

# Máquinas Universais

Prof. Marcus Vinícius Midená Ramos

Universidade Federal do Vale do São Francisco

20 de junho de 2013

`marcus.ramos@univasf.edu.br`  
`www.univasf.edu.br/~marcus.ramos`

- 1 *Teoria da Computação (capítulos 4, 5, 6 e 7)*  
T. A. Diverio e P. B. Menezes  
Bookman, 2011, 3ª edição
- 2 *Introduction to Automata Theory, Languages and Computation (capítulo 8)*  
J. E. Hopcroft, R. Motwani e J. D. Ullman  
Addison-Wesley, 2007, 3ª edição

# Roteiro

- 1 Introdução
- 2 Hipótese de Church
- 3 Codificação de dados estruturados
- 4 Máquina Norma
- 5 Máquina de Turing
- 6 Máquina de Post
- 7 Máquina com Pilhas
- 8 Autômato com Duas Pilhas
- 9 Variações das Máquinas de Turing

# Algoritmo

- ▶ Definição informal;
- ▶ Descrição finita e não-ambígua;
- ▶ Passos discretos, executáveis mecanicamente;
- ▶ Tempo finito;
- ▶ Restrições de ordem prática: tempo e espaço;
- ▶ Restrições de ordem teórica: tanto quanto necessário.

# Algoritmo

- ▶ Realização na forma de programa;
- ▶ Programa demanda uma máquina para sua execução;
- ▶ Características desejáveis das máquinas:
  - ▶ *Simplicidade*: Apenas características essenciais, com omissão de características não-relevantes. Permitir conclusões generalizadas sobre a classe das funções computáveis.
  - ▶ *Poder*: Representação de qualquer função computável. Simulação de qualquer outra máquina real ou teórica.

# Máquina universal

## Conceito

- ▶ Aquela que permite a representação de qualquer algoritmo na forma de um programa para a mesma;
- ▶ Evidências que permitem caracterizar uma máquina como sendo universal:
  - ▶ *Interna*: Quaisquer extensões ou variações não aumentam o seu poder computacional (o conjunto de funções computáveis permanece inalterado).
  - ▶ *Externa*: Equivalência com outros modelos (máquinas ou não) que representam a noção de algoritmo.

# Máquina universal

## Modelos estudados

- ▶ Máquina Norma;
- ▶ Máquina de Turing;
- ▶ Máquina de Post;
- ▶ Máquina com Pilhas.

# Hipótese de Church

- ▶ Alonzo Church, 1903-1995, matemático norte-americano;
- ▶ Também conhecida como Hipótese de Church-Turing, 1936;
- ▶ Mesmo ano em que foi apresentada a Máquina de Turing;
- ▶ Estabelece a equivalência entre a noção de algoritmo e Máquina de Turing;
- ▶ Como a noção de algoritmo é informal, a hipótese não pode ser provada;
- ▶ A necessidade por uma definição formal de algoritmo é grande, pois apenas a partir dela é que é possível investigar a existência de algoritmos que resolvem (ou não) certos problemas e calculam (ou não) certas funções, além de poder demonstrar certas propriedades dos mesmos.

# Hipótese de Church

- ▶ “Qualquer função computável pode ser processada por alguma Máquina de Turing”;
- ▶ “A Máquina de Turing é o dispositivo de computação mais genérico que existe”;
- ▶ “Tudo que é computável é computável por uma Máquina de Turing”;
- ▶ “ A capacidade de computação representada pela Máquina de Turing é o limite máximo que pode ser atingido por qualquer dispositivo de computação”;
- ▶ “Qualquer outra forma de expressar algoritmos terá, no máximo, a mesma capacidade computacional da Máquina de Turing”.

# Hipótese de Church

- ▶ Ao longo das décadas, evidências internas e externas apenas reforçam a Hipótese de Church, que é aceita como verdadeira de forma praticamente generalizada e não questionada;
- ▶ A Máquina de Turing (entre outros modelos), pela sua simplicidade, passa a ser usada como definição formal de algoritmo, atendendo aos propósitos citados anteriormente.

# Algoritmos e tipos de dados

- ▶ Algoritmos manipulam, normalmente, diversos tipos de dados (inteiros positivos, negativos, racionais, reais, lógicos, cadeias de caracteres, vetores, estruturas etc);
- ▶ Com o objetivo de evitar que os modelos matemáticos abstratos se tornem (desnecessariamente) complexos, o escopo de manipulação de dados dos algoritmos que serão estudados é restrito aos números inteiros positivos;
- ▶ Essa restrição não traz maiores conseqüências, uma vez que esses e vários outros tipos de dados podem ser representados através de codificações apropriadas dos mesmos no espaço dos números inteiros não negativos.

# Função de codificação

Seja  $X$  um conjunto de dados estruturados. A função *injetora*:

$$c : X \rightarrow \mathbb{N}$$

é tal que,  $\forall x \in X, c(x)$  representa a codificação do dado estruturado  $x$ .  
Como  $c$  é injetora,

$$(c(x) = c(y)) \Rightarrow (x = y)$$

portanto a codificação representa de forma unívoca o dado estruturado  $x$  na forma de um número natural  $c(x)$ .

# Teorema fundamental da aritmética

## Enunciado

Seja  $a > 1$ . Então:

$$a = p_1^{n_1} p_2^{n_2} \dots p_k^{n_k}$$

onde:

- ▶  $p_1 < p_2 < \dots < p_k$  são números primos (não necessariamente os primeiros, não necessariamente consecutivos);
- ▶  $n_1, n_2, \dots, n_k$  são números inteiros positivos maiores ou iguais a 1;
- ▶ Essa decomposição é única, a menos de permutações.

⇒ Qualquer número inteiro maior que 1 pode ser decomposto, de forma unívoca, no produto de potências de números primos.

⇒ Números primos são a base para a definição dos demais números (números compostos).

# Teorema fundamental da aritmética

## Exemplos

- ▶  $2 = 2^1$ ;
- ▶  $17 = 17^1$ ;
- ▶  $256 = 2^8$ ;
- ▶  $143 = 11^1 \cdot 13^1$ ;
- ▶  $42706587 = 3^1 \cdot 7^6 \cdot 11^2$ ;
- ▶  $132187055 = 5^1 \cdot 7^5 \cdot 11^2 \cdot 13^1$ .

# Exemplo

## $n$ -uplas de números naturais

- ▶ Deseja-se obter  $c : \mathbb{N}^n \rightarrow \mathbb{N}$
- ▶ Teorema fundamental da aritmética;
- ▶ Considere os  $n$  primeiros números primos,  $p_1, p_2, \dots, p_n$ ;
- ▶ Então  $c(x_1, x_2, \dots, x_n) = p_1^{x_1} \cdot p_2^{x_2} \cdot \dots \cdot p_n^{x_n}$
- ▶ Todo número natural decomponível nos  $n$  primeiros números primos corresponde a uma (única)  $n$ -upla;
- ▶ Representação unívoca de  $n$ -uplas como números naturais.

### Exemplo:

- ▶  $c(1, 2, 3) = 2^1 \cdot 3^2 \cdot 5^3 = 2 \cdot 9 \cdot 125 = 2250$ ;
- ▶ 2250 representa, de forma unívoca, a tripla  $(1, 2, 3)$ .

# Exemplo

## Programas monolíticos

- ▶ Deseja-se obter  $c : \mathbb{P} \rightarrow \mathbb{N}$ , onde  $\mathbb{P}$  é o conjunto dos programas monolíticos;
- ▶ Considere que o programa  $P$  possui as operações  $O_1, O_2, \dots, O_m$  e os testes  $T_1, T_2, \dots, T_n$ ;
- ▶ Considere rótulos numéricos sequenciais, com rótulo inicial 1 e rótulo final (único) 0;
- ▶ Quádruplas representam as instruções;
- ▶ Considere que  $(0, k, r_2, r_2)$  representa a instrução  $r_1$  : faça  $O_k$  vá \_ para  $r_2$
- ▶ Considere que  $(1, k, r_2, r_3)$  representa a instrução  $r_1$  : se  $T_k$  então vá \_ para  $r_2$  senão vá \_ para  $r_3$

# Exemplo

## Programas monolíticos

- ▶ Cada instrução de  $P$  é codificada na forma de uma quádrupla;
- ▶ Cada quádrupla é codificada na forma de um número inteiro;
- ▶ Se  $P$  contém  $t$  instruções, serão geradas  $t$  quádruplas e, conseqüentemente,  $t$  números inteiros;
- ▶ Considere a  $t$ -upla formada por esses  $t$  números inteiros;
- ▶ Codifique a  $t$ -upla como um número inteiro.

# Exemplo

## Programas monolíticos

Considere o programa monolítico  $P$ :

1: se  $T_1$  vá para 2 senão vá para 0

2: faça  $O_1$  vá para 1

- ▶  $(1, 1, 2, 0)$  representa a instrução associada ao rótulo 1;
- ▶  $(0, 1, 1, 1)$  representa a instrução associada ao rótulo 2;
- ▶  $c(1, 1, 2, 0) = 2^1 \cdot 3^1 \cdot 5^2 \cdot 7^0 = 150$ ;
- ▶  $c(0, 1, 1, 1) = 2^0 \cdot 3^1 \cdot 5^1 \cdot 7^1 = 105$ ;
- ▶ Considere  $(150, 105)$  como a representação de  $P$ ;
- ▶  $c(150, 105) = 2^{150} \cdot 3^{105}$
- ▶ O número  $2^{150} \cdot 3^{105}$  representa  $P$ .

# Exemplo

## Programas monolíticos

Genericamente, se  $w$  representa um programa monolítico  $P$  com  $t$  instruções, então:

- ▶  $w = 2^{i_1}.3^{i_2}.5^{i_3} \dots p_t^{i_t}$
- ▶  $\forall j, 1 \leq j \leq t$ :
  - ▶  $i_j = 2^a.3^b.5^c.7^d$
  - ▶ Se  $a = 0$ ,  $i_j$  representa a instrução:  
 $r_j$ : faça  $O_b$  vá para  $r_c$
  - ▶ Se  $a = 1$ ,  $i_j$  representa a instrução:  
 $r_j$ : se  $T_b$  então vá para  $r_c$  senão vá para  $r_d$

# Generalidades

- ▶ Definida por Richard Bird em 1976;
- ▶ Number Theoretic Register MAchine (e, também, o nome da esposa dele...);
- ▶ É uma máquina de registradores (possui uma quantidade ilimitada deles);
- ▶ Arquitetura semelhante à dos computadores modernos;
- ▶ Cada registrador armazena um único número natural;
- ▶ Operações e testes (para cada registrador):
  - ▶ Adicionar o valor 1;
  - ▶ Subtrair o valor 1 (se 0, continua com 0);
  - ▶ Testar se o conteúdo é 0.
- ▶ Máquina Universal.

# Definição

Norma =  $(\mathbb{N}^\infty, \mathbb{N}, \mathbb{N}, ent, sai, \{add_k, sub_k | k \geq 0\}, \{zero_k | k \geq 0\})$

- ▶ Os registradores são denotados  $A, B, \dots, X, Y$ ;
- ▶  $A(k = 0), B(k = 1), \dots$ ;
- ▶  $ent : \mathbb{N} \rightarrow \mathbb{N}^\infty$ , transfere o valor da entrada para  $X$  e zera os demais registradores;
- ▶  $sai : \mathbb{N}^\infty \rightarrow \mathbb{N}$ , transfere o valor de  $Y$  para a saída;

## Definição

Norma =  $(\mathbb{N}^\infty, \mathbb{N}, \mathbb{N}, ent, sai, \{add_k, sub_k | k \geq 0\}, \{zero_k | k \geq 0\})$

- ▶  $add_k : \mathbb{N}^\infty \rightarrow \mathbb{N}^\infty$ , adiciona 1 ao  $k$ -ésimo registrador, mantendo os demais inalterados;
- ▶  $sub_k : \mathbb{N}^\infty \rightarrow \mathbb{N}^\infty$ , subtrai 1 do  $k$ -ésimo registrador, mantendo os demais inalterados; se 0, mantém 0;
- ▶  $zero_k : \mathbb{N}^\infty \rightarrow \{\text{verdadeiro}, \text{falso}\}$ , retorna *verdadeiro* se o conteúdo do  $k$ -ésimo registrador é 0, *falso* caso contrário;
- ▶ Notação:  $K := K + 1, K := K - 1, K = 0$

# Evidências internas

- ▶ Operações e testes;
- ▶ Tipos de dados;
- ▶ Agregados;
- ▶ Endereçamento indireto;
- ▶ Recursão.

# Operações e testes

Definições incrementais, através da expansão sucessiva do repertório de operações e testes da Máquina Norma.

- ▶ Atribuição do valor 0 a um registrador;
- ▶ Atribuição de um valor qualquer a um registrador;
- ▶ Adição de dois registradores;
- ▶ Atribuição de registrador à registrador;
- ▶ Multiplicação de dois registradores;
- ▶ Operador relacional menor;
- ▶ Teste de divisibilidade;
- ▶ Teste se o valor de um registrador é primo;
- ▶ Atribuição do  $n$ -ésimo número primo a um registrador.

