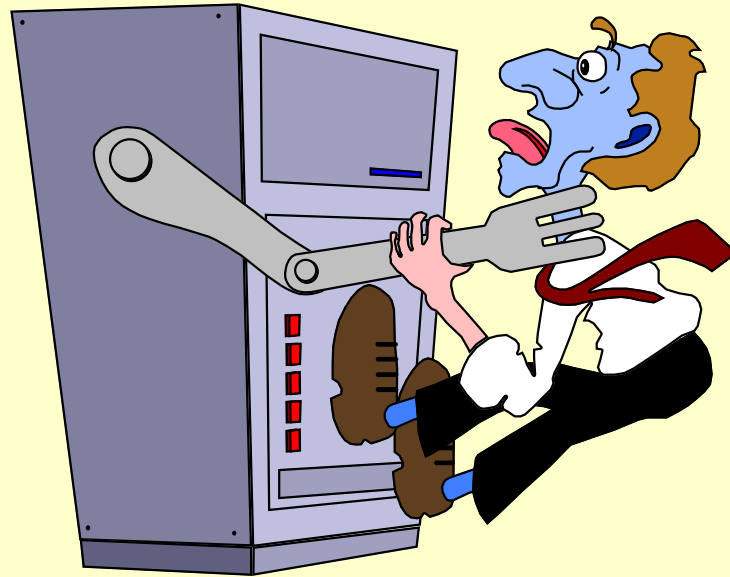


Máquinas de Turing



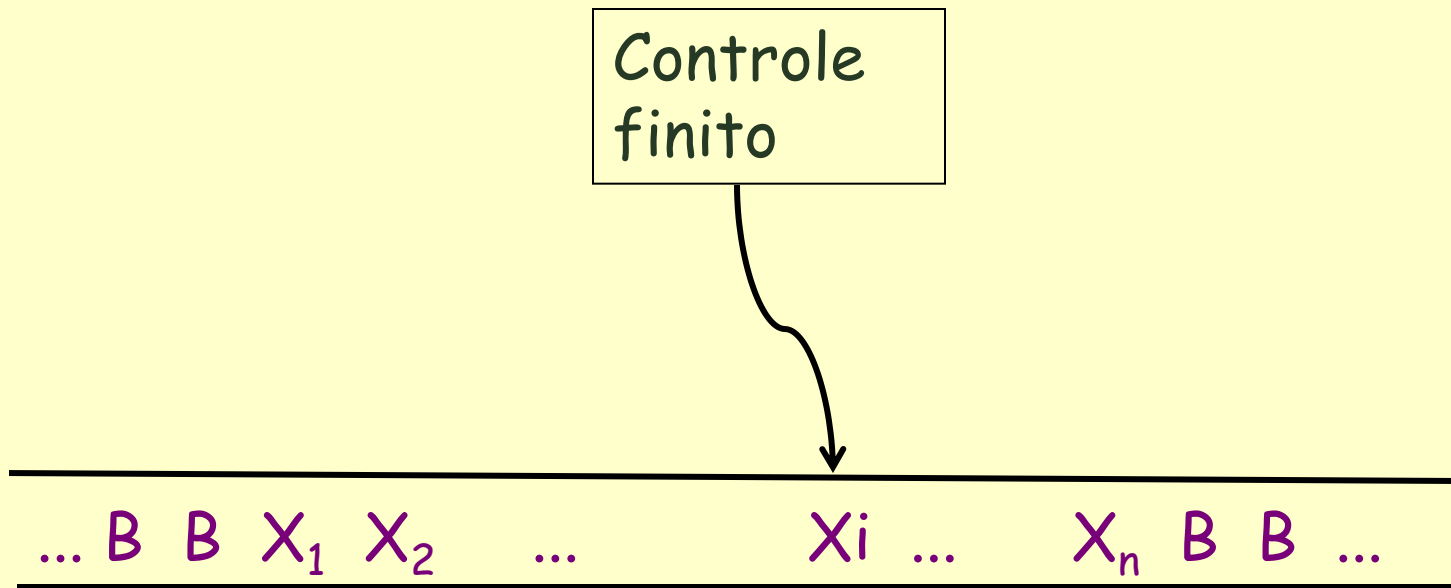
Máquinas de Turing podem fazer tudo o que um computador real faz.

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.

