à linguagem Lx (de cadeias válidas na linguagem de programação X)?
- Se provarmos que é difícil resolver P2 (H), então **não será mais fácil** converter programas na linguagem X para código-objeto (C).
- Por contrapositiva: se fosse fácil (eficiente) gerar código-objeto, poderíamos usar o próprio compilador para, ao ter sucesso ao produzir o código para um programa (cadeia), concluir que essa entrada se trata de um elemento válido de Lx. Assim, seria fácil testar a pertinência. Ou seja, se ~C→~H.

Assim, temos uma prova por contrapositiva da afirmação:

Se o teste de pertinência a Lx é difícil, então compilar programas na linguagem de programação X é difícil.

Essa técnica de redução é extremamente útil no estudo da complexidade de problemas, e é facilitada pela noção de que problemas são questões sobre pertinência a uma linguagem, e não tipos mais gerais de questões.

MT e sua parada

• Há uma outra noção de "aceitação" para MT: a aceitação por parada. Em geral, usada quando o conteúdo final da fita representa alguma resposta ao problema que a MT representa.

•Dizemos que uma MT *pára* se ela entra em um estado q, olhando um símbolo de fita a, e não existe qualquer movimento previsto nessa situação, i.e., $\delta(q,a)$ é indefinida.

Usos de uma MT

- como reconhecedor de linguagens
- para calcular funções

Usos de uma MT

- como reconhecedor de linguagens

  (*A linguagem corresponde ao conjunto de instâncias que geram resposta SIM ao problema de decisão associado: esta cadeia pertence à linguagem L?* ou *esta é uma solução do problema?*)

- para calcular funções

  (equivale ao problema propriamente dito)

Exemplo

- Vamos projetar uma MT para reconhecer

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

Ou seja, para toda entrada de cadeias de 0s seguidos de 1s, em igual número, a MT deve parar num estado final.

Ex.: 01, 0011, 000111, etc. Entradas como 1, 10, 001, 1010, etc. não podem fazer parar a MT num estado final.

Fita inicial: 000...111*BBBBBB*

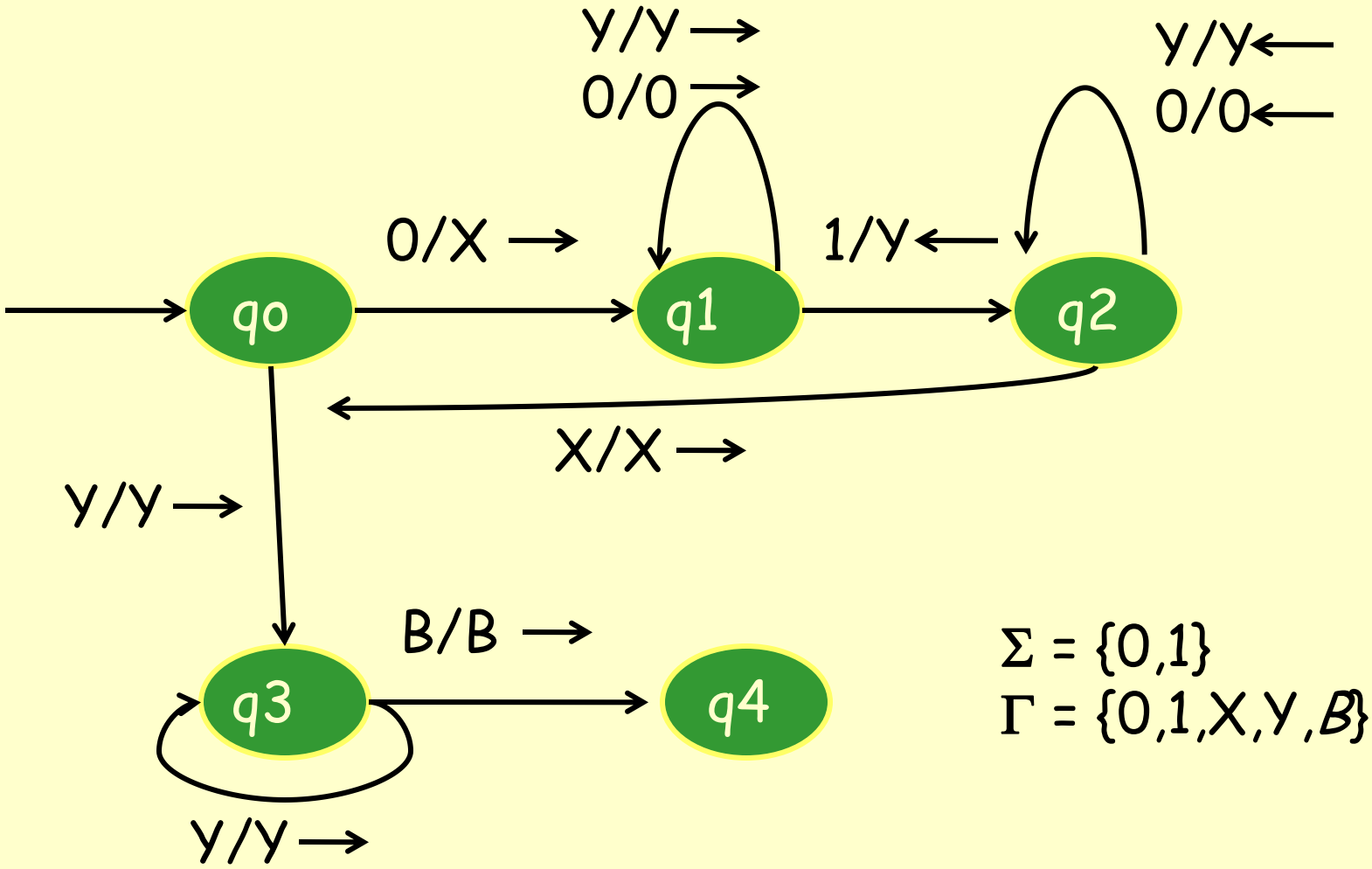
- Estratégia: em cada passo, a MT trocará um 0 por um X, e depois um 1 por um Y, até todos os 0s e 1s terem sido trocados aos pares.