# Operações e testes

## Atribuição do valor 0 a um registrador

Denotado:

$$A := 0$$

para o registrador  $A$ .

- ▶ Operação implementada através do programa iterativo:  
até  $A = 0$   
faça  $A := A - 1$
- ▶ Considerada como *macro*,  $A := 0$  representa uma nova operação.

# Operações e testes

Atribuição de um valor qualquer a um registrador

Denotado:

$$A := n$$

para o registrador  $A$ .

- ▶ Operação implementada através do programa iterativo, com  $n$  repetições da operação  $A := A + 1$ :  
 $A := 0$   
 $A := A + 1$   
 $A := A + 1$   
...  
 $A := A + 1$
- ▶ Considerada como *macro*,  $A := n$  representa uma nova operação.

# Operações e testes

## Adição de dois registradores

Denotado:

$$A := A + B$$

para os registradores  $A$  e  $B$ .

- ▶ Operação implementada através do programa iterativo:  
até  $B = 0$   
faça  $(A := A + 1; B := B - 1)$
- ▶ O registrador  $B$  é zerado;
- ▶ Para preservar o valor de  $B$ , deve-se usar um registrador auxiliar;
- ▶ Considerada como *macro*,  $A := A + B$  representa uma nova operação.

# Operações e testes

## Adição de dois registradores

Denotado:

$$A := A + B \text{ usando } C$$

para os registradores  $A$  e  $B$ , empregando  $C$  como auxiliar.

- ▶ Operação implementada através do programa iterativo:

$C := 0$

até  $B = 0$

faça  $(A := A + 1; C := C + 1; B := B - 1);$

até  $C = 0$

faça  $(B := B + 1; C := C - 1)$

- ▶ O registrador  $C$  é zerado;
- ▶ O identificação explícita do registrador  $C$  serve para evitar conflitos no uso do mesmo;
- ▶ Considerada como *macro*, “ $A := A + B$  usando  $C$ ” representa uma nova operação.

# Operações e testes

## Atribuição de registrador à registrador

Denotado:

$$A := B \text{ ou } A := B \text{ usando } C$$

para os registradores  $A$  e  $B$ , empregando  $C$  como auxiliar.

- ▶ “ $A := B$  usando  $C$ ” denota:

$$A := 0$$

$$A := A + B \text{ usando } C$$

ou seja,  $B$  permanece inalterado após a atribuição.

- ▶ “ $A := B$ ” denota:

$$A := 0$$

$$A := A + B$$

ou seja,  $B$  é zerado após a atribuição.

- ▶ Consideradas como *macros*, “ $A := B$ ” e “ $A := B$  usando  $C$ ” representam novas operações;

# Operações e testes

## Multiplicação de dois registradores

Denotado:

$$A := A * B \text{ usando } C, D$$

para os registradores  $A$  e  $B$ , empregando  $C$  e  $D$  como auxiliares.

- ▶ Operação implementada através do programa iterativo:

$C := 0$

até  $A = 0$

faça  $(C := C + 1; A := A - 1)$

até  $(C = 0)$

faça  $(A := A + B \text{ usando } D; C := C - 1)$

- ▶ Considerada como *macro*, “ $A := A * B$  usando  $C, D$ ” representa uma nova operação.

# Operações e testes

## Operador relacional menor

Denotado:

$$A < B \text{ usando } C, D, E$$

- ▶ Operação implementada através do programa iterativo:

$C := A$  usando  $E$ ;

$D := B$  usando  $E$ ;

até  $C = 0$

faça (se  $D = 0$

então FALSO

senão  $C := C - 1; D := D - 1$ );

se  $D = 0$

então FALSO

senão VERDADEIRO

# Operações e testes

## Teste de divisibilidade

Denotado:

$\text{teste\_mod}(A, B)$  usando  $C, D, E, C', D', E'$

- ▶ Operação implementada através do programa iterativo:

$C := A$  usando  $E$ ;

$D := B$  usando  $E$ ;

(se  $B = 0$

então FALSO

senão (se  $A = 0$

então VERDADEIRO

senão até  $C < B$  usando  $C', D', E'$

faça ( $C := C - D$  usando  $E$ );

(se  $C = 0$

então VERDADEIRO

senão FALSO)))

# Operações e testes

Teste se o valor de um registrador é primo

Denotado:

$\text{teste\_primo}(A)$  usando  $C, D$

para o registrador  $A$ , empregando  $C$  como auxiliar.

- ▶ Operação implementada através do programa iterativo:
  - se  $A = 0$  então FALSO senão
    - $C := A$  usando  $D$ ;
    - $C := C - 1$ ;
    - se  $C = 0$  então FALSO senão
      - até teste\_mod ( $A, C$ )
      - faça  $C := C - 1$ ;
      - $C := C - 1$
      - se  $C = 0$  então VERDADEIRO senão FALSO

# Operações e testes

Atribuição do  $n$ -ésimo número primo a um registrador

Denotado:

$$A := \text{primo}(B) \text{ usando } D$$

para o registrador  $A$ , supondo que  $B$  contém  $n \geq 1$  e empregando  $D$  como auxiliar.

- ▶ Operação implementada através do programa iterativo:

$A := 1;$

$D := B;$

até  $D = 0$  faça

$D := D - 1;$

$A := A + 1;$

até teste\_ primo ( $A$ ) faça  $A := A + 1$

# Tipos de dados

## Números inteiros positivos e negativos

Números inteiros com sinal  $m$  podem ser representados pela dupla:

$$(s, |m|)$$

onde

- ▶  $|m|$  representa o valor absoluto de  $m$ ;
- ▶ se  $m < 0$  então  $s = 1$  senão  $s = 0$ .

A representação em Norma pode ser feita:

- ▶ Codificação de duplas, ou
- ▶ Par de registradores.

# Tipos de dados

## Números inteiros positivos e negativos

Denotado:

$$A := A + 1$$

supondo que  $A$  representa o par de registradores  $A_1$  ( $s$ ) e  $A_2$  ( $m$ ).

- ▶ Operação implementada através do programa iterativo:
  - se  $A_1 = 0$  então  $A_2 := A_2 + 1$  senão
  - $A_2 := A_2 - 1$ ;
  - se  $A_2 = 0$  então  $A_1 := A_1 - 1$  senão ✓
- ▶  $(0, 0) + 1 = (0, 1)$ ;  $(0, 1) + 1 = (0, 2)$ ;  $(1, 1) + 1 = (0, 0)$ ;  $(1, 2) + 1 = (1, 1)$
- ▶ Outras operações podem ser implementadas sem dificuldade.

# Tipos de dados

## Números racionais

Números racionais  $r = \frac{a}{b}$  podem ser representados pela dupla:

$$(a, b)$$

com  $b > 0$ . Algumas operações e testes sobre os números racionais:

- ▶ Soma:  $(a, b) + (c, d) = (a * d + b * c, b * d)$
- ▶ Subtração:  $(a, b) - (c, d) = (a * d - b * c, b * d)$
- ▶ Multiplicação:  $(a, b) * (c, d) = (a * c, b * d)$
- ▶ Divisão:  $(a, b) \div (c, d) = (a * d, b * c)$ , para  $c \neq 0$
- ▶ Igualdade:  $(a, b) = (c, d)$  se e somente se  $a * d = b * c$

# Agregados

## Vetores

- ▶ Vetores com  $n$  elementos (inclusive com  $n$  variável) podem ser representados em um único registrador, usando codificação de  $n$ -uplas;
- ▶ Suponha que o registrador  $A$  representa o vetor com os elementos  $A[1], A[2], \dots$ ,
- ▶ Indexação direta (com número natural) ou indireta (com registrador).

Algumas operações e testes sobre vetores:

- ▶ Adiciona 1 à uma posição indexada;
- ▶ Subtrai 1 de uma posição indexada;
- ▶ Testa se uma posição indexada contém o valor 0.

# Agregados

## Vetores

Suposições:

- ▶  $p_n$  representa o  $n$ -ésimo número primo;
- ▶ A macro teste\_div ( $A, C$ ), que retorna verdadeiro se  $C$  é divisor de  $A$  e falso caso contrário, é dada;
- ▶ A macro  $A := A/C$ , que retorna o resultado da divisão inteira de  $A$  por  $C$ , é dada;
- ▶ Será omitido o termo “usando” das macros já definidas.

# Agregados

## Vetores

Definição da macro:

$add_{A[n]}$  usando  $C$

Adição de uma unidade ao elemento  $n$  do vetor  $A$ , usando indexação direta.

- ▶  $C := p_n$   
 $A := A * C$
- ▶ Considere o vetor  $[4, 2, 3]$  e seja  $A = c(4, 2, 3) = 2^4 * 3^2 * 5^3 = 18000$ :
  - ▶ Para executar  $add_{A[2]}$ , basta fazer  $A = A * 3$ ; o valor resultante (54000) representa o vetor  $[4, 3, 3]$ ;
  - ▶ Para representar  $[4, 2, 3, 5]$  (acrécimo de elemento), basta fazer  $A = A * 7^5$ .

# Agregados

## Vetores

Definição da macro:

$$sub_{A[n]} \text{ usando } C$$

Subtração de uma unidade do elemento  $n$  do vetor  $A$ , usando indexação direta.

- ▶  $C := p_n$ ;  
se teste\_mod ( $A, C$ )  
então  $A := A/C$   
senão ✓
- ▶ Considere o vetor  $[4, 2, 3]$  e seja  $A = c(4, 2, 3) = 2^4 * 3^2 * 5^3 = 18000$ :
  - ▶ Para executar  $sub_{A[2]}$ , basta fazer  $A = A/3$ ; o valor resultante (6000) representa o vetor  $[4, 1, 3]$ ;
  - ▶ Para representar  $[4, 2]$  (eliminação de elemento), basta fazer  $A = A/5^3$ .

## Agregados

## Vetores

Definição da macro:

$zero_{A[n]}$  usando  $C$

Testa se o elemento  $n$  do vetor  $A$  contém o valor 0, usando indexação direta.

- ▶  $C := p_n$ ;  
se teste\_mod ( $A, C$ )  
então FALSO  
senão VERDADEIRO
- ▶ Considere o vetor  $[4, 2, 3]$  e seja  $A = c(4, 2, 3) = 2^4 * 3^2 * 5^3 = 18000$ :
  - ▶ Como teste\_mod ( $18000, 3$ ) = 0, segue que  $A[2] \neq 0$ ;
  - ▶ Como teste\_mod ( $18000, 7$ )  $\neq 0$ , segue que  $A[4] = 0$ .

# Agregados

## Vetores

Definição da macro:

$add_{A[B]}$  usando  $C$

Adição de uma unidade ao elemento do vetor  $A$ , usando indexação indireta através do registrador  $B$ .

- ▶  $C := \text{primo}(B);$   
 $A := A * C$

# Agregados

## Vetores

Definição da macro:

$sub_{A[n]}$  usando  $C$

Subtração de uma unidade do elemento do vetor  $A$ , usando indexação indireta através do registrador  $B$ .

- ▶  $C := \text{primo}(B)$ ;
- se teste\_mod( $A, C$ )
- então  $A := A/C$
- senão ✓

# Agregados

## Vetores

Definição da macro:

*zero*<sub>A[n]</sub> usando *C*

Testa se o elemento do vetor *A* contém o valor 0, usando indexação indireta através do registrador *B*.

- ▶ *C* := primo (*B*);  
se teste\_mod (*A*, *C*)  
então FALSO  
senão VERDADEIRO

# Agregados

## Máquina Norma com apenas 2 registradores

- ▶ Os registradores  $A, B, \dots$  da Máquina Norma podem ser simulados numa máquina equivalente, com apenas dois registradores, usando a representação de vetores na forma de  $n$ -uplas;
- ▶ Suponha que a máquina tenha apenas os registradores  $X$  e  $Y$ ;
- ▶ Todo o processamento de uma Máquina Norma pode ser simulado na nova máquina com apenas esses dois registradores;
- ▶ Convencionou-se que  $X[1]$  representa o registrador  $A$ ,  $X[2]$  o registrador  $B$  e assim por diante;
- ▶ As seguintes operações são definidas:
  - ▶  $add_{X[k]}$  usando  $Y$
  - ▶  $sub_{X[k]}$  usando  $Y$
  - ▶  $zero_{X[k]}$  usando  $Y$

# Agregados

## Pilhas

- ▶ Estruturas do tipo *last-in-first-out*;
- ▶ Podem ser simuladas em Máquinas Norma através de dois registradores;
- ▶ O primeiro representa o conteúdo da pilha, considerado como um vetor e conforme visto anteriormente;
- ▶ O segundo contém o número do elemento que corresponde ao topo da pilha;
- ▶ As operações abaixo podem ser definidas facilmente:
  - ▶ *empilha*
  - ▶ *desempilha*

# Endereçamento indireto

Desviar para a instrução cujo rótulo corresponde ao conteúdo de um registrador.