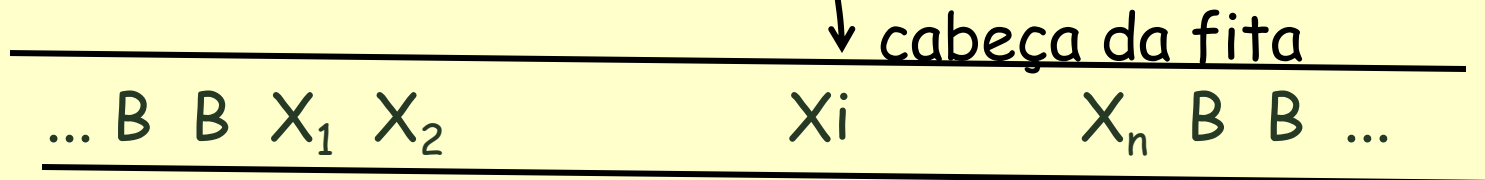
MT: notação formal

$$M = (Q, \Sigma, \Gamma, \delta, q_0, F)$$

Controle
finito

Q = conj. finito de estados;
 F = conj. estados finais (de aceitação)

Γ = alfabeto finito da fita



Σ = alfabeto finito de entrada

Máquina de Turing

Função de transição δ :

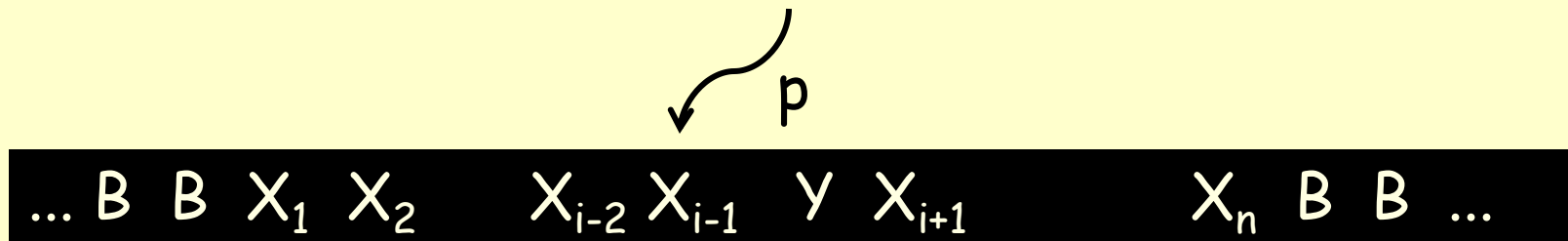
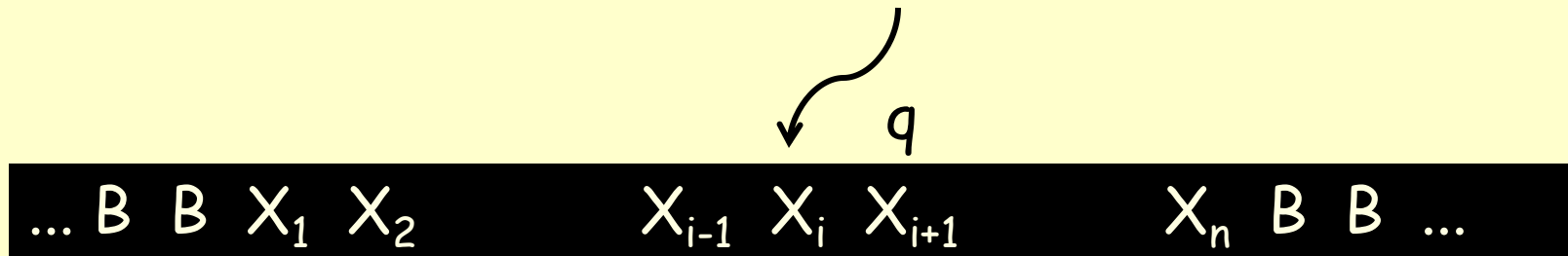
$$\delta: Q \times \Gamma \rightarrow Q \times \Gamma \times \{L,R\}$$

Ou seja, $\delta(q,X) = (p,Y,D)$ onde:

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

Descrições Instantâneas para MT

Suponha que $\delta(q, X_i) = (p, Y, L)$, ou seja, o movimento foi para a esquerda. Então:

$$X_1 X_2 \dots X_{i-1} q X_i X_{i+1} \dots X_n \mid \xrightarrow{M} X_1 X_2 \dots X_{i-2} p X_{i-1} Y X_{i+1} \dots X_n$$


Duas exceções:

- Se $i = 1$, então M se move para o B à esquerda de X_1 . Nesse caso:

$$qX_1X_2 \dots X_n \mid \text{---} pBYX_2 \dots X_n$$

- Se $i = n$ e $Y = B$, então o B gravado sobre X_n se junta ao sufixo de B s e não aparece na próxima DI:

$$X_1X_2 \dots X_{n-1}qX_n \mid \text{---} X_1X_2 \dots X_{n-2}pX_{n-1}$$

Agora suponha que $\delta(q, X_i) = (p, Y, R)$, ou seja, o movimento foi para a direita. Então:

$$X_1 X_2 \dots X_{i-1} q X_i X_{i+1} \dots X_n \mid \xrightarrow{M} X_1 X_2 \dots X_{i-1} Y p X_{i+1} \dots X_n$$

Duas exceções:

- Se $i = n$, então a $(i+1)$ -ésima célula contém um B e ela não faz parte da DI anterior. Nesse caso:

$$X_1 X_2 \dots X_{n-1} q X_n \mid \xrightarrow{M} X_1 X_2 \dots X_{n-1} Y p B$$

- Se $i = 1$ e $Y = B$, então o B gravado sobre X_1 se junta ao prefixo de Bs e não aparece na próxima DI:

$$q X_1 X_2 \dots X_n \mid \xrightarrow{M} p X_2 \dots X_n$$

Exemplo

Vamos projetar uma MT para reconhecer

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

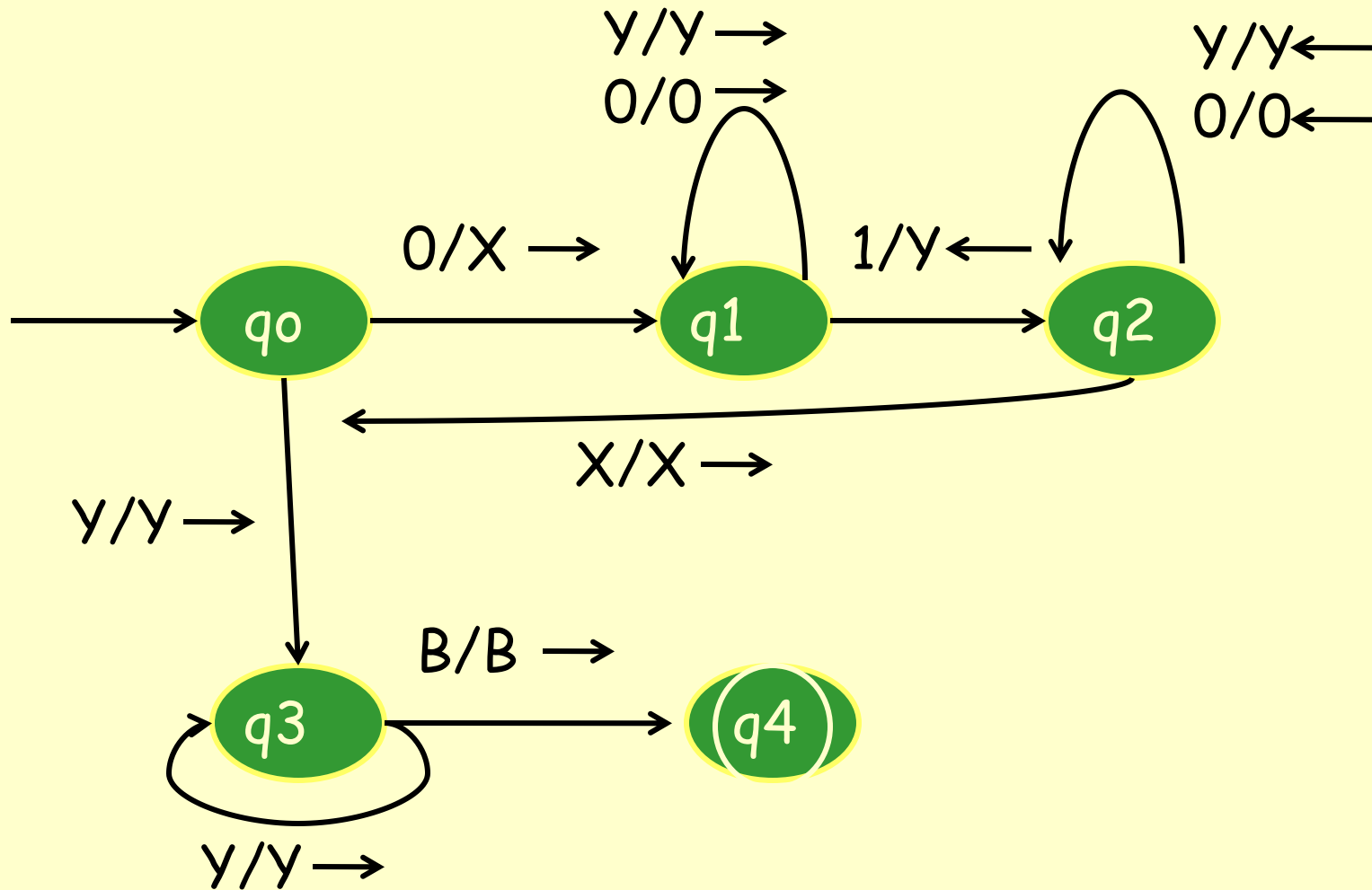
- **Estratégia:** a MT trocará um 0 por um X, e depois um 1 por um Y, até todos os 0s e 1s terem sido comparados.
- Em cada passo, da esq. para dir., ela troca um 0 por X e vai para a direita, ignorando 0s e Ys até encontrar 1. Troca esse 1 por Y e se move para a esquerda, ignorando Ys e 0s, até encontrar um X. Procura um 0 a direita e troca por X, repetindo o processo.
- Se a entrada não estiver em $0^n 1^n$ eventualmente a MT não vai ter um movimento previsto e vai parar sem aceitar.
- Se, por outro lado, na busca por mais um 0, ela só encontrar Xs e Ys, então ela descobre que deve aceitar a entrada.

$$M = (\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \{0, 1, X, Y, B\}, \delta, q_0, \{q_4\})$$

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 *	--	--	--	--	--

Verifique se a cadeia 000111 é aceita

Diagrama de Transição



Exercício

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

Estágios para a resolução:

- 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.