Exemplo

- Em cada passo, da esq. para dir., ela troca um 0 por X e vai para a direita, ignorando 0s e Ys até encontrar 1. Troca esse 1 por Y e se move para a esquerda, ignorando Ys e 0s, até encontrar um X. Procura um 0 a direita e troca por X, repetindo o processo.

- Se a entrada não for da forma 0^n1^n eventualmente a MT não vai ter um movimento previsto (previmos apenas os movimentos para cadeias válidas) e vai parar sem aceitar, ou seja, num estado que não é de F.
-
- Se, por outro lado, na busca por mais um 0, ela só encontrar Xs e Ys, então ela descobre que deve aceitar a entrada, e vai para um estado final.

Diagrama de Transição



$\Sigma = \{0,1\}$

$\Gamma = \{0,1,X,Y,B\}$

$M = (\{q0,q1,q2,q3,q4\}, \{0,1\}, \{0,1,X,Y,B\}, \delta, q0, \{q4\})$

Estado	0	1	X	Y	B
→ q0	(q1,X,R)	--	--	(q3,Y,R)	--
q1	(q1,0,R)	(q2,Y,L)	--	(q1,Y,R)	--
q2	(q2,0,L)	--	(q0,X,R)	(q2,Y,L)	--
q3	--	--	--	(q3,Y,R)	(q4,B,R)
q4*	--	--	--	--	--

Verifique se a cadeia 000111 é aceita

- Qual é a complexidade dessa MT (desse algoritmo)?