- ▶  $r$ : faça  $F$  vá\_ para  $A$
- ▶  $r$ : se  $T$  vá\_ para  $A$  senão vá\_ para  $B$
- ▶ “ $A$ ” e “ $B$ ” são registradores;
- ▶ Desvia para o endereço contido em “ $A$ ” (“ $B$ ”);
- ▶ A macro “ $End_A$ ” para calcula o endereço correspondente;
- ▶ “ $r$ : faça  $F$  vá\_ para  $End_A$ ”
- ▶ “ $r$ : se  $T$  vá\_ para  $End_A$  senão vá\_ para  $End_B$ ”

## Endereçamento indireto

Suponha que  $A$  contém valores  $\leq k$ . O valor de  $A$  permanece inalterado.

Macro “ $End_A$ ”:

$i$	:	se $zero_A$ então vá_ para 0 senão vá_ para $i + 1$
$i + 1$	:	faça $sub_A$ vá_ para $i + 2$
$i + 2$	:	se $zero_A$ então vá_ para $i + 3$ senão vá_ para $i + 4$
$i + 3$	:	faça $A := 1$ vá_ para 1
$i + 4$	:	faça $sub_A$ vá_ para $i + 5$
$i + 5$	:	se $zero_A$ então vá_ para $i + 6$ senão vá_ para $i + 7$
$i + 6$	:	faça $A := 2$ vá_ para 2
$i + 7$	:	faça $sub_A$ vá_ para $i + 8$
...	:	...
$i + k * 3 - 1$	:	se $zero_A$ então vá_ para $i + k * 3$ senão vá_ para $i + k * 3 + 1$
$i + k * 3$	:	faça $A := k$ vá_ para $k$
$i + k * 3 + 1$	:	faça $sub_A$ vá_ para $i + k * 3 + 2$

# Recursão

- ▶ Chamada de subprogramas e recursão podem ser simuladas em programas monolíticos com o uso do endereçamento indireto;
- ▶ Demonstração em Bird76.

# Generalidades

- ▶ Definida por Alan Turing em 1936;
- ▶ Formulada antes da construção do primeiro computador digital;
- ▶ Aceita como formalização na noção informal de algoritmo;
- ▶ Possui, no mínimo, o mesmo poder computacional de qualquer computador moderno ou outro modelo de computação;
- ▶ Incorpora o programa na sua definição.

# Conceito

Procura reproduzir uma pessoa trabalhando na solução de um problema:

- ▶ Instrumento para escrever, outro para apagar;
- ▶ Folha de papel dividida em regiões;
- ▶ Dados iniciais na folha de papel.

Durante o trabalho:

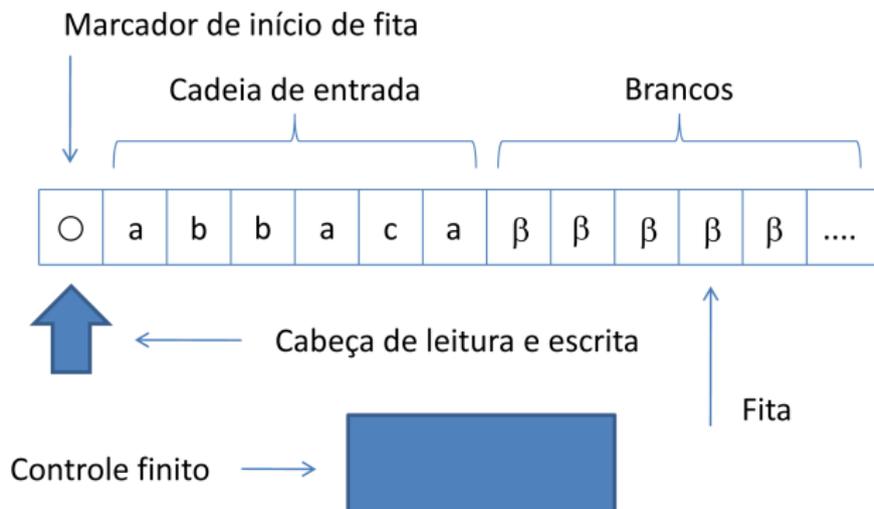
- ▶ Novo símbolo pode ser lido;
- ▶ Símbolo existente pode ser alterado;
- ▶ Olhos podem ser deslocar de região;
- ▶ Ação a ser executada depende do símbolo lido e do “estado mental” do trabalhador;
- ▶ Estados inicial e finais indicam começo e término das atividades.

# Conceito

Algumas simplificações:

- ▶ A folha de papel tem dimensões tão grandes quanto necessárias;
- ▶ Ela é organizada de forma unidimensional e dividida em células;
- ▶ O conjunto de símbolos é finito;
- ▶ O conjunto de estados mentais é finito;
- ▶ Apenas um símbolo é lido de cada vez;
- ▶ A atenção se desloca apenas para as células adjacentes.

# Componentes



# Formalização

Uma Máquina de Turing é uma 8-upla:

$$M = (\Sigma, Q, \Pi, q_0, F, V, \beta, \circ)$$

onde:

- ▶  $\Sigma$  é o alfabeto de entrada;
- ▶  $Q$  é o conjunto de estados;
- ▶  $\Pi$  é a função (parcial) de transição:

$$\Pi : Q \times (\Sigma \cup V \cup \{\beta, \circ\}) \rightarrow Q \times (\Sigma \cup V \cup \{\beta, \circ\}) \times \{E, D\}$$

# Formalização

Uma Máquina de Turing é uma 8-upla:

$$M = (\Sigma, Q, \Pi, q_0, F, V, \beta, \circ)$$

onde:

- ▶  $q_0 \in Q$  é o estado inicial;
- ▶  $F \subseteq Q$  é o conjunto de estados finais;
- ▶  $V$  é o alfabeto auxiliar,  $V \cap \Sigma = \emptyset$ ;
- ▶  $\beta \notin (\Sigma \cup V)$  é o símbolo especial “branco”;
- ▶  $\circ \notin (\Sigma \cup V)$  é o marcador de início de fita.

# Configuração

A configuração de uma Máquina de Turing deve representar:

- ▶ O estado corrente;
- ▶ O conteúdo corrente da fita;
- ▶ A posição do cursor sobre a fita.

Isso é feito considerando-se a configuração como um elemento  $(\alpha, q, \beta)$  do conjunto:

$$(\Sigma \cup V \cup \{\beta, \circ\})^* \times Q \times (\Sigma \cup V \cup \{\beta, \circ\})^*$$

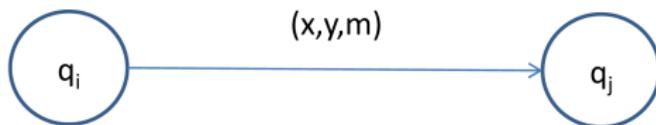
- ▶  $\alpha \in (\Sigma \cup V \cup \{\beta, \circ\})^*$  representa a parte da fita que está situada à esquerda da posição corrente do cursor;
- ▶  $q \in Q$  representa o estado corrente;
- ▶  $\beta \in (\Sigma \cup V \cup \{\beta, \circ\})^*$  representa a parte da fita que está situada à direita da posição corrente do cursor, incluindo a mesma.

# Diagrama de estados

Se:

$$\Pi(q_i, x) = (q_j, y, m)$$

então:



Nesse caso,

- ▶  $(\alpha, q_i, x\beta) \Rightarrow (\alpha y, q_j, \beta)$ , se  $m = D$ , ou
- ▶  $(\alpha z, q_i, x\beta) \Rightarrow (\alpha, q_j, zy\beta)$ , se  $m = E$ ,

representam possíveis movimentações a partir de uma mesma configuração pela aplicação da transição  $\Pi(q_i, x) = (q_j, y, m)$ .

# Critérios de aceitação

Existem várias maneiras de formular a aceitação de uma cadeia  $w$  por uma Máquina de Turing  $M$ . Todas elas são equivalentes entre si:

- 1 “Estado final”:  $w$  é aceita se, após a parada,  $M$  se encontra em um estado final; uma cadeia é rejeitada se, após a parada,  $M$  se encontra em um estado não-final;
- 2 “Entrada”:  $w$  é aceita imediatamente após a entrada de  $M$  em um estado final, mesmo que existam outras possibilidades de movimentação nesse estado; uma cadeia é rejeitada se, após a parada,  $M$  se encontra em estado não-final;
- 3 “Parada”:  $w$  é aceita se  $M$  pára; uma cadeia é rejeitada se  $M$  entra em loop infinito;

Em todos os casos,  $w$  é rejeitada se a cabeça de leitura/escrita se deslocar à esquerda da primeira célula da fita de entrada.

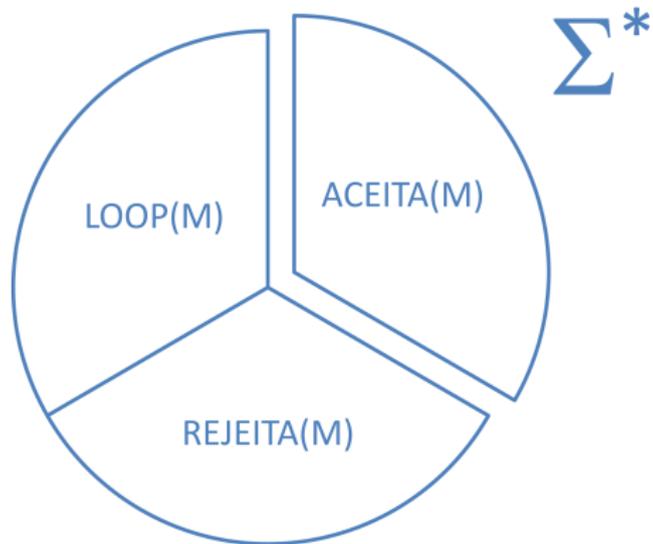
# MT e linguagens

Considere-se o critério de aceitação por “Entrada” e a Máquina de Turing:

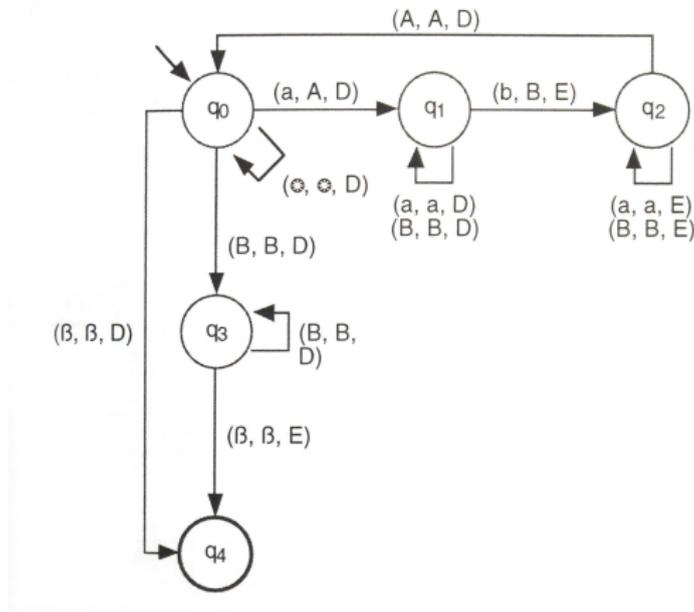
$$M = (\Sigma, Q, \Pi, q_0, F, V, \beta, \circ)$$

- ▶ A linguagem aceita por  $M$ , denotada  $ACEITA(M)$  ou  $L(M)$  é:  
 $\{w \in \Sigma^* | M \text{ assume algum estado } q_f \in F \text{ ao processar a entrada } w\}$
- ▶ A linguagem rejeitada por  $M$ , denotada  $REJEITA(M)$  é:  
 $\{w \in \Sigma^* | M \text{ pára em um estado } q_f \notin F \text{ ao processar a entrada } w$   
 ou a cabeça de leitura/escrita se desloca para a  
 esquerda da primeira posição}
- ▶ A linguagem para a qual  $M$  entra em loop, denotada  $LOOP(M)$  é:  
 $\{w \in \Sigma^* | M \text{ processa a entrada indefinidamente}\}$

## Particionamento



## Exemplo



## Exemplo

- ▶  $ACEITA(M) = \{a^n b^n \mid n \geq 0\}$
- ▶  $REJEITA(M) = \Sigma^* - ACEITA(M)$
- ▶  $LOOP(M) = \{\}$

Computação de  $M$  com a entrada  $aabb$ :

- ▶  $(\epsilon, q_0, \circ aabb), (\circ, q_0, aabb),$   
 $(\circ A, q_1, abb), (\circ Aa, q_1, bb),$   
 $(\circ A, q_2, aBb), (\circ, q_2, AaBb),$   
 $(\circ A, q_0, aBb), (\circ AA, q_1, Bb),$   
 $(\circ AAB, q_1, b), (\circ AA, q_2, BB),$   
 $(\circ A, q_2, BB), (\circ AA, q_0, BB),$   
 $(\circ AAB, q_3, B), (\circ AAB B, q_3, \epsilon),$   
 $(\circ AAB, q_4, B)$

# Linguagem recursivamente enumerável

## Definição

$L \subseteq \Sigma^*$  é dita recursivamente enumerável se existe uma Máquina de Turing  $M$  tal que:

- ▶ Se  $w \in L$ ,  $M$  pára e aceita a entrada;
- ▶ Se  $w \notin L$ ,  $M$ :
  - ▶ Pára e rejeita a entrada, ou
  - ▶ Entra em processamento indefinido e não pára (“loop infinito”).