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 entrar eventualmente num estado de aceitação, a entrada será aceita; caso contrário, não.
- **Formalmente:** seja $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ uma MT. Então $L(M)$ é o conjunto de cadeias w em Σ^* tais que $q_0 w \xrightarrow{*} \alpha p \beta$ para algum estado p em F e quaisquer cadeias de fita α e β . **(aceitação por estado final)**

A linguagem de uma MT

- As linguagens aceitas por MT são também chamadas de *linguagens recursivamente enumeráveis (RE)*

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 **para** se ela entra em um estado **q** , olhando um símbolo de fita **X** , e não existe mais nenhum movimento previsto nessa situação, i.e., **$\delta(q, X)$ é indefinido**.

Usos de uma MT

- como reconhecedor de linguagens (Visto)
- para calcular funções

MT como um processador de funções inteiras

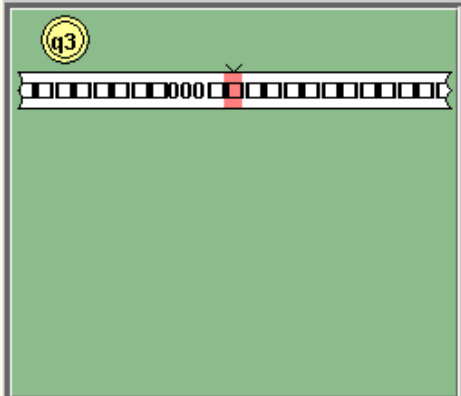
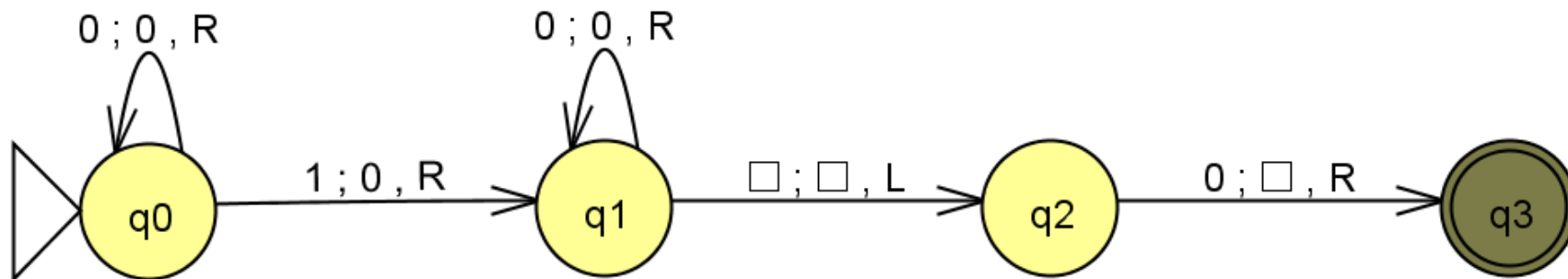
- Tradicionalmente, os inteiros são representados em vocabulário unário.
- O inteiro $i \geq 0$ é representado pela cadeia 0^i .
- Se a função tem k argumentos (i_1, i_2, \dots, i_k) então esses inteiros são colocados na fita separados por 1's como:

$0^{i_1} \ 1 \ 0^{i_2} \ 1 \ \dots \ 1 \ 0^{i_k}$

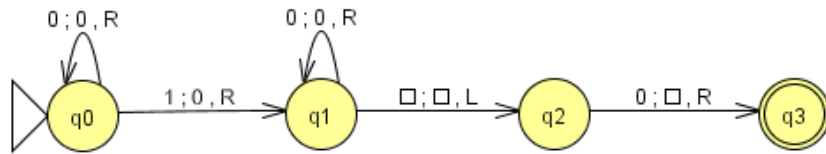
- O inverso também é possível.
- Se a máquina para (não importa se num estado final) com a fita consistindo de 0^m para algum m então dizemos que $f(i_1, i_2, \dots, i_k) = m$, onde f é uma função de k argumentos computados por essa MT.