- Qual é a complexidade dessa MT (desse algoritmo)?
- Seja n é o tamanho da cadeia de entrada;
- (a) Em cada passo, busca-se um 0 e um 1 na cadeia, portanto, no máximo todos os símbolos da cadeia são lidos;
- (b) O número de passos aumenta linearmente em relação ao tamanho da cadeia.
- Logo, (a) * (b) = $O(n^2)$

Exercício

- Construa uma MT para reconhecer cadeias de $L=\{w\#w \mid w \in \{0,1\}^*\}$

Estágios para a resolução:

- Verifique se a entrada tem um único símbolo #, cc rejeite.
- Verifique (zigue-zague) se antes e depois do # existem os mesmos símbolos, cc rejeite. Ao checar um símbolo marque-o (use um X por exemplo) para ter controle sobre os que estão sendo analisados num dado momento.
- Quando todos os da esquerda forem checados (com X) verifique se existe algum símbolo à direita ainda não checado. Se houver, rejeite; cc aceite.
- Por raciocínio análogo ao anterior, $O(n^2)$

MT como um processador de funções inteiras

- Tradicionalmente, os inteiros são representados em vocabulário unário.
- O inteiro i >= 0 é representado pela cadeia 0^i.
- Se a função tem k argumentos (i1, i2, ..., ik) então esses inteiros são colocados na fita separados por 1´s como:

$$0^{i1}\ 1\ 0^{i2}\ 1\ \ldots\ 1\ 0^{ik}$$

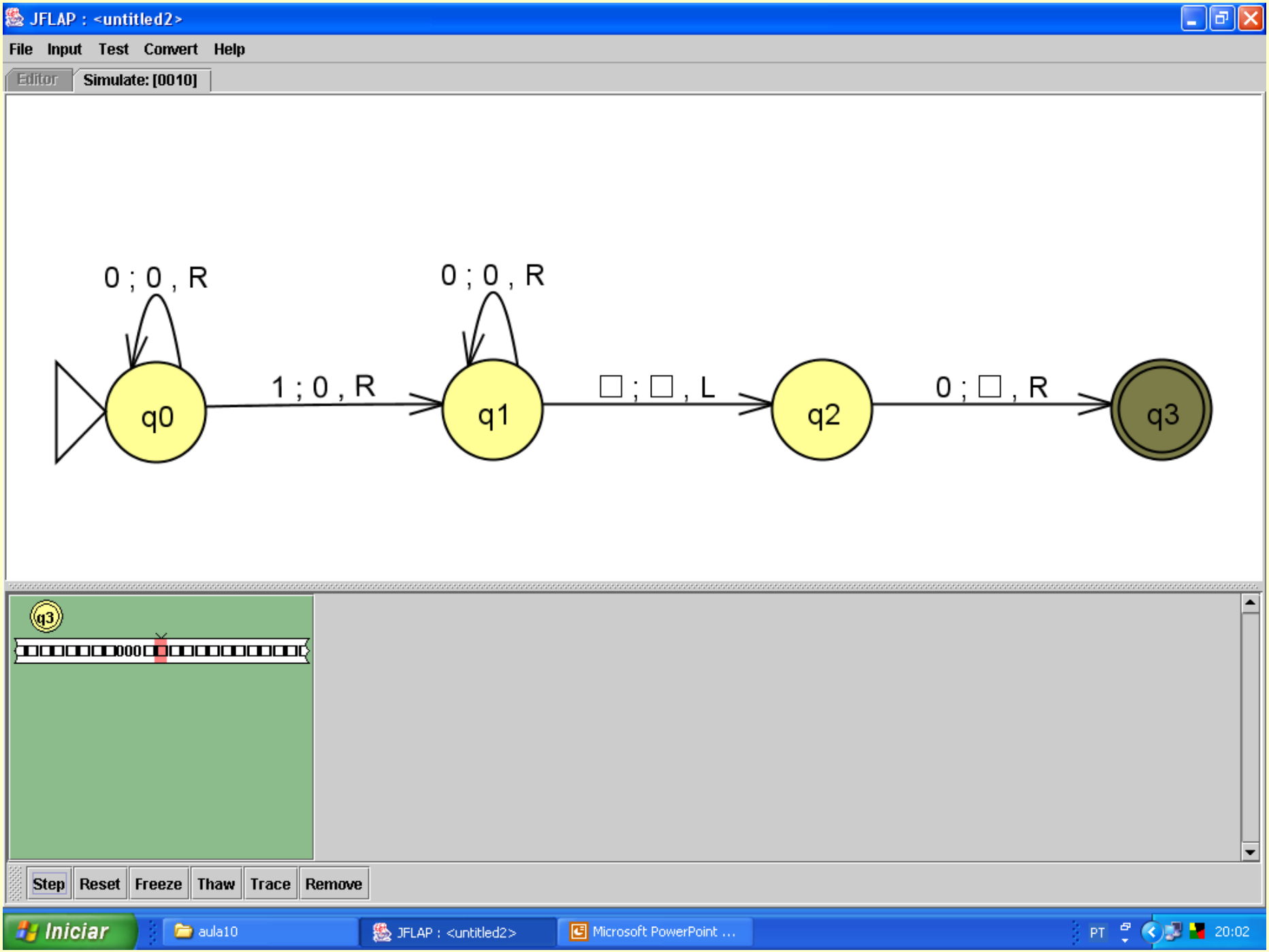
- O inverso também é possível.
- Se a máquina pára (não importa em que estado) com a fita consistindo de 0^m para algum m, então dizemos que f(i1,i2,...ik) = m, onde f é uma função de k argumentos computados por essa MT.

