

# 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  $\delta$ ;
- 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  $y$ .
- 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 idéia 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.

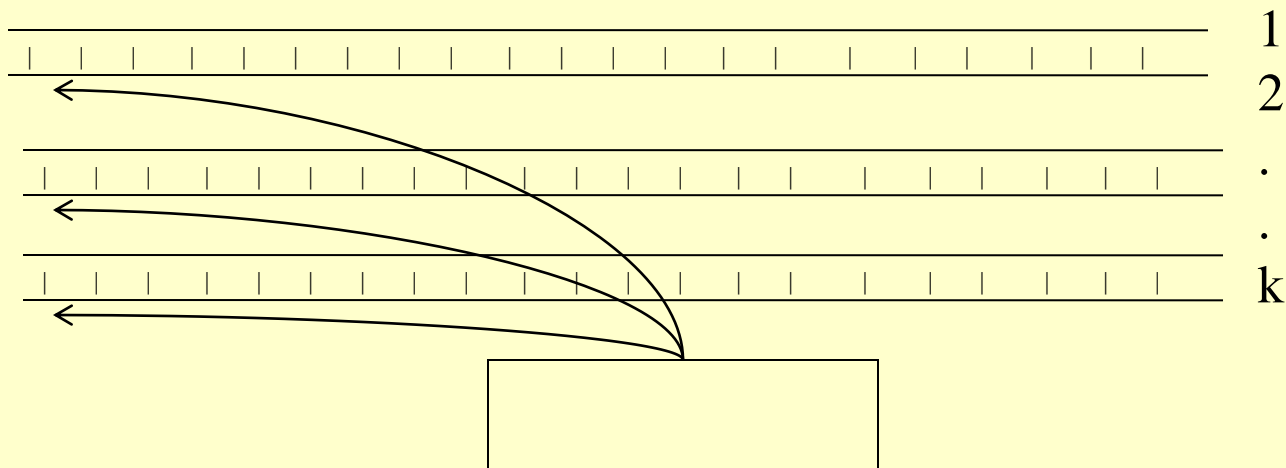


# Subrotinas

- 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 é indefinida.