Exemplo: MT que soma dois números naturais

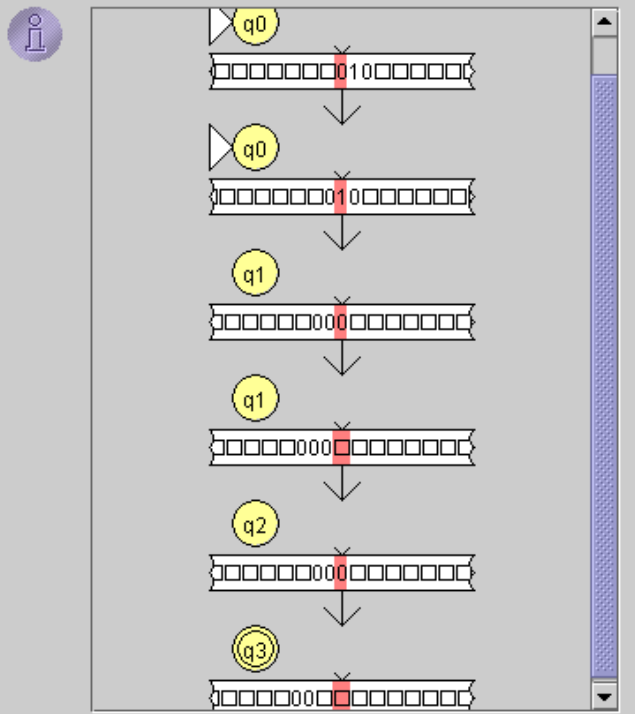
- Conteúdo inicial da Fita: ...B 0^a 1 0^b B...
- Quando a MT parar, o conteúdo da fita dever ser:
...B 0^{a+b} B....
- **Processo:**
 - Ler o 0 mais à esquerda, mantendo-o como 0, e mover à direita até encontrar o 1.
 - Substitua o 1 por 0 (nesse momento a cadeia da fita é 0^{a+b+1}). Continue movendo à direita sem mudar a fita, até que um B seja encontrado.
 - Mantenha o B e mova à esquerda para encontrar o último 0 mais a direita.
 - Substitua esse 0 por B. O resultado é 0^{a+b}



Editor



Accepting configuration found!



Keep looking I'm done

Exercício

- Projete uma MT que calcule, para dois inteiros positivos m e n , $m \dot{-} n$, chamada *monus* ou *subtração própria*, e definida por:

$m \dot{-} n = \max(m-n, 0)$. Isto é,

$m \dot{-} n = m-n$, se $m \geq n$

$= 0$, se $m < n$

início

...BB000....0100.....0BB...

0^m

0^n

final

...BB000....000.....0BB...

0^{m-n}

- F é vazio se a MT é transformadora de uma cadeia de entrada em uma cadeia de saída, isto é, como um modelo para **descrever procedimentos** (ou computar funções).
- F é relevante quando a MT é usada para reconhecer uma linguagem.

Exercícios

- 1) Construir uma MT que decida se uma sequência de parênteses é bem formada.
 - Escreve 0 se mal formada
 - Escreve 1 se bem formada
 - Dica: considere que a cadeia de parênteses é limitada por 2 A's (um a esq e outra à direita).
 - Ideia: Procurem por um $)$ e substitua por X e em seguida voltar a esquerda procurando o $($ (mais próximo para substituir por X também).
- 2) Construir uma MT tal que, dada uma cadeia w pertencente ao fecho de $\{0,1\}$, duplique w . Quando a máquina parar, a fita deve conter $w\#w$ sendo que $\#$ indica fim de w .

Exercícios

3. Faça uma MT que reconheça $L = \{0^{2^n} \mid n \geq 0\}$ **cadeias de 0 cujo tamanho é potência de 2**

4. Faça uma MT que reconheça $L = \{x \mid x \in \{a,b,c\}^* \text{ e } x \text{ é uma permutação de } a^n b^n c^n \text{ para algum } n \geq 0\}$

ex. aabbcc bca cccaabbb

Comentários sobre os Exercícios

4: (Na verdade, um ALL)

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.

Comentários sobre os Exercícios

3: estágios para a resolução:

0. Marque o primeiro zero com Y

1. Atravesse da esquerda para direita marcando um zero sim outro não com um X

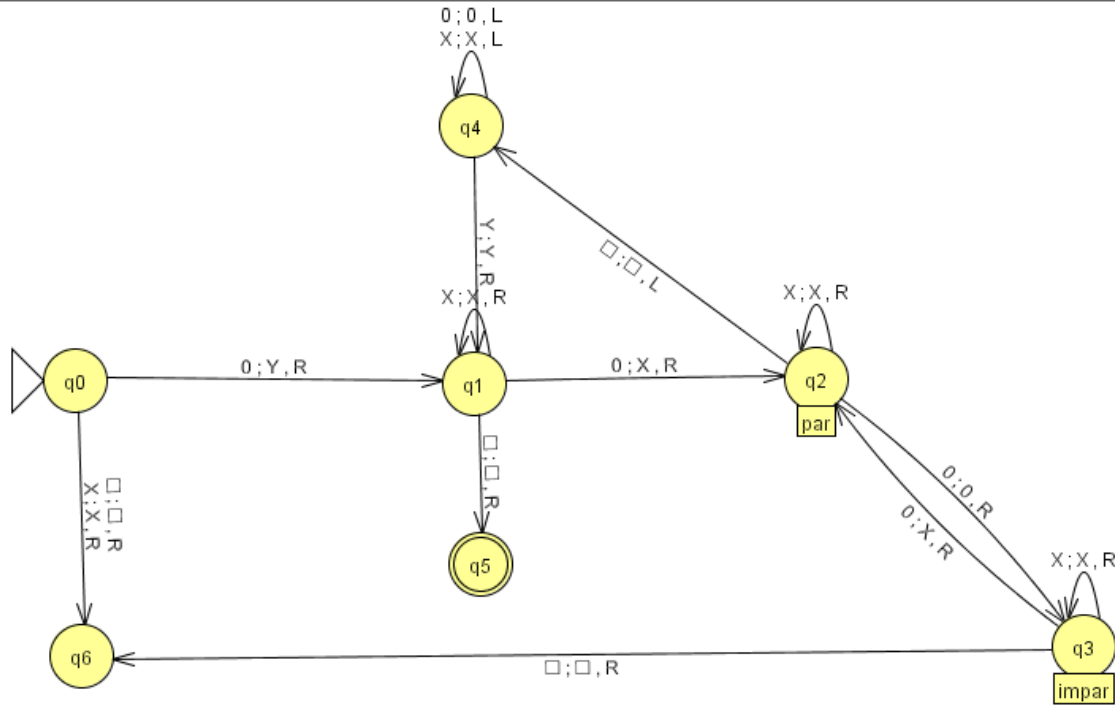
2. Se a fita contiver 1 único 0, aceite. Se contiver mais do que 1 zero e o número for ímpar, rejeite.

3. Retorne ao marcador Y

4. Recomece no estágio 1.



3



Máquinas de Turing: Poder Computacional e Teoremas

- A fim de demonstrar, empiricamente, o poder computacional das MT, existem técnicas de combinação e extensão das MT. Dessa forma, o modelo de Turing vai ganhando recursos que facilitam a construção de MT para funções mais complexas.
- O ponto importante, no entanto, é que todas essas extensões comprovadamente NÃO alteram o conjunto de funções computáveis por MT.
- Isto é, o poder computacional das MT permanece inalterado com o acréscimo de todos esses recursos.
- Isso traz evidências para a *Tese de Church-Turing: uma função é computável se e somente se ela for MT-computável.*

Técnicas de Extensão de MT

- Permitir armazenamento no controle finito
- Permitir que a fita tenha várias trilhas;
- Permitir várias fitas e cabeças de leitura/escrita;
- Permitir não-determinismo na definição de δ_i ;
- Permitir a combinação de quaisquer extensões acima.

Armazenamento no controle finito

- Uma quantidade finita de informação pode ser armazenada no controle finito.
- Podemos pensar na MT como tendo um nro. fixo de registradores X_1, X_2, \dots, X_n , cada um com um nro. fixo de bits.
- a cada movimento, tanto lendo quanto escrevendo na fita, a MT pode ler e escrever um registrador.

Armazenamento no controle finito

- Ex.: quando se quer comparar 2 cadeias x e y , símbolo a símbolo, devemos "lembrar" do símbolo atual da cadeia que está sendo verificada, x , até deslocarmos a cabeça até o símbolo correspondente de y . Essa "lembrança" pode se dar na forma de um registrador.
- Repare que essa "memória" pode ser simulada por meio de diferentes estados.

Fita com múltiplas trilhas

- Considere a MT que reconhece $\{0^n 1^n \mid n \geq 0\}$, onde trocávamos 0 por X e 1 por Y.
- Podemos pensar na fita contendo 2 trilhas:

..... B v v B B v v B.....

..... B 0 0 0 0 1 1 1 1 B.....

- A inferior contém 0, 1 e B; a superior contém B ou v.
- Podemos generalizar para permitir qualquer conjunto finito de símbolos.

Fita com múltiplas trilhas

- Assim, uma dada MT com um número fixo n de trilhas em sua fita, para $1 \leq i \leq n$, a i -ésima trilha terá seu próprio alfabeto A_i ($B \in A_i$).
- Um subconjunto de trilhas será de entrada, e possivelmente um subconjunto será de saída.
- Se a i -ésima trilha é de entrada, então ela terá seu próprio alfabeto, V_i , tal que $V_i \subseteq A_i$ e $B \notin V_i$.
- A entrada (conj. ordenado de cadeias) é colocada nas trilhas de entrada e, se a máquina parar, a saída é o conjunto ordenado de cadeias obtidas nas trilhas de saída.