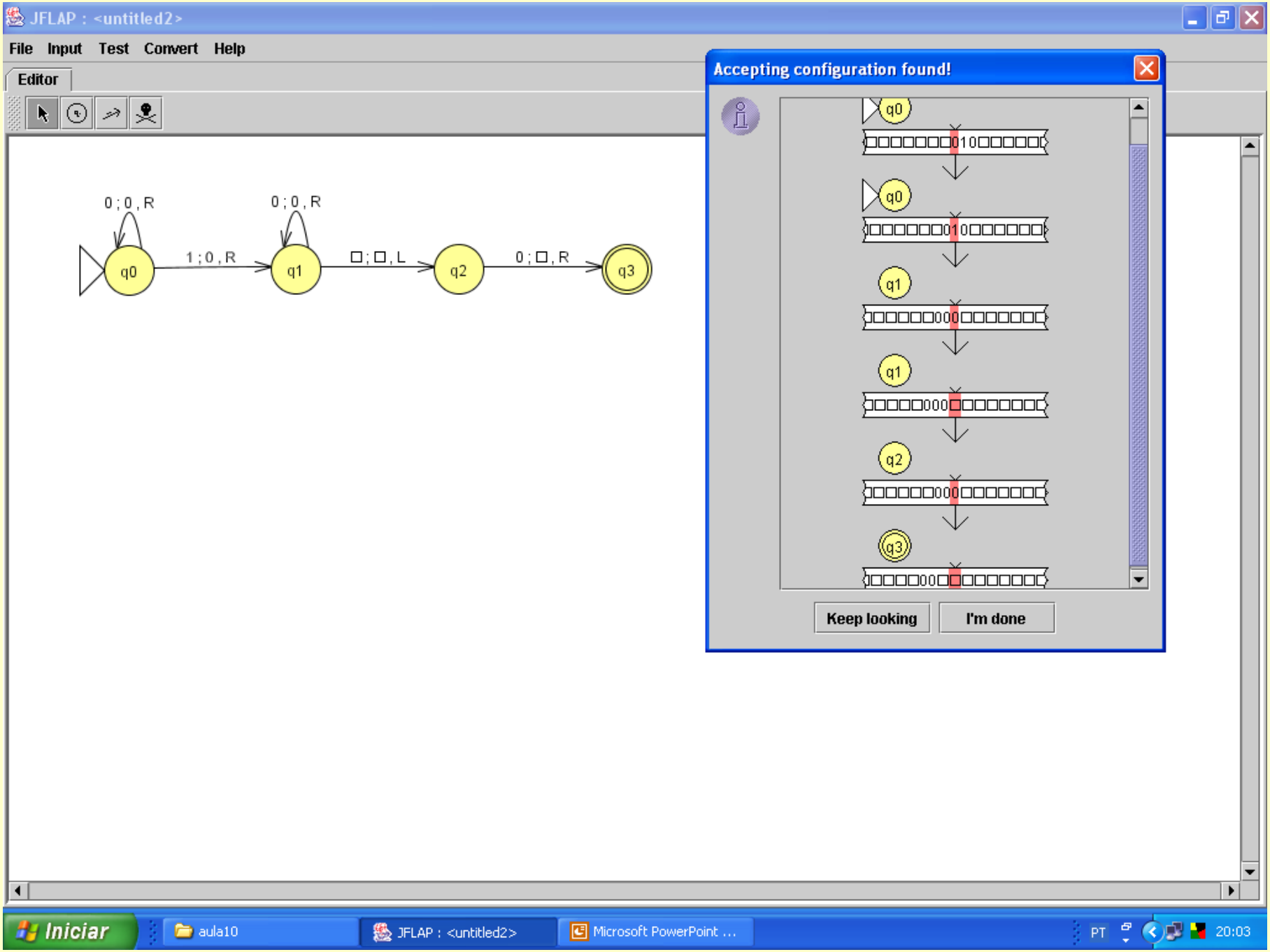
Exemplo: MT que soma dois números naturais, a + b