Corresponde à maior classe de linguagens que pode ser reconhecida mecanicamente, porém sem garantia de que o processamento pára quando a cadeia de entrada não pertence à linguagem definida.

# Linguagem recursiva

## Definição

$L \subseteq \Sigma^*$  é dita recursiva se existe uma Máquina de Turing  $M$  tal que:

- ▶ Se  $w \in L$ ,  $M$  pára e aceita a entrada;
- ▶ Se  $w \notin L$ ,  $M$ : pára e rejeita a entrada.

Corresponde à maior classe de linguagens que pode ser reconhecida mecanicamente, com garantia de que o processamento pára para toda e qualquer cadeia de entrada.

# Linguagem recursivamente enumerável $\times$ linguagem recursiva

- ▶ Toda linguagem recursiva é também recursivamente enumerável;
- ▶ Existem linguagens que são recursivamente enumeráveis porém não são recursivas;
- ▶  $C_{LR} \subset C_{LRE}$ ;
- ▶ Se  $L$  é uma dessas linguagens, então toda e qualquer Máquina de Turing que aceita  $L$  é tal que:
  - ▶  $ACEITA(M) = L$ ;
  - ▶  $REJEITA(M) = \Sigma^* - L - LOOP(M)$ ;
  - ▶  $LOOP(M) \neq \{\}$ .
- ▶ Ou seja, existe pelo menos uma cadeia de entrada que faz  $M$  entrar em loop infinito.

# Propriedades

Serão demonstradas mais adiante:

- ▶ O complemento de uma linguagem recursiva é uma linguagem recursiva;
- ▶ Se uma linguagem e o seu complemento são recursivamente enumeráveis, então a linguagem é recursiva.

# MT funções

- ▶ Máquinas de Turing pode ser vistas e estudadas como dispositivos que definem linguagens;
- ▶ Máquinas de Turing podem, também, ser vistas como dispositivos que computam funções:
  - ▶ O argumento é posicionado na fita de entrada;
  - ▶ Ao término da computação o conteúdo da fita representa o resultado da aplicação da função ao argumento fornecido.
- ▶ Definição de linguagens  $\Leftrightarrow$  Computação de funções.

# Função computável

Uma função parcial:

$$f : (\Sigma^*)^n \rightarrow \Sigma^*$$

é dita *Função Turing-Computável*, ou simplesmente *Função Computável* se existe uma Máquina de Turing  $M = (\Sigma, Q, \Pi, q_0, F, V, \beta, \circ)$  que computa  $f$ , ou seja:

- ▶ Considere  $(w_1, w_2, \dots, w_n) \in (\Sigma^*)^n$ , representada na fita de entrada como  $\circ w_1 w_2 \dots w_n$
- ▶ Se  $f(w_1, w_2, \dots, w_n) = w$ , então o processamento de  $M$  com a entrada  $\circ w_1 w_2 \dots w_n$ :
  - ▶ Pára (não importa se aceitando ou rejeitando);
  - ▶ O conteúdo da fita de entrada é  $\circ w$ .
- ▶ Se  $f$  não é definida para o argumento  $(w_1, w_2, \dots, w_n)$ , então o processamento de  $M$  com a entrada  $\circ w_1 w_2 \dots w_n$ :
  - ▶ Entra em loop infinito.

# Função computável total

Uma função total:

$$f : (\Sigma^*)^n \rightarrow \Sigma^*$$

é dita *Função Turing-Computável Total*, ou simplesmente *Função Computável Total* se existe uma Máquina de Turing:

$$M = (\Sigma, Q, \Pi, q_0, F, V, \beta, \circ)$$

que computa  $f$  e que sempre pára para qualquer entrada.

# Função computável total

## Exemplo 1

Considere a função total:

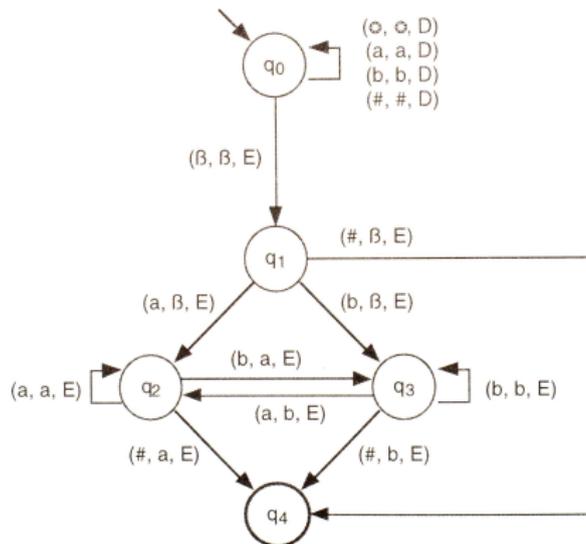
$$f : (\{a, b\}^*)^2 \rightarrow \{a, b\}^*$$

$f$  devolve a concatenação de duas cadeias quaisquer fornecidas como entrada, ou seja  $f(w_1, w_2) = w_1w_2$ . O símbolo  $\#$  será usado para delimitar  $w_1$  e  $w_2$  na cadeia de entrada. Exemplos:

- ▶  $f(b, a) = ba$ . A fita inicia com  $\circ b\#a$  e termina com  $\circ ba$
- ▶  $f(abb, abab) = abbabab$ . A fita inicia com  $\circ abb\#abab$  e termina com  $\circ abbabab$

# Função computável total

## Exemplo 1



# Função computável total

## Exemplo 1

Algoritmo: a segunda cadeia é deslocada uma posição para a esquerda, símbolo por símbolo; o símbolo  $\#$  desaparece.

- ▶ Desloca a cabeça para a direita até encontrar o primeiro branco;
- ▶ Desloca uma posição para a esquerda, memoriza o símbolo lido no estado e desloca novamente para a esquerda;
- ▶  $q_2$  representa que o último símbolo lido foi  $a$  e  $q_3$  representa  $b$ ;
- ▶ Conforme o símbolo corrente, grava um novo símbolo no lugar dele correspondente ao estado em que a máquina se encontra;
- ▶ Se houver necessidade, mudar de  $q_2$  para  $q_3$  e vice-versa para manter a coerência no significado atribuído aos estados;
- ▶ Fazer isso sucessivas vezes, até encontrar o símbolo  $\#$ .

# Função computável total

## Exemplo 2

Considere a função total:

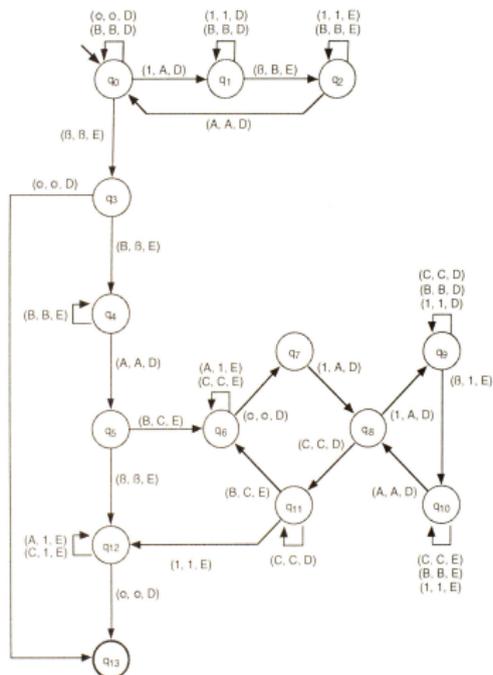
$$g : \{1\}^* \rightarrow \{1\}^*$$

$f$  devolve o quadrado do número de entrada (ambos representados em unário), ou seja  $g(n) = n^2$ . Exemplos:

- ▶  $f(1) = 1$ . A fita inicia com  $\circ 1$  e termina com  $\circ 1$
- ▶  $f(111) = 11111111$ . A fita inicia com  $\circ 111$  e termina com  $\circ 11111111$

## Função computável total

## Exemplo 2



# Teorema 1

Máquina de Turing  $\leq$  Máquina Norma

- ▶ Toda Máquina de Turing pode ser simulada por alguma Máquina Norma.
- ▶ Se  $M = (\Sigma, Q, \Pi, q_0, F, V, \beta, \circ)$  é uma Máquina de Turing, então existe um programa monolítico  $P$  que simula  $M$  na Máquina Norma.

# Teorema 1

## Pré-requisitos

- ▶ O critério de aceitação deve ser por “Entrada”;
- ▶ A função de transição  $\Pi$  deve ser total:
 
$$Q' \leftarrow Q \cup \{q_e\}$$

$$\Pi' \leftarrow \Pi$$

$\forall q \in Q', \tau \in (\Sigma \cup V)$ , se  $\Pi$  não é definida para  $(q, \tau)$   
então  $\Pi' \leftarrow \Pi' \cup \{(q, \tau) \rightarrow (q_e, \tau, E)\}$
- ▶ As cadeias serão rejeitadas por tentativa de movimentação da cabeça de leitura/escrita à esquerda da primeira posição da fita.

# Teorema 1

## Convenções

- ▶ Símbolos de  $\Sigma \cup V$ ,  $|\Sigma \cup V| = m$ :
  - $\tau_j$ , considerando  $1 \leq j \leq m$ , é representado pelo valor  $j$ ;
  - $\beta$  é representado por 0;
  - $\circ$  é representado por  $m + 1$ .
- ▶ A fita de entrada é representada como um vetor armazenado no registrador  $X$ ;
- ▶ Observar que a escolha da representação de  $\beta$  por 0 faz com que existam infinitos símbolos  $\beta$  à direita do último símbolo da cadeia de entrada. Qualquer elemento do vetor que não contenha um elemento de  $\Sigma \cup V$  retorna, na codificação de ênuplas, o valor 0 (de  $2^0$ ), que representa o símbolo  $\beta$ .

# Teorema 1

## Convenções

- ▶ A posição referenciada pela cabeça de leitura/escrita corresponde ao conteúdo do registrador  $C$ :  
valor inicial 1 aponta para o símbolo  $\circ$ ;
- ▶ O estado corrente é representado pelo conteúdo do registrador  $Q$ :  
 $q_i, i \geq 0$ , é representado pelo valor  $i$ ;
- ▶ Ao término do processamento,  $Y = 0$  indica rejeição da cadeia de entrada;  $Y \neq 0$  indica aceitação da cadeia de entrada, e o valor de  $Y$  representa o conteúdo da fita nessa situação;
- ▶ Observar que o conteúdo da fita é representado por um valor sempre maior ou igual a 1 (será 1 se ela contiver apenas brancos).

## Teorema 1

## Algoritmo

Instruções iniciais de  $P$  para a Máquina Norma:

$r_0$  : se  $zero_C$  então vá para  $r_{0_1}$  senão vá para  $r_{0_2}$

$r_{0_1}$  : faça  $Y := 0$  vá para  $r_{0_5}$

$r_{0_2}$  : faça  $A := 2^Q * 3^{X[C]}$  vá para  $End\_A$

$r_{0_3}$  : se  $zero_C$  então vá para  $r_{0_1}$  senão vá para  $r_{0_4}$

$r_{0_4}$  : faça  $Y := X$  vá para  $r_{0_5}$

- ▶  $r_0$  é o rótulo inicial;
- ▶ O controle retorna para  $r_0$  sempre que o próximo estado é não-final;
- ▶ O controle retorna para  $r_{0_3}$  sempre que o próximo estado é final;
- ▶  $r_{0_5}$  é o rótulo final.

## Teorema 1

## Algoritmo

Para cada transição  $\Pi(q_i, \tau_m) = (q_j, \tau_n, D)$ , acrescentar à  $P$  o seguinte conjunto de instruções:

$r_{2^i * 3^m}$  : faça  $X[C] := n$  vá\_ para  $r_{2^i * 3^{m_1}}$

$r_{2^i * 3^{m_1}}$  : faça  $add_C$  vá\_ para  $r_{2^i * 3^{m_2}}$

$r_{2^i * 3^{m_2}}$  : faça  $Q := j$  vá\_ para  $r_0$

- ▶ Grava  $\tau_n$  na posição corrente da fita;
- ▶ Desloca a cabeça de leitura/escrita para a direita;
- ▶ Atualiza o estado corrente para  $q_j$ ;
- ▶ Se o movimento for para a esquerda, usar  $sub_C$  no lugar de  $add_C$ ;
- ▶ Se  $q_j \in F$ , então substituir  $r_0$  por  $r_{0_3}$ .