Fita com múltiplas trilhas

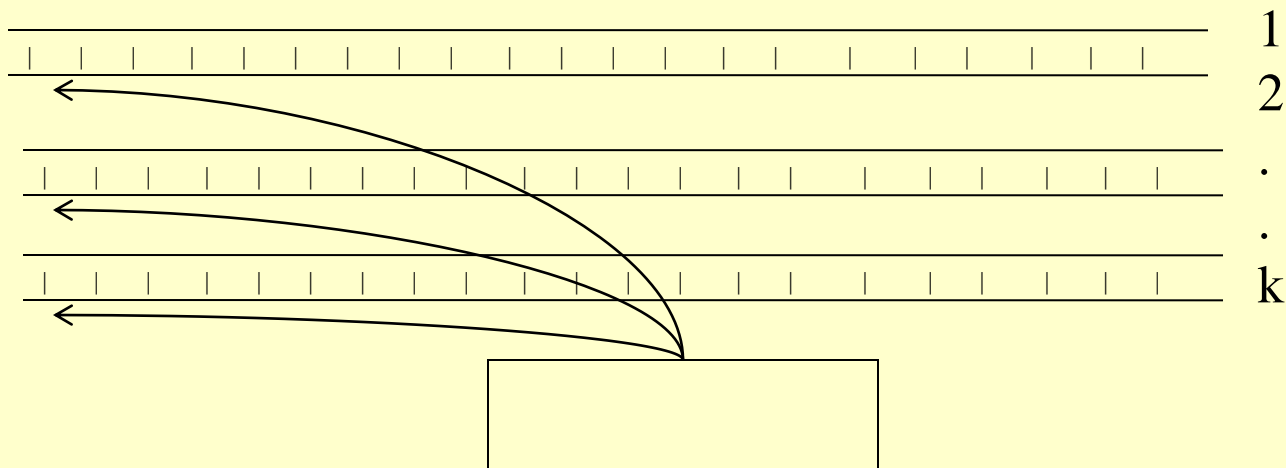
- A cada estado da máquina está associado um subconjunto não vazio de trilhas.
- A ideia é que, se o estado s está associado a um subconjunto de k trilhas, quando a MT está no estado s , ela lê uma k -upla de símbolos dessas trilhas e substitui esses símbolos por outra k -upla. As outras trilhas não são alteradas. Se não especificarmos um subconjunto para s , então s fica associado a todas as n trilhas.
- Se forem reconhecedoras, é necessário especificar o subconjunto F , de estados finais.

Subrotinas

- Subrotinas podem ser simuladas por MT com memória no controle finito e múltiplas trilhas na fita.
- O estado de retorno pode ser "lembrado" no controle finito e a célula da fita para a qual irá retornar pode ser marcada numa trilha.
- As trilhas fariam o papel de memória em separado, onde seriam executadas as ações da subrotina, sem conflitar com o "status do programa principal".

Máquinas com Múltiplas Fitas

- **Def.** Uma MT com k fitas consiste de um controle finito e k fitas de trabalho, cada uma conectada com o controle finito por meio de uma cabeça de fita.
- A função de transição para esta máquina, com base no estado atual, escolhe uma fita para ler, escrever e mover à esquerda ou à direita desta fita.
- Assim como antes, a máquina pára quando sua função de transição for indefinida.



Máquinas com Múltiplas Fitas

- A entrada para a MT é colocada em algumas fitas, designadas de entrada; e a saída, se houver, é obtida de algumas fitas designadas de saída.
- Se a MT for reconhecedora, um subconjunto de estados é designado como final.
- Exemplo: uma MT de 4 fitas que reconhece $\{x \mid x \text{ é uma permutação de } a^n b^n c^n \ n \geq 0\}$

- **Processo:** usar fita 1 como entrada; a MT conta o número de a's, b's e c's e os coloca nas fitas 2, 3 e 4 respectivamente, representados por cadeias de 1's, limitadas à esquerda por um 0.