- Conteúdo inicial da Fita: $0^a\ 1\ 0^b\ B\ldots$
- Quando a MT parar, o conteúdo da fita dever ser:

$$0^{a+b}\ B\ldots.$$

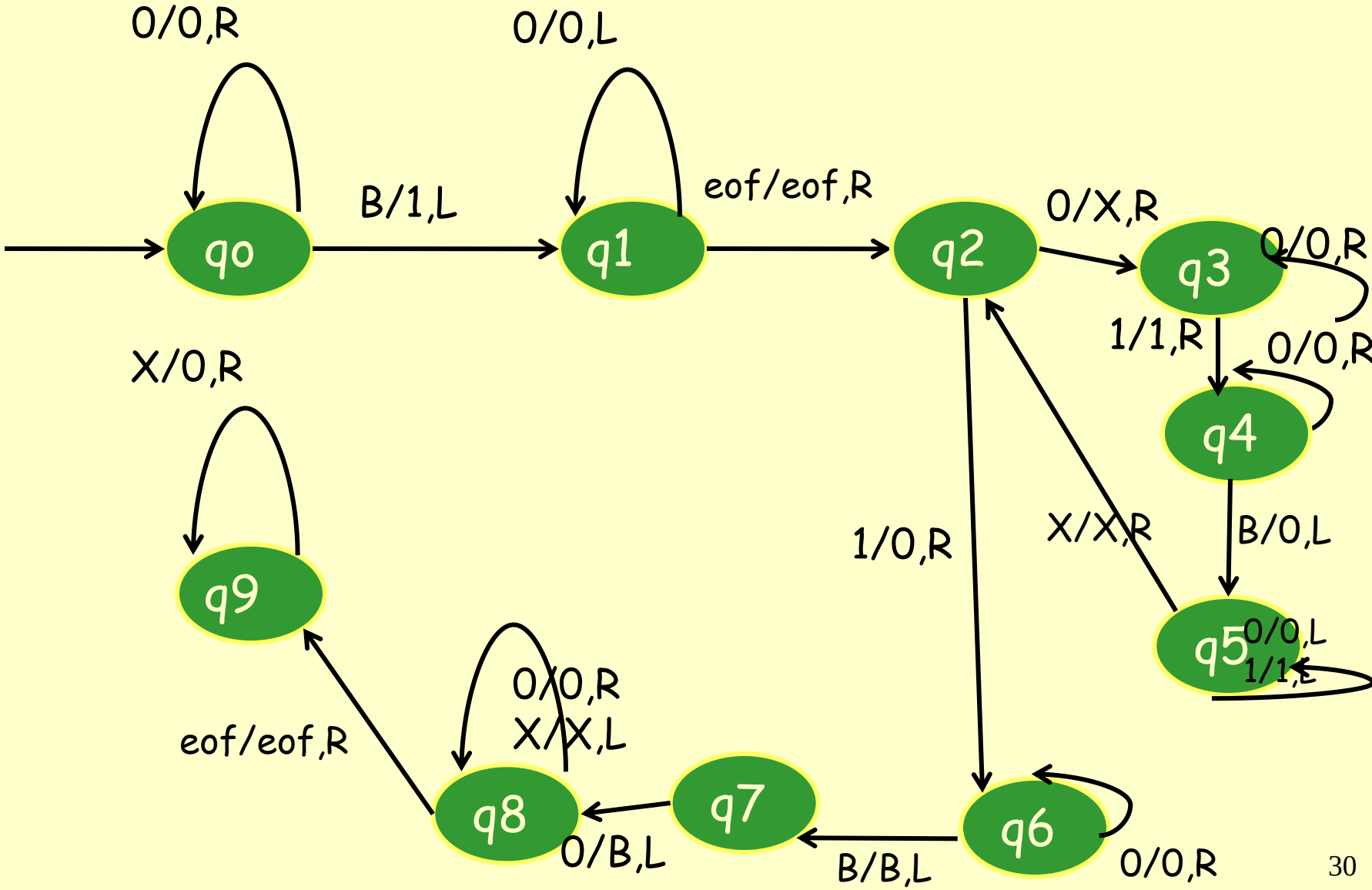
- Processo:
- Ler o 0 mais à esquerda, mantendo-o como 0, e mover à direita até encontrar o 1.
- Substitua o 1 por 0 (nesse momento a cadeia da fita é 0^{a+b+1}. Continue movendo à direita sem mudar a fita, até que um *B* seja encontrado.
- Mantenha o *B* e mova a esquerda para encontrar o último 0 mais a direita.
- Substitua esse 0 por *B*. O resultado é $B\ 0^{a+b}$
- Qual é a complexidade desse algoritmo?