# Teorema 1

## Algoritmo

Se  $|Q| = m$  e  $|\Sigma \cup V| = n$ , então o programa monolítico correspondente possuirá:

$$5 + m * (n + 2) * 3$$

instruções rotuladas.

Detalhamento do cálculo:

- ▶ 5: quantidade de instruções rotuladas iniciais;
- ▶  $n + 2$ :  $n$  símbolos, mais  $\beta$  e  $\circ$ ;
- ▶  $m * (n + 2)$ : função de transição total, quantidade total de transições;
- ▶ 3: quantidade de instruções rotuladas por transição.

## Exemplo

## Função de transição total

Considere a Máquina de Turing  $M$  que aceita a linguagem  $\{a^n b^n | n \geq 0\}$ .

Então  $\Pi' \leftarrow \Pi \cup$

{

$$\begin{aligned}
 & (q_0, b) \rightarrow (q_e, b, E) \quad , \quad (q_0, A) \rightarrow (q_e, A, E) \quad , \quad (q_1, A) \rightarrow (q_e, A, E), \\
 & (q_1, B) \rightarrow (q_e, B, E) \quad , \quad (q_1, \circ) \rightarrow (q_e, \circ, E) \quad , \quad (q_1, \beta) \rightarrow (q_e, \beta, E), \\
 & (q_2, b) \rightarrow (q_e, b, E) \quad , \quad (q_2, B) \rightarrow (q_e, B, E) \quad , \quad (q_2, \circ) \rightarrow (q_e, \circ, E), \\
 & (q_2, \beta) \rightarrow (q_e, \beta, E) \quad , \quad (q_3, a) \rightarrow (q_e, a, E) \quad , \quad (q_3, b) \rightarrow (q_e, b, E), \\
 & (q_3, A) \rightarrow (q_e, \circ, E) \quad , \quad (q_3, \circ) \rightarrow (q_e, \circ, E) \quad , \quad (q_4, a) \rightarrow (q_e, a, E), \\
 & (q_4, b) \rightarrow (q_e, b, E) \quad , \quad (q_4, A) \rightarrow (q_e, A, E) \quad , \quad (q_4, B) \rightarrow (q_e, B, E), \\
 & (q_4, \circ) \rightarrow (q_e, \circ, E) \quad , \quad (q_4, \beta) \rightarrow (q_e, \beta, E) \quad , \quad (q_e, a) \rightarrow (q_e, a, E), \\
 & (q_e, b) \rightarrow (q_e, b, E) \quad , \quad (q_e, A) \rightarrow (q_e, A, E) \quad , \quad (q_e, B) \rightarrow (q_e, B, E), \\
 & (q_e, \circ) \rightarrow (q_e, \circ, E) \quad , \quad (q_e, \beta) \rightarrow (q_e, \beta, E)
 \end{aligned}$$

}

# Exemplo

## Representação dos estados

Considere a Máquina de Turing  $M$  que aceita a linguagem  $\{a^n b^n | n \geq 0\}$ .  
Então:

- ▶  $Q' = \{q_0, q_1, q_2, q_3, q_4, q_e\}$   
Representação no registrador  $Q$ :

- ▶  $q_0$  por 0
- ▶  $q_1$  por 1
- ▶  $q_2$  por 2
- ▶  $q_3$  por 3
- ▶  $q_4$  por 4
- ▶  $q_e$  por 5

# Exemplo

## Representação dos símbolos

Considere a Máquina de Turing  $M$  que aceita a linguagem  $\{a^n b^n \mid n \geq 0\}$ .  
Então:

- ▶  $\Sigma \cup V = \{a, b\} \cup \{A, B\} = \{a, b, A, B\}$

Representação no registrador  $X$ :

- ▶  $a$  por 1,
- ▶  $b$  por 2,
- ▶  $A$  por 3,
- ▶  $B$  por 4,

Adicionalmente:

- ▶  $\beta$  por 0
- ▶  $\circ$  como 5

# Exemplo

## Configuração inicial

Situação inicial dos registradores na Máquina Norma para a cadeia de entrada  $\circ aabb$ :

- ▶  $\circ aabb$  é representada pela seqüência 51122
- ▶  $X = 2^5 * 3^1 * 5^1 * 7^2 * 11^2 = 2.845.920$
- ▶  $Q = 0$
- ▶  $C = 1$

# Exemplo

## Programa para Máquina Norma

$r_0$  : se  $zero_C$  então vá para  $r_{0_1}$  senão vá para  $r_{0_2}$   
 $r_{0_1}$  : faça  $Y := 0$  vá para  $r_{0_5}$   
 $r_{0_2}$  : faça  $A := 2^Q * 3^{X[C]}$  vá para  $End\_A$   
 $r_{0_3}$  : se  $zero_C$  então vá para  $r_{0_1}$  senão vá para  $r_{0_4}$   
 $r_{0_4}$  : faça  $Y := X$  vá para  $r_{0_5}$

# Exemplo

## Programa para Máquina Norma

$$\Pi(q_0, \circ) = (q_0, \circ, D)$$

$r_{243}$  : faça  $X[C] := 5$  vá\_ para  $r_{243_1}$

$r_{243_1}$  : faça  $add_C$  vá\_ para  $r_{243_2}$

$r_{243_2}$  : faça  $Q := 0$  vá\_ para  $r_0$

$$\Pi(q_0, a) = (q_1, A, D)$$

$r_3$  : faça  $X[C] := 3$  vá\_ para  $r_{3_1}$

$r_{3_1}$  : faça  $add_C$  vá\_ para  $r_{3_2}$

$r_{3_2}$  : faça  $Q := 1$  vá\_ para  $r_0$

...

## Exemplo

## Programa para Máquina Norma

...

$$\Pi(q_0, B) = (q_3, B, D)$$

$r_{81}$  : faça  $X[C] := 4$  vá\_ para  $r_{81_1}$

$r_{81_1}$  : faça  $add_C$  vá\_ para  $r_{82_2}$

$r_{81_2}$  : faça  $Q := 3$  vá\_ para  $r_0$

...

$$\Pi(q_3, \beta) = (q_4, \beta, E)$$

$r_8$  : faça  $X[C] := 0$  vá\_ para  $r_{8_1}$

$r_{8_1}$  : faça  $sub_C$  vá\_ para  $r_{8_2}$

$r_{8_2}$  : faça  $Q := 4$  vá\_ para  $r_{0_3}$

...

## Exemplo

## Endereços das transições

Q	$\tau$	X[C]	r
0	$\circ$	5	243
0	a	1	3
0	b	2	9
0	A	3	27
0	B	4	81
0	$\beta$	0	1

Q	$\tau$	X[C]	r
1	$\circ$	5	486
1	a	1	6
1	b	2	18
1	A	3	54
1	B	4	162
1	$\beta$	0	2

Q	$\tau$	X[C]	r
2	$\circ$	5	972
2	a	1	12
2	b	2	36
2	A	3	108
2	B	4	324
2	$\beta$	0	4

Q	$\tau$	X[C]	r
3	$\circ$	5	1944
3	a	1	24
3	b	2	72
3	A	3	216
3	B	4	648
3	$\beta$	0	8

Q	$\tau$	X[C]	r
4	$\circ$	5	3888
4	a	1	48
4	b	2	144
4	A	3	432
4	B	4	1296
4	$\beta$	0	16

Q	$\tau$	X[C]	r
5	$\circ$	5	7776
5	a	1	96
5	b	2	288
5	A	3	864
5	B	4	2592
5	$\beta$	0	32

## Exemplo

## Configurações

Cadeia  $ab \in L(M)$ :

Turing	rótulo	$Q$	$C$	$X$	$A$
$\Pi(q_0, \circ) = (q_0, \circ, D)$	$r_0$	0	1	$2^5 * 3^1 * 5^2$	$2^0 * 3^5 = 243$
	$r_{243}$	0	1	$2^5 * 3^1 * 5^2$	$2^0 * 3^5 = 243$
	$r_{243_1}$	0	1	$2^5 * 3^1 * 5^2$	$2^0 * 3^5 = 243$
	$r_{243_2}$	0	2	$2^5 * 3^1 * 5^2$	$2^0 * 3^5 = 243$
$\Pi(q_0, a) = (q_1, A, D)$	$r_0$	0	2	$2^5 * 3^1 * 5^2$	$2^0 * 3^1 = 3$
	$r_3$	0	2	$2^5 * 3^1 * 5^2$	$2^0 * 3^1 = 3$
	$r_{3_1}$	0	2	$2^5 * 3^3 * 5^2$	$2^0 * 3^1 = 3$
	$r_{3_2}$	0	3	$2^5 * 3^3 * 5^2$	$2^0 * 3^1 = 3$

## Exemplo

## Configurações

Turing	rótulo	$Q$	$C$	$X$	$A$
$\Pi(q_1, b) = (q_2, B, E)$	$r_0$	1	3	$2^5 * 3^3 * 5^2$	$2^1 * 3^2 = 18$
	$r_{18}$	1	3	$2^5 * 3^3 * 5^2$	$2^1 * 3^2 = 18$
	$r_{18_1}$	1	3	$2^5 * 3^3 * 5^4$	$2^1 * 3^2 = 18$
	$r_{18_2}$	1	2	$2^5 * 3^3 * 5^4$	$2^1 * 3^2 = 18$
$\Pi(q_2, A) = (q_0, A, D)$	$r_0$	2	2	$2^5 * 3^3 * 5^4$	$2^2 * 3^3 = 108$
	$r_{108}$	2	2	$2^5 * 3^3 * 5^4$	$2^2 * 3^3 = 108$
	$r_{108_1}$	2	2	$2^5 * 3^3 * 5^4$	$2^2 * 3^3 = 108$
	$r_{108_2}$	2	3	$2^5 * 3^3 * 5^4$	$2^2 * 3^3 = 108$

## Exemplo

## Configurações

Turing	rótulo	$Q$	$C$	$X$	$A$
$\Pi(q_0, B) = (q_3, B, D)$	$r_0$	0	3	$2^5 * 3^3 * 5^4$	$2^0 * 3^4 = 81$
	$r_{81}$	0	3	$2^5 * 3^3 * 5^4$	$2^0 * 3^4 = 81$
	$r_{81_1}$	0	3	$2^5 * 3^3 * 5^4$	$2^0 * 3^4 = 81$
	$r_{81_2}$	0	4	$2^5 * 3^3 * 5^4$	$2^0 * 3^4 = 81$
$\Pi(q_3, \beta) = (q_4, \beta, E)$	$r_0$	3	4	$2^5 * 3^3 * 5^4$	$2^3 * 3^0 = 8$
	$r_8$	3	4	$2^5 * 3^3 * 5^4$	$2^3 * 3^0 = 8$
	$r_{81}$	3	4	$2^5 * 3^3 * 5^4$	$2^3 * 3^0 = 8$
	$r_{82}$	3	3	$2^5 * 3^3 * 5^4$	$2^3 * 3^0 = 8$

# Exemplo

## Configurações

Turing	rótulo	$Q$	$C$	$X$	$A$
	$r_{0_3}$	4	3	$2^5 * 3^3 * 5^4$	$2^3 * 3^0 = 8$
	$r_{0_4}$	4	3	$2^5 * 3^3 * 5^4$	$2^3 * 3^0 = 8$
	$r_{0_5}$	4	3	$2^5 * 3^3 * 5^4$	$2^3 * 3^0 = 8$

## Exemplo

## Configurações

Cadeia  $ba \notin L(M)$ :

Turing	rótulo	$Q$	$C$	$X$	$A$
$\Pi(q_0, \circ) = (q_0, \circ, D)$	$r_0$	0	1	$2^5 * 3^2 * 5^1$	$2^0 * 3^5 = 243$
	$r_{243}$	0	1	$2^5 * 3^2 * 5^1$	$2^0 * 3^5 = 243$
	$r_{243_1}$	0	1	$2^5 * 3^2 * 5^1$	$2^0 * 3^5 = 243$
	$r_{243_2}$	0	2	$2^5 * 3^2 * 5^1$	$2^0 * 3^5 = 243$
$\Pi(q_0, b) = (q_e, b, E)$	$r_0$	0	2	$2^5 * 3^2 * 5^1$	$2^0 * 3^2 = 9$
	$r_9$	0	2	$2^5 * 3^2 * 5^1$	$2^0 * 3^2 = 9$
	$r_{9_1}$	0	2	$2^5 * 3^2 * 5^1$	$2^0 * 3^2 = 9$
	$r_{9_2}$	0	1	$2^5 * 3^2 * 5^1$	$2^0 * 3^2 = 9$

## Exemplo

## Configurações

Turing	rótulo	$Q$	$C$	$X$	$A$
$\Pi(q_e, \circ) = (q_e, \circ, E)$	$r_0$	5	1	$2^5 * 3^2 * 5^1$	$2^5 * 3^5 = 7776$
	$r_{7776}$	5	1	$2^5 * 3^2 * 5^1$	$2^5 * 3^5 = 7776$
	$r_{7776_1}$	5	1	$2^5 * 3^2 * 5^1$	$2^5 * 3^5 = 7776$
	$r_{7776_2}$	5	0	$2^5 * 3^2 * 5^1$	$2^5 * 3^5 = 7776$
	$r_0$	5	0	$2^5 * 3^2 * 5^1$	$2^5 * 3^5 = 7776$
	$r_{0_1}$	5	0	$2^5 * 3^2 * 5^1$	$2^5 * 3^5 = 7776$
	$r_{0_5}$	5	0	$2^5 * 3^2 * 5^1$	$2^5 * 3^5 = 7776$

## Teorema 2

Máquina Norma  $\leq$  Máquina de Turing

- ▶ Conforme demonstrado em Bird76, programas recursivos (com definição e chamada de subprogramas e recursão) podem ser simulados em Máquinas Norma através de programas monolíticos, com o uso de endereçamento indireto.
- ▶ Portanto, é suficiente considerar programas monolíticos e as Máquinas de Turing que computam as mesmas funções;
- ▶ Também é suficiente considerar Máquina Norma com apenas dois registradores ( $X$  e  $Y$ ). Ela será denotada  $Norma_2$ .