a b a c c b

0 1 1

0 1 1

0 1 1

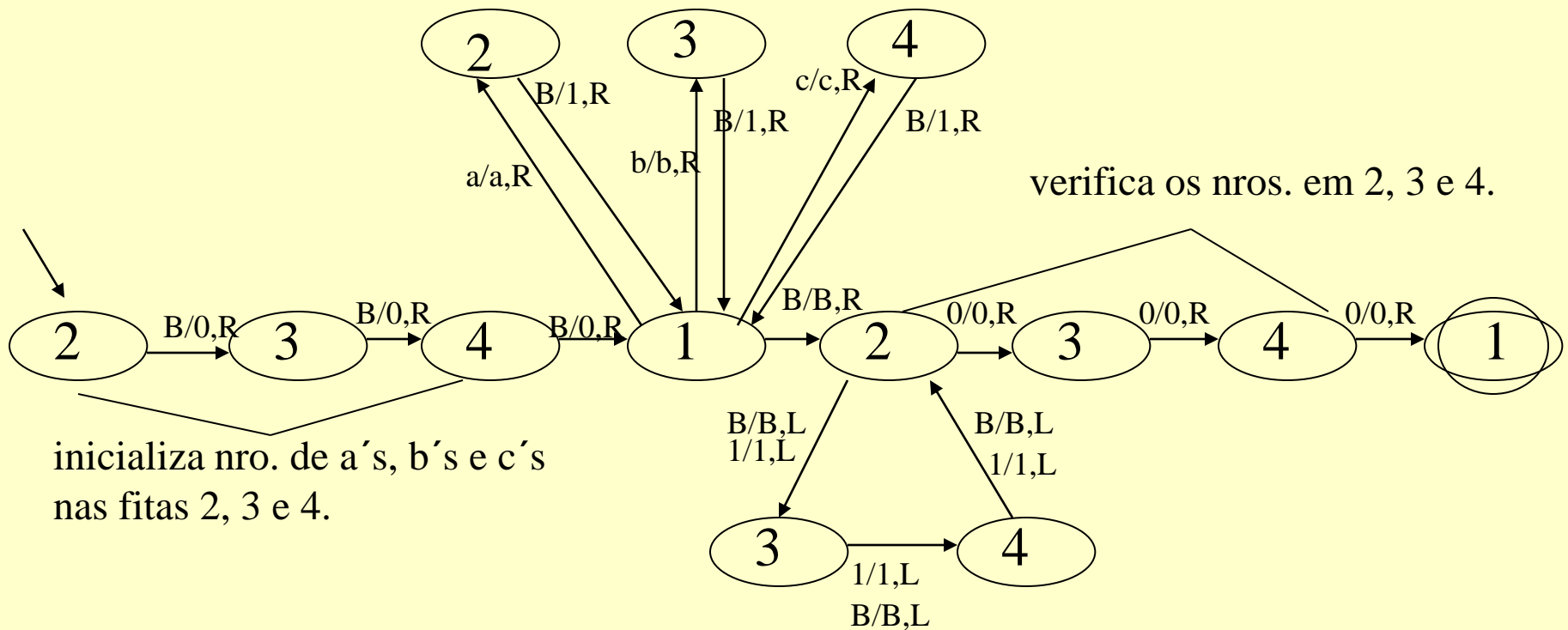
Fita 1 (entrada)

Fita 2 (# a)

Fita 3 (# b)

Fita 4 (# c)

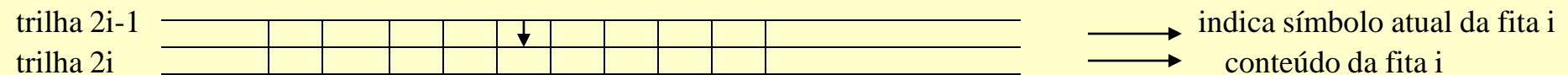
Diagrama de transição: como anterior, e o número da fita substitui o nome do estado.



Teorema

Toda função computável por uma MT com k fitas é computável por uma MT de uma fita.

Prova: mostrar como simular qualquer MT de múltiplas fitas numa MT de fita única com $2k$ trilhas.



As MT com k fitas são **polinomialmente** equivalentes às MT com uma fita.

MT Não Determinísticas

- Uma MTND difere da determinística pelo fato de que, para cada estado q e símbolo de fita X , $\delta(q, X)$ é um conjunto de triplas $\{(q_1, Y_1, D_1), \dots, (q_k, Y_k, D_k)\}$
- A linguagem aceita por uma MTND é dada pelo conjunto de cadeias para as quais haja ao menos uma sequência de escolhas de movimentos que leve da DI inicial a uma DI com um estado final.

- **Teorema:** Se M_N é uma MTND, então existe uma MT determinística M_D tal que $L(M_N) = L(M_D)$.
- Ou seja, as MTND não aceitam nenhuma linguagem não aceita por uma MTD.
- As MTND são equivalentes às MTD porém com uma perda **exponencial** de eficiência.

MT e os Computadores

- Aceitam o mesmo conjunto de linguagens - as linguagens recursivamente enumeráveis.
- Mostra-se isso simulando-se uma MT por um programa de computador (fácil) e simulando-se um computador por uma MT (de várias fitas - cada fita um recurso: memória, contador de instruções, arquivo de E/S, etc.).

Tempos de Execução

- A questão do tempo de execução é importante, pois a MT é usada não apenas para determinar o que pode ser calculado, mas o que pode ser calculado com eficiência suficiente para ser usado na prática.
- A linha divisória entre os problemas *tratáveis* (computados com eficiência) e os *intratáveis* (computados, mas em tempo inaceitável) em geral é considerada entre o que pode ser calculado em tempo *polinomial* e o que exige mais que qualquer tempo de execução *polinomial* (p.ex. *exponencial*, *fatorial*).

Tempos de Execução

- Se um problema pode ser resolvido em **tempo polinomial** em um computador típico, então ele pode ser resolvido em **tempo polinomial** por uma MT e vice-versa.
- Devido a essa **equivalência polinomial**, tudo o que se conclui sobre o que uma MT pode ou não pode fazer com eficiência adequada se aplica igualmente bem a um computador.

Hierarquia das Classes de Máquinas e Linguagens

L Recursivamente Enumeráveis/Máquinas de Turing que Reconhecem L

L Recursivas/Máquinas de Turing que Decidem L

L Livres de Contexto/Máquinas a Pilha não Determinísticas

L Livres de Contexto Determinísticas/
Máquinas a Pilha Determinísticas

L Regulares/Autômatos Finitos