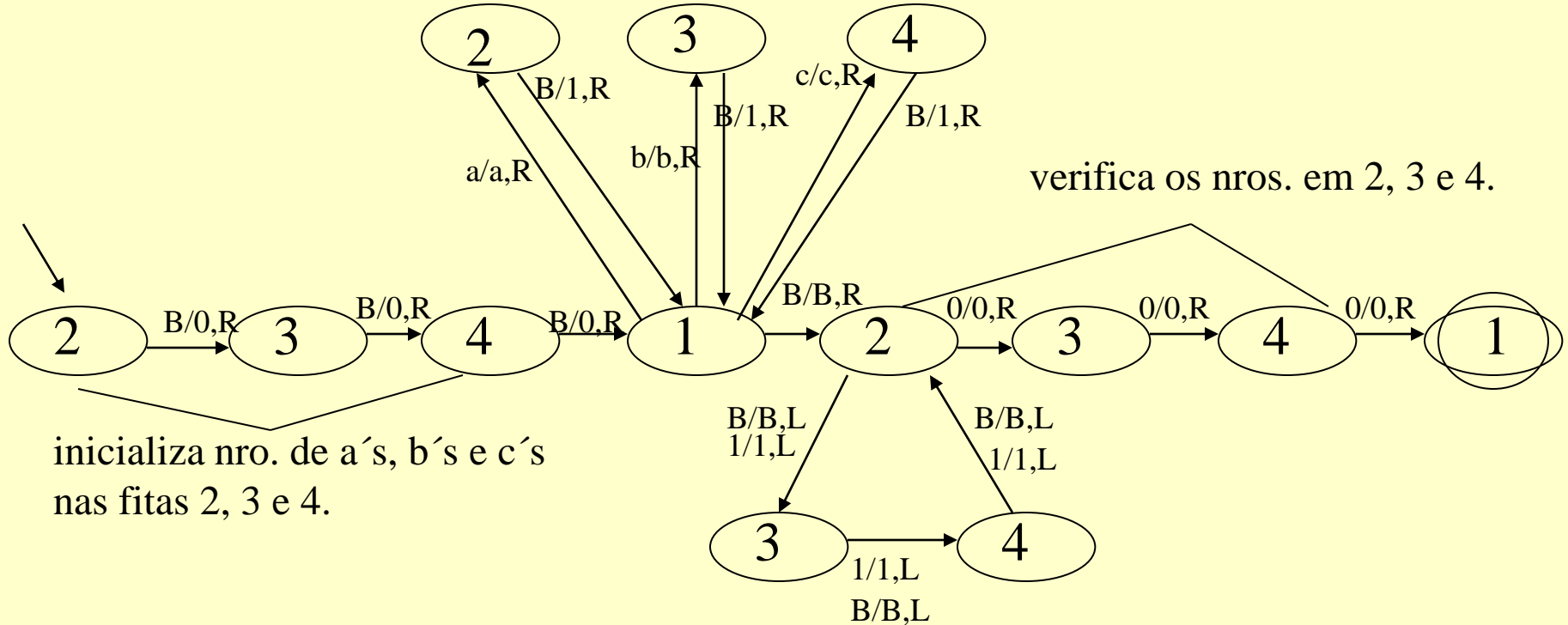
# Máquinas com Múltiplas Fitas

- 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 inferiormente por um 0.

|             |                  |
|-------------|------------------|
| a b a c c b | Fita 1 (entrada) |
| 0 1 1       | Fita 2 (# a)     |
| 0 1 1       | Fita 3 (# b)     |
| 0 1 1       | Fita 4 (# c)     |

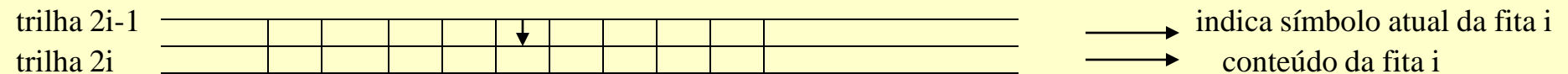
Diagrama de transição: como anterior, e o número da fita substitui o nome do estado.



# 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.



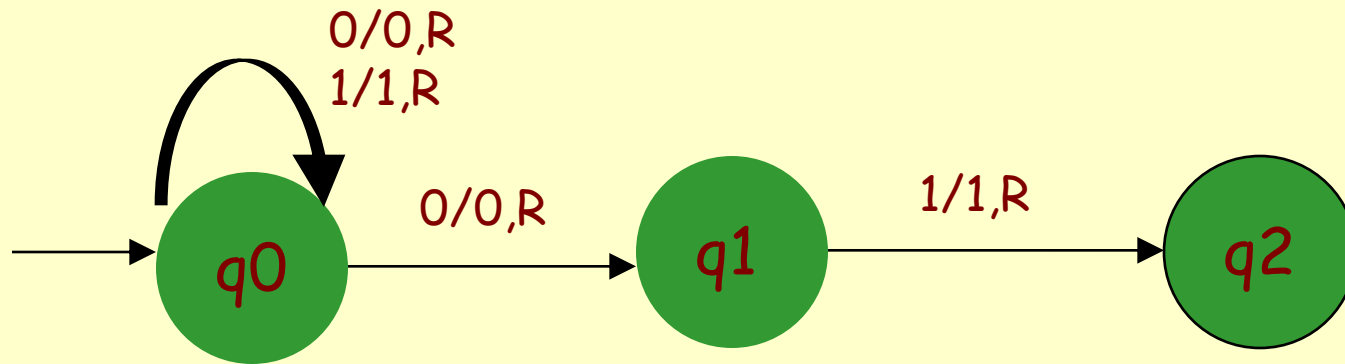
As MT com  $k$  fitas são **polinomialmente** equivalentes às MT com uma fita.

# MT Não-Determinísticas: MTND

- Numa MTND, dado um estado e um símbolo, é possível fazer mais de uma ação, ou seja:

$$\delta(q, X) = \{(q_1, Y_1, D_1), \dots, (q_k, Y_k, D_k)\}$$

- É equivalente dizer que pode haver mais do que um caminho possível, a partir de um certo ponto.



Esse autômato aceita cadeias de 0's e 1's que terminam em 01.  
 Analisem a aceitação de "00101"

Quando está em  $q_0$  e o símbolo lido é 0 ele tem a opção de:  
 continuar em  $q_0$  no caso do fim da cadeia não estar  
 próximo

OU

ir para  $q_1$  porque aposta que o fim está chegando.

**E na verdade ele executa as duas opções!**

Por isso costumamos pensar que ele "advinha" qual é a  
 alternativa correta entre muitas.



# MTND

- 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 do estado inicial a um estado final**. Ou seja, se por um dos caminhos possíveis, terminar em estado final, então a cadeia é aceita.
- É como se a MTND tentasse, em paralelo, reconhecer a cadeia, por diferentes caminhos.

- **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 – LRE.
- Mostra-se isso simulando-se uma MT por um 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).

*(\*) Linguagens que podem ter seus elementos listados (enumerados) em alguma ordem coincidem com as linguagens aceitas por alguma MT.*

# 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*.

# 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