# Teorema 2

Máquina Norma  $\leq$  Máquina de Turing

- ▶ Todo Máquina Norma pode ser simulada por alguma Máquina de Turing.
- ▶ Se  $P = (I, r_0)$  para  $Norma_2$ , então existe  $M : (\Sigma, Q, \Pi, q_0, F, V, \beta, \circ)$  que simula  $P$ .
- ▶ Máquina de Turing simula  $Norma_2$ , que por sua vez simula Norma.

## Teorema 2

## Convenções

- ▶ Registrador  $X$ : Seu conteúdo é representado em unário e armazenado nas células pares da fita de  $M$ ;  
A fita de entrada  $|\circ|\underbrace{|1|}_{2}|\underbrace{|1|}_{4}|\underbrace{|1|}_{6}|\underbrace{|1|}_{8}|\beta|\beta|\beta|\dots|$  representa  $X = 4$
- ▶ Registrador  $Y$ : Seu conteúdo é representado em unário e armazenado nas células ímpares (exceto a primeira) da fita de  $M$ ; A fita de entrada  $|\circ|1|\underbrace{|1|}_{3}|\underbrace{|1|}_{5}|\underbrace{|1|}_{7}|\beta|\beta|\beta|\dots|$  representa  $Y = 3$
- ▶ Rótulos das instruções: Cada rótulo  $r$  de  $P$  está em correspondência com um estado  $q_r$  de  $M$ .  $r_0$  corresponde ao estado inicial  $q_0$  e cada rótulo final  $r_f$  corresponde a um estado final  $q_f \in F$ .

# Teorema 2

## Componentes da Máquina de Turing

- ▶  $\Sigma = \{1\}$
- ▶  $Q = \{q_i | r_i \text{ é rótulo de } P\}$
- ▶  $\Pi = \{\}$
- ▶ Estado inicial =  $q_0$  (supondo que  $r_0$  é o rótulo inicial de  $P$ )
- ▶  $F = \{q_i | r_i \text{ é rótulo final de } P\}$
- ▶  $V = \{\}$

# Teorema 2

## Algoritmo

Para cada instrução  $i \in I$ :

- ▶  $r$ : faça  $add_k$  vá  $\_$  para  $r'$ 
  - ▶ A partir do estado  $q_r$ , deslocar a cabeça de leitura/escrita até encontrar a primeira célula par (se  $K = X$ ) ou ímpar (se  $K = Y$ ) que contenha um símbolo  $\beta$ ;
  - ▶ Substituir esse  $\beta$  por 1;
  - ▶ Deslocar a cabeça de leitura/escrita, posicionando-a sobre o primeiro símbolo da fita ( $\circ$ );
  - ▶ Ir para o estado  $q_{r'}$ .

# Teorema 2

## Algoritmo

Para cada instrução  $i \in I$ :

- ▶  $r$ : faça  $sub_k$  vá \_ para  $r'$ 
  - ▶ A partir do estado  $q_r$ , deslocar a cabeça de leitura/escrita até encontrar a última célula par (se  $K = X$ ) ou ímpar (se  $K = Y$ ) que contenha um símbolo 1 (se a primeira célula pesquisada for  $\beta$ , ir para 3);
  - ▶ Substituir esse 1 por  $\beta$ ;
  - ▶ Deslocar a cabeça de leitura/escrita, posicionando-a sobre o primeiro símbolo da fita ( $\circ$ );
  - ▶ Ir para o estado  $q_{r'}$ .

## Teorema 2

## Algoritmo

Para cada instrução  $i \in I$ :

- ▶ se  $zero_K$  então vá \_ para  $r'$  senão vá \_ para  $r''$ 
  - ▶ A partir do estado  $q_r$ , deslocar a cabeça de leitura/escrita até encontrar a primeira célula par (se  $K = X$ ) ou a segunda célula ímpar (se  $K = Y$ );
  - ▶ Se a célula encontrada contiver o símbolo  $\beta$ :
    - ▶ Deslocar a cabeça de leitura/escrita, posicionando-a sobre o primeiro símbolo da fita ( $\circ$ );
    - ▶ Ir para o estado  $q_{r'}$
  - ▶ Se a célula encontrada não contiver o símbolo  $\beta$ :
    - ▶ Deslocar a cabeça de leitura/escrita, posicionando-a sobre o primeiro símbolo da fita ( $\circ$ );
    - ▶ Ir para o estado  $q_{r''}$

# Teorema 2

## Exemplo

$r_1$ : se  $zero_X$  vá para  $r_4$  senão vá para  $r_2$

$r_2$ : faça  $sub_X$  vá para  $r_3$

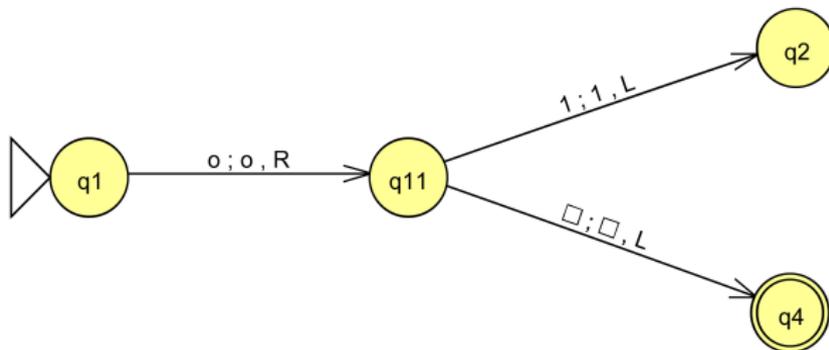
$r_3$ : faça  $add_Y$  vá para  $r_1$

- ▶  $\Sigma = \{1\}$
- ▶  $Q = \{q_1, q_2, q_3, q_4\}$
- ▶  $\Pi = \{\}$
- ▶ Estado inicial  $q_1$  (pois  $r_1$  é o rótulo inicial de  $P$ )
- ▶  $F = \{q_4\}$  (pois  $r_4$  é rótulo final de  $P$ )
- ▶  $V = \{\}$

## Teorema 2

## Exemplo

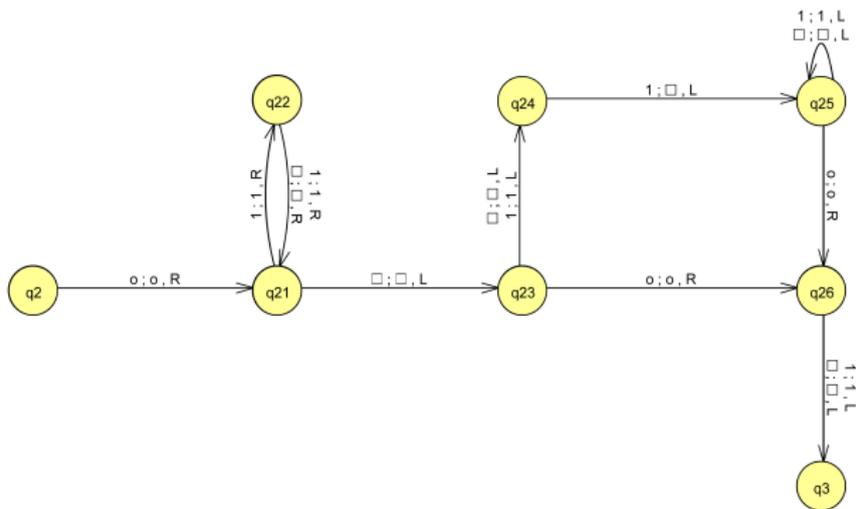
$r_1$  : se  $zero_X$  vá \_ para  $r_4$  senão vá \_ para  $r_2$



## Teorema 2

## Exemplo

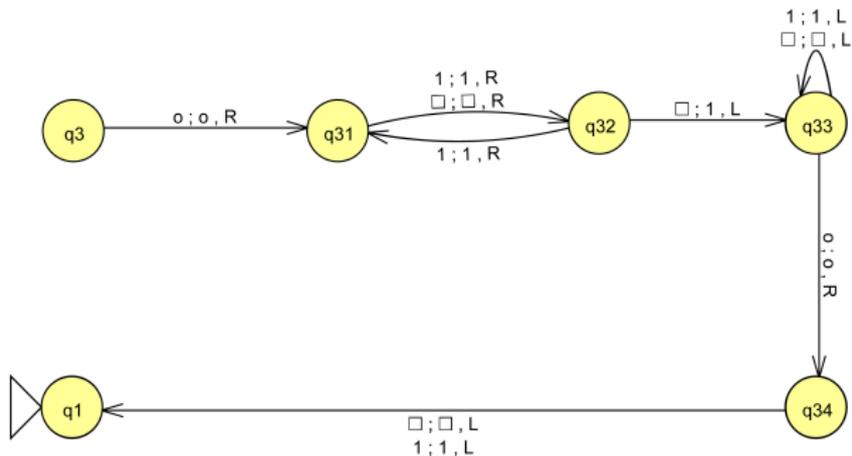
$r_2$  : faça  $sub_X$  vá\_ para  $r_3$



## Teorema 2

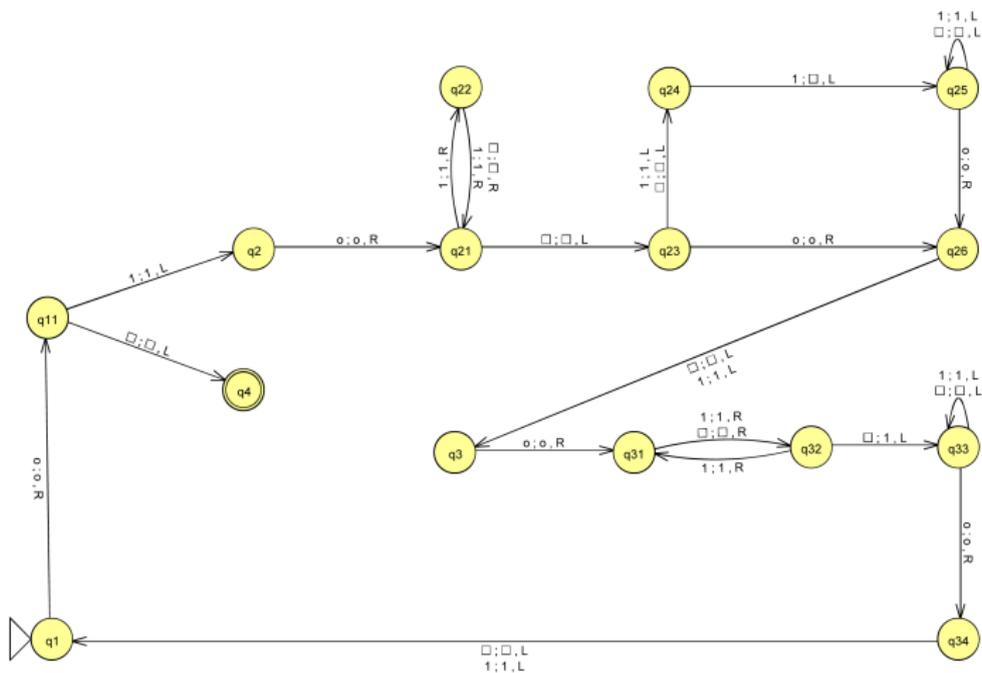
## Exemplo

$r_3$  : faça  $add_Y$  vá\_ para  $r_1$



## Teorema 2

## Exemplo



# Generalidades

- ▶ Definida por Emil Leon Post em 1936;
- ▶ Possui uma única variável, denominada  $X$ :
  - ▶ Fila (*first-in-first-out*);
  - ▶ Entrada, saída e trabalho;
  - ▶ Tamanho inicial igual ao comprimento da cadeia de entrada;
  - ▶ Tamanho pode variar, sem restrições.
- ▶ Possui um programa associado (fluxograma):
  - ▶ Partida;
  - ▶ Parada;
  - ▶ Desvio condicional;
  - ▶ Atribuição.
- ▶ Máquina Universal.