JFLAP : <untitled2>
File Input Test Convert Help
Editor
Simulate: [0010]
0 ; 0 , R
0 ; 0 , R
1 ; 0 , R
□ ; □ , L
0 ; □ , R
q0
q1
q2
q3
q3
Step Reset Freeze Thaw Trace Remove
Iniciar
aula10
JFLAP : <untitled2>
Microsoft PowerPoint ...
PT
20:02


JFLAP : <untitled2>
File Input Test Convert Help
Editor
0 ; 0 , R
0 ; 0 , R
1 ; 0 , R
□ ; □ , L
0 ; □ , R
q0
q1
q2
q3
Accepting configuration found!
Keep looking
I'm done
Iniciar
aula10
JFLAP : <untitled2>
Microsoft PowerPoint ...
PT
20:03

Exemplo: MT que multiplica um número natural por 2- a * 2

- Conteúdo inicial da Fita: 0^a *B*...
- Quando a MT parar, o conteúdo da fita dever ser:
  0^{a+a} *B*....
- Sugestão:
- Grave 1 depois do último 0.
- Grave o mesmo número de 0s da entrada, à direita do 1.
- Substitua o 1 por 0 (nesse momento a cadeia da fita é 0^{a+a+1}). Continue movendo à direita sem mudar a fita, até que um *B* seja encontrado.
- Mantenha o *B* e mova a esquerda para encontrar o último 0 mais a direita.
- Substitua esse 0 por *B*. O resultado é 0^{a+a}


0/0,R
0/0,L
B/1,L
eof/eof,R
0/X,R
0/0,R
1/1,R
0/0,R
X/0,R
1/0,R
X/X,R
B/0,L
0/0,L
1/1,L
0/0,R
X/X,L
eof/eof,R
0/B,L
B/B,L
0/0,R
q0
q1
q2
q3
q4
q5
q6
q7
q8
q9

- F é vazio se a MT computa uma função.

- F é relevante quando a MT é usada para reconhecer uma linguagem.

Ex. Uma MT para reconhecer a Linguagem

$$L = \{ a^n b^n c^n \mid n \geq 0 \}$$

Exemplos:

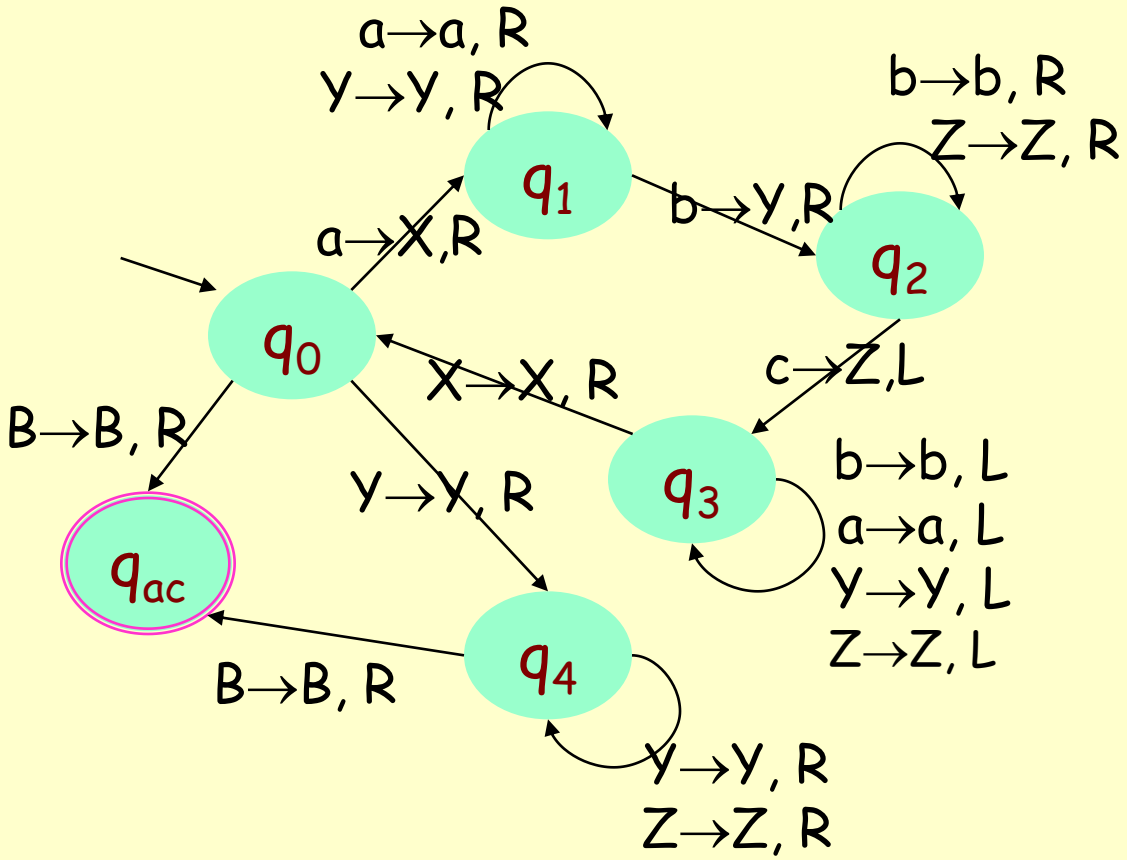
Pertence à L: aaabbbccc

Não Pertence à L: aaabbcccc

A Máquina deTuring

1. $Q = \{q_0, q_1, q_2, q_3, q_4, q_{ac}\}$
2. $\Sigma = \{a, b, c\}$
3. $\Gamma = \{a, b, c, B, X, Y, Z\}$
4. δ a seguir.
5. q_0 – o estado inicial
6. $F = \{q_{ac}\}$

Ideia: em cada passo, reconhecer um a, um b e um c, substituindo-os por X, Y e Z, respectivamente.



Exercícios

1) Construir uma MT que decida se uma sequência de parênteses é bem formada.

- Dica: considere que a cadeia de parênteses é limitada por 2 A´s (um a esq. e outro à direita).
- Ideia: Procure por um) e substitua por X; em seguida, volte à esquerda procurando o (mais próximo para substituir por X também.

2) Construir uma MT tal que, dada uma cadeia w pertencente a {0,1}*, duplique w. Quando a máquina parar, a fita deve conter w#w sendo que # indica fim de w.

Exercício

Faça uma MT que reconheça L = {x | x ∈ {a,b,c}* e x é uma permutação de $a^n b^n c^n$ para algum n >= 0}

Exs.: aabbcc √ babacc ✗

bca √ aabccc ✗

cccaaabbb √ aacc ✗

Sugestão

a) trocar um a,b, ou c do começo por 1 para marcar o final à esquerda;

b) substituir um a, um b e um c por 0´s.

c) M aceita se, ao percorrer a cadeia de entrada, a fita consiste somente de 0´s.