# Definição

Uma Máquina de Post é uma tripla:

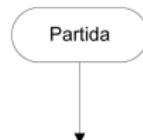
$$M = (\Sigma, D, \#)$$

onde:

- ▶  $\Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_n\}$  é o alfabeto de entrada;
- ▶  $D$  é o programa (ou fluxograma), constituído pelos componentes partida, parada, desvio condicional e atribuição;
- ▶  $\#$  é o símbolo auxiliar,  $\# \notin \Sigma$ .

# Definição

Componente “Partida”:



- ▶ Único em cada programa  $P$ ;
- ▶ Indica o início da execução de  $P$ .

# Definição

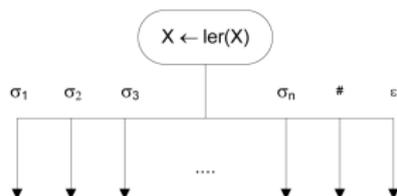
Componente “Parada”:



- ▶ Pode ser de dois tipos: parada com aceitação ou parada com rejeição;
- ▶ Múltiplas ocorrências são permitidas, sem restrições (inclusive zero ocorrências de qualquer ou de ambos os componentes).

# Definição

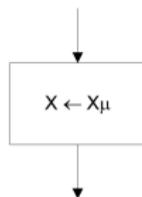
Componente “Desvio condicional”:



- ▶ Analisa o primeiro símbolo da fila (variável  $X$ );
- ▶ Conforme o símbolo encontrado, desvia de acordo;
- ▶ O símbolo encontrado é removido do início da fila;
- ▶ Se  $|\Sigma| = n$ , devem ser previstos  $n + 2$  desvios;
- ▶ Desvio para o símbolo auxiliar ( $\#$ ) e também para o caso de  $X$  conter a cadeia vazia ( $\epsilon$ ).

# Definição

Componente “Atribuição”:

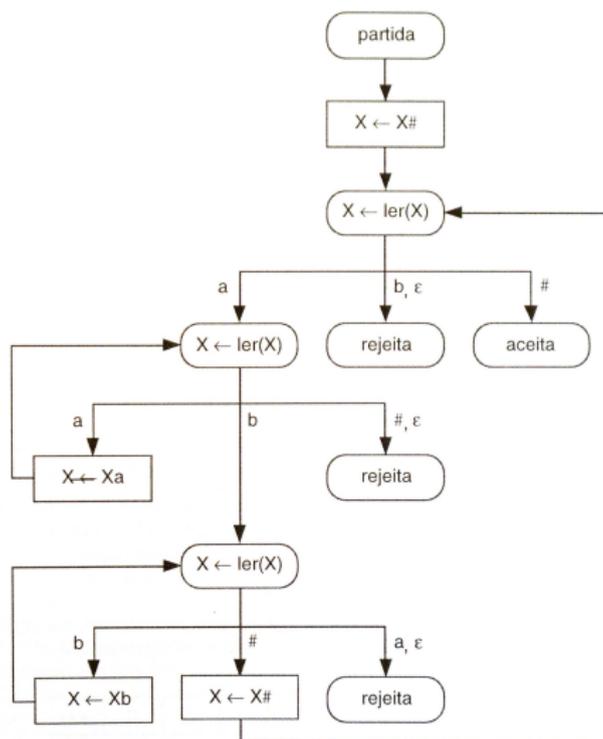


- ▶ Concatena o símbolo  $\mu$  ao final da cadeia contida em  $X$  (vai para o final da fila);
- ▶  $\mu \in (\Sigma \cup \{\#\})$

# Exemplo — programa $P$ que reconhece $a^n b^n$

## Estratégia:

- ▶  $X$  contém a cadeia a ser analisada;
- ▶ Acrescentar o símbolo  $\#$  ao final da mesma, para indicar final de cadeia;
- ▶ Se o primeiro símbolo for  $a$ , remover;
- ▶ Transportar todos os demais  $a$  para o final da cadeia;
- ▶ Se chegar num  $b$ , remover;
- ▶ Transportar todos os demais  $b$  para o final da cadeia;
- ▶ Repetir;
- ▶ Se a cadeia contiver apenas o símbolo de final de cadeia, ACEITA; senão REJEITA.

Exemplo — programa  $P$  que reconhece  $a^n b^n$ 

Exemplo — programa  $P$  que reconhece  $a^n b^n$ 

- ▶ Valores da variável  $X$  para a entrada  $aabb$ ;
- ▶ Parada com a condição ACEITA:

$aaabbb$	$aaabbb\#$	$aabbb\#$
$abbb\#$	$abbb\#a$	$bbb\#a$
$bbb\#aa$	$bb\#aa$	$b\#aa$
$b\#aab$	$\#aab$	$\#aabb$
$aabb$	$aabb\#$	$abb\#$
$bb\#$	$bb\#a$	$b\#a$
$\#a$	$\#ab$	$ab$
$ab\#$	$b\#$	$\#$
$\epsilon$	$\#$	$\epsilon$

# Teorema 3

Máquina de Turing  $\leq$  Máquina de Post

Seja:

$$M = (\Sigma, Q, \Pi, q_0, F, V, \beta, \circ)$$

uma Máquina de Turing. Então, existe uma Máquina de Post:

$$M' = (\Sigma \cup V, D, \#)$$

que simula  $M$ .

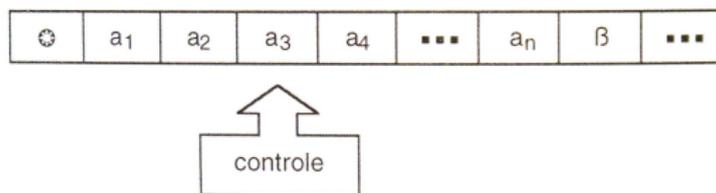
# Teorema 3

## Fita

- ▶ Representada pela variável  $X$ ;
- ▶ A posição correntemente referenciada pelo cursor indica a primeira posição da fila contida na variável  $X$ ;
- ▶ O símbolo  $\#$  é usado para sinalizar o final da cadeia de entrada;
- ▶ A parte situada à esquerda da fita de entrada corresponde à parte final da fila, situada após o símbolo  $\#$ .

## Teorema 3

Fita



A situação da fita acima é representada na Máquina de Post por:

$$X = a_3 a_4 \dots a_n \# \circ a_1 a_2$$

# Teorema 3

## Movimento para a DIREITA

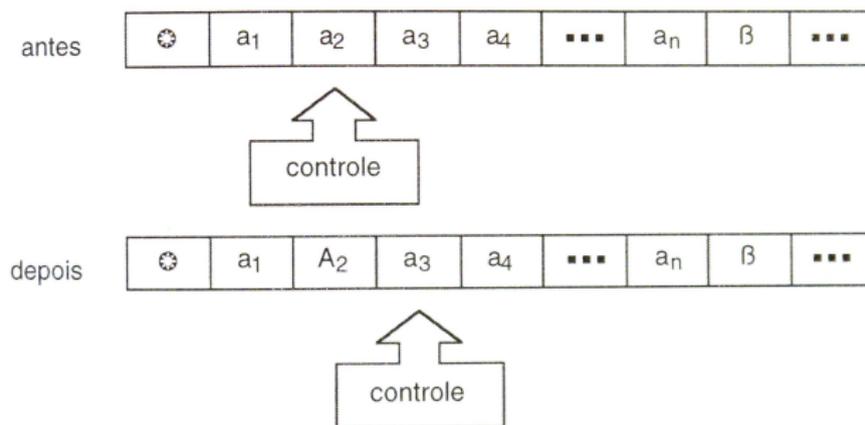
É necessário representar, na Máquina de Post:

- ▶ A substituição de um símbolo por outro, conforme a função de transição;
- ▶ O deslocamento do cursor uma posição para a direita;
- ▶ Seqüência de testes e atribuições que resultem na modificação pretendida (único teste e única atribuição).

## Teorema 3

## Movimento para a DIREITA

Supondo  $\Pi(q_i, a_2) = (q_j, A_2, D)$ :



Deseja-se mapear  $a_2 a_3 a_4 \dots a_n \# \circ a_1$  em  $a_3 a_4 \dots a_n \# \circ a_1 A_2$ .

# Teorema 3

## Movimento para a DIREITA

Direto (quando o símbolo corrente é diferente de  $\#$ ):

- ▶ Ler e remover o símbolo corrente  $a_2$  do início da fila;
- ▶ Inserir o símbolo  $A_2$  no final da fila;

Necessário considerar quando o símbolo corrente é  $\#$ :

- ▶ Remover  $\#$  do início da fila;
- ▶ Adicionar o novo símbolo no final da fila;
- ▶ Inserir o símbolo especial  $\$$  no final da fila;
- ▶ Inserir o símbolo  $\#$  no final da fila.
- ▶ Transportar símbolos do início para o final, até ler (e remover) o símbolo  $\$$ ;
- ▶  $\#abc$  - que representa  $(abc, q, \epsilon)$  - é mapeado em  $\#abcd$  (supondo que  $\beta$  é substituído por  $d$ ), que representa  $(abcd, q, \epsilon)$ .

## Teorema 3

## Movimento para a DIREITA

Situação	$X$
INÍCIO	$\#abc$
Remove $\#$	$abc$
Inserir $d$	$abcd$
Inserir $\$$	$abcd\$$
Inserir $\#$	$abcd\#\$$
Remove $a$	$bcd\#\$$
Inserir $a$	$bcd\#\$a$

## Teorema 3

Movimento para a DIREITA

Situação	$X$
Remove $b$	$cd\$#a$
Inserir $b$	$cd\$#ab$
Remove $c$	$d\$#ab$
Inserir $c$	$d\$#abc$
Remove $d$	$\$#abc$
Inserir $d$	$\$#abcd$
Remove $\$$	$\#abcd$
TÉRMINO	$\#abcd$

# Teorema 3

## Movimento para a ESQUERDA

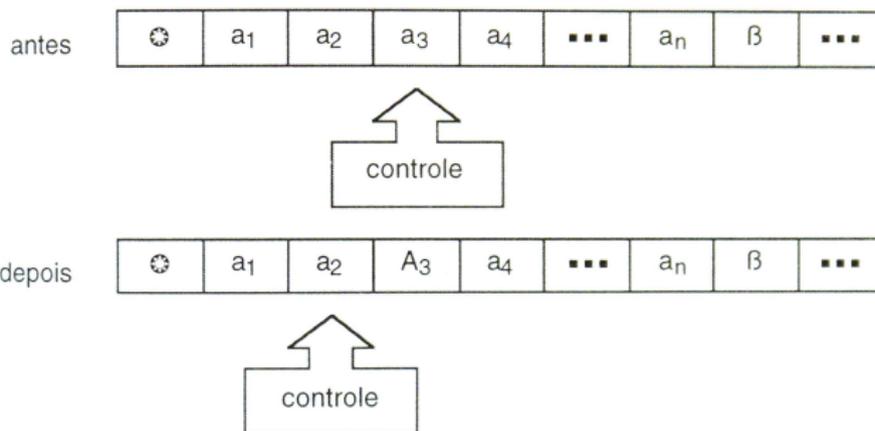
É necessário representar, na Máquina de Post:

- ▶ A substituição de um símbolo por outro, conforme a função de transição;
- ▶ O deslocamento do cursor uma posição para a esquerda;
- ▶ Seqüência de testes e atribuições que resultem na modificação pretendida.

## Teorema 3

## Movimento para a ESQUERDA

Supondo  $\Pi(q_i, a_3) = (q_j, A_3, E)$ :



Deseja-se mapear  $a_3a_4\dots a_n\# \circ a_1a_2$  em  $a_2A_3a_4\dots a_n\# \circ a_1$ .

# Teorema 3

## Movimento para a ESQUERDA

Será feito em duas etapas:

- ▶ Parte 1:
  - ▶ Entrada:  $a_3a_4\dots a_n\# \circ a_1a_2$
  - ▶ Saída:  $\$A_3a_4\dots a_n\# \circ a_1$
- ▶ Parte 2:
  - ▶ Entrada:  $\$A_3a_4\dots a_n\# \circ a_1$
  - ▶ Saída:  $a_2A_3a_4\dots a_n\# \circ a_1$

Necessário, também, considerar quando o símbolo corrente é  $\#$ .

# Teorema 3

## Movimento para a ESQUERDA

### Parte 1:

- ▶ Inserir \$\$ no final da fila;
- ▶ Inserção do novo símbolo no final da fila;
- ▶ Remover o símbolo do início da fila;
- ▶  $S_1 \leftarrow^1$  símbolo do início da fila (leitura e remoção);
- ▶  $S_2 \leftarrow$  símbolo do início da fila (leitura e remoção);
- ▶ Enquanto  $S_2$  for diferente de \$ faça:
  - ▶ Inserir  $S_1$  no final da fila;
  - ▶  $S_1 \leftarrow S_2$ ;
  - ▶  $S_2 \leftarrow$  símbolo do início da fila (leitura e remoção).
- ▶ Escrever  $S_1$  na posição mais à esquerda da fila (no lugar do segundo \$, usando a Parte 2).

---

<sup>1</sup>pode ser simulado através do uso de estados no programa que implementa o algoritmo na Máquina de Post

# Teorema 3

## Movimento para a ESQUERDA

Exemplo:

- ▶ Suponha a substituição de  $f$  por  $X$ ;
- ▶ Entrada:  $fghij\#abcde$ ;
- ▶ Saída:  $\$Xghij\#abcd$ .

## Teorema 3

## Movimento para a ESQUERDA

Situação	$S_1$	$S_2$	$X$
INÍCIO	-	-	$fghij\#abcde$
Insera \$	-	-	$fghij\#abcde\$$
Insera \$	-	-	$fghij\#abcde\$\$$
Insera $X$	-	-	$fghij\#abcde\$\$X$
Remove $f$	-	-	$ghij\#abcde\$\$X$
Remove $g$	$g$	-	$hij\#abcde\$\$X$
Remove $h$	$g$	$h$	$ij\#abcde\$\$X$
Insera $g$	$g$	$h$	$ij\#abcde\$\$Xg$
Copia $S_2$ para $S_1$	$h$	$h$	$ij\#abcde\$\$Xg$
Remove $i$	$h$	$i$	$j\#abcde\$\$Xg$

## Teorema 3

## Movimento para a ESQUERDA

Situação	$S_1$	$S_2$	$X$
Inserir $h$	$h$	$i$	$j\#abcde\$\$Xgh$
Copia $S_2$ para $S_1$	$i$	$i$	$j\#abcde\$\$Xgh$
Remove $j$	$i$	$j$	$\#abcde\$\$Xgh$
Inserir $i$	$i$	$j$	$\#abcde\$\$Xghi$
Copia $S_2$ para $S_1$	$j$	$j$	$\#abcde\$\$Xghi$
Remove $\#$	$j$	$\#$	$abcde\$\$Xghi$
Inserir $j$	$j$	$\#$	$abcde\$\$Xghij$
Copia $S_2$ para $S_1$	$\#$	$\#$	$abcde\$\$Xghij$
Remove $a$	$\#$	$a$	$bcde\$\$Xghij$

## Teorema 3

## Movimento para a ESQUERDA

Situação	$S_1$	$S_2$	$X$
Inserir #	#	$a$	$bcde\$Xghij\#$
Copia $S_2$ para $S_1$	$a$	$a$	$bcde\$Xghij\#$
Remove $b$	$a$	$b$	$cde\$Xghij\#$
Inserir $a$	$a$	$b$	$cde\$Xghij\#a$
Copia $S_2$ para $S_1$	$b$	$b$	$cde\$Xghij\#a$
Remove $c$	$b$	$c$	$de\$Xghij\#a$
Inserir $b$	$b$	$c$	$de\$Xghij\#ab$
Copia $S_2$ para $S_1$	$c$	$c$	$de\$Xghij\#ab$
Remove $d$	$c$	$d$	$e\$Xghij\#ab$

## Teorema 3

## Movimento para a ESQUERDA

Situação	$S_1$	$S_2$	$X$
Inserir $c$	$c$	$d$	$e$$$Xghij\#abc$
Copia $S_2$ para $S_1$	$d$	$d$	$e$$$Xghij\#abc$
Remove $e$	$d$	$e$	$$$$Xghij\#abc$
Inserir $d$	$d$	$e$	$$$$Xghij\#abcd$
Copia $S_2$ para $S_1$	$e$	$e$	$$$$Xghij\#abcd$
Remove $\$$	$e$	$\$$	$\$Xghij\#abcd$
TÉRMINO	$e$	$\$$	$\$Xghij\#abcd$

# Teorema 3

## Movimento para a ESQUERDA

### Parte 2:

- ▶ Inserir um símbolo especial \$ no final da fila;
- ▶ Ler (e remover) o símbolo do início da fila;
- ▶ Inserir o novo símbolo no final da fila;
- ▶ Transferir os símbolos do início da fila para o final, até atingir o símbolo \$;
- ▶ Remover o símbolo \$ do início da fila.

# Teorema 3

## Movimento para a ESQUERDA

Exemplo:

- ▶ Suponha a substituição de  $a$  por  $X$ ;
- ▶ Entrada:  $abcde\#fghij$ ;
- ▶ Saída:  $Xbcde\#fghij$ .

## Teorema 3

## Movimento para a ESQUERDA

Situação	$X$
INÍCIO	$abcde\#fghij$
Inserir $\$$	$abcde\#fghij\$$
Remove $a$	$bcd\#fghij\$$
Inserir $X$	$bcd\#fghij\$X$
Remove $b$	$cd\#fghij\$X$
Inserir $b$	$cd\#fghij\$Xb$
...	...
Remove $j$	$\$Xbcde\#fghi$
Inserir $j$	$\$Xbcde\#fghij$
Remove $\$$	$Xbcde\#fghij$
TÉRMINO	$Xbcde\#fghij$

# Teorema 3

## Estados, aceitação e rejeição

- ▶ Instrução “Partida” simula o estado inicial  $q_0$ ;
- ▶ Instrução “Aceita” simula os estados finais  $q_i \in F$ ;
- ▶ Cada um dos demais estados corresponde a uma instrução “Desvio condicional”;
- ▶ A rejeição por função de transição indefinida ou movimento inválido são simuladas pela instrução “Rejeita”.

# Teorema 4

Máquina de Post  $\leq$  Máquina de Turing

Seja:

$$M = (\Sigma, D, \#)$$

uma Máquina de Post. Então, existe uma Máquina de Turing:

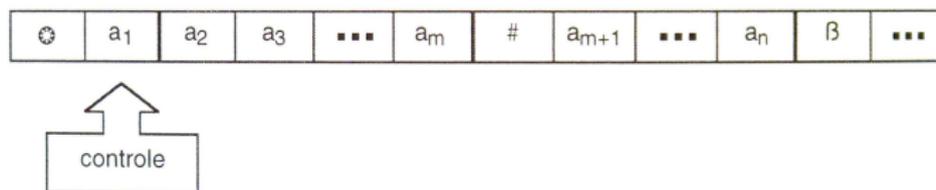
$$M' = (\Sigma, Q, \Pi, q_0, F, \{\#\}, \beta, \circ)$$

que simula  $M$ .

## Teorema 4

Variável  $X$ 

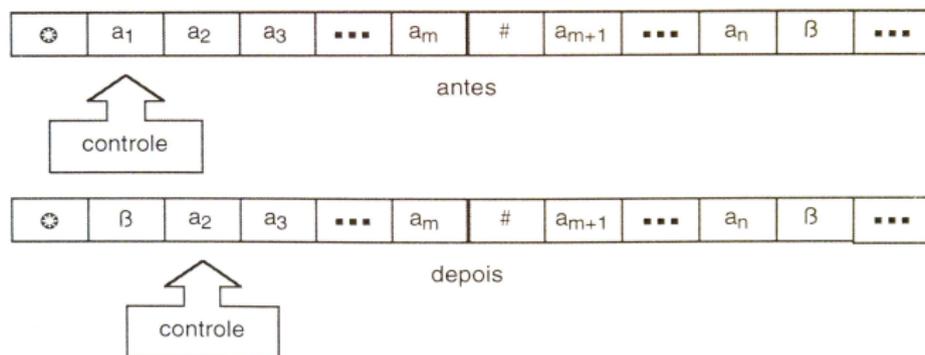
- ▶  $X$  é simulada pela fita;
- ▶ O cursor aponta para o primeiro de  $X$ ;
- ▶ Se  $X = a_1a_2a_3\dots a_m\#a_{m+1}\dots a_n$ , então a representação de  $X$  na fita é:



## Teorema 4

## Desvio condicional

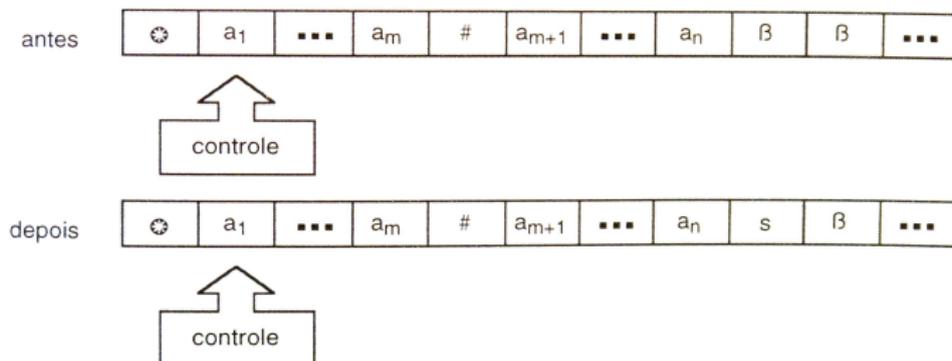
- ▶ O consumo do símbolo mais à esquerda é representado pela substituição do símbolo lido por  $\beta$  seguido do deslocamento do cursor para à direita;
- ▶ Se  $X = a_1 a_2 a_3 \dots a_m \# a_{m+1} \dots a_n$ , um teste com o símbolo  $a_1$  resulta em  $X = a_2 a_3 \dots a_m \# a_{m+1} \dots a_n$



## Teorema 4

## Atribuição

- ▶ A atribuição de um símbolo à variável  $X$  é representado pelo acréscimo de um símbolo no final da fita seguido do retorno do cursor para a posição mais à esquerda da fita que não seja  $\beta$ ;
- ▶ Se  $X = a_1a_2a_3\dots a_m\#a_{m+1}\dots a_n$ , uma atribuição com o símbolo  $s$  resulta em  $X = a_1a_2a_3\dots a_m\#a_{m+1}\dots a_ns$



# Teorema 4

## Partida, aceita e rejeita

- ▶ A instrução “Partida” é simulada pelo estado inicial  $q_0$ ;
- ▶ A instrução “Aceita” é simulada por  $q_F \in F$ ;
- ▶ A instrução “Rejeita” é simulada por movimento inválido.

# Generalidades

- ▶ Formalizada por vários autores na década de 1960;
- ▶ A memória de entrada é separada das memórias auxiliar e de saída (diferente das Máquinas de Turing e de Post);
- ▶ Fita de entrada contém a cadeia a ser analisada;
- ▶ Memória auxiliar:
  - ▶ Pilha (*first-in-last-out*);
  - ▶ Uma ou mais pilhas;
  - ▶ As pilhas não tem limitação de tamanho.
- ▶ Possui um programa associado (fluxograma):
  - ▶ Partida;
  - ▶ Parada;
  - ▶ Desvio condicional (desempilha);
  - ▶ Empilha.
- ▶ Máquina Universal.

# Definição

Uma Máquina de Pilhas é uma dupla:

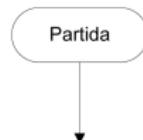
$$M = (\Sigma, D)$$

onde:

- ▶  $\Sigma = \{\sigma_1, \sigma_2, \dots, \sigma_n\}$  é o alfabeto de entrada;
- ▶  $D$  é o programa (ou fluxograma), constituído pelos componentes partida, parada, desvio condicional (desempilha) e empilha;
- ▶  $X$  representa a fita de entrada;
- ▶  $Y_i, i \geq 0$ , representa as pilhas.

# Definição

Componente “Partida”:



- ▶ Único em cada programa  $P$ ;
- ▶ Indica o início da execução de  $P$ .

# Definição

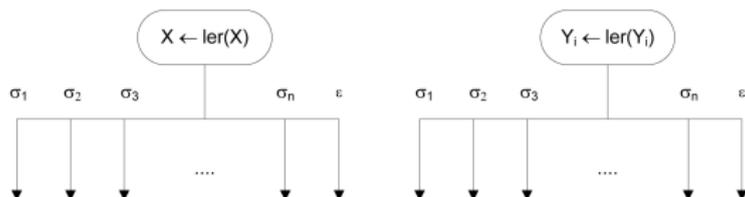
Componente “Parada”:



- ▶ Pode ser de dois tipos: parada com aceitação ou parada com rejeição;
- ▶ Múltiplas ocorrências são permitidas, sem restrições (inclusive zero ocorrências de qualquer ou de ambos os componentes).

# Definição

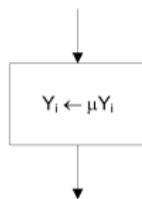
Componente “Desvio condicional (desempilha)”:



- ▶ Analisa o primeiro símbolo da entrada ou da pilha  $i$  (variáveis  $X$  e  $Y_i$ , respectivamente);
- ▶ Conforme o símbolo encontrado, desvia de acordo;
- ▶ O símbolo encontrado é removido do início da entrada ou da pilha;
- ▶ Se  $|\Sigma| = n$ , devem ser previstos  $n + 1$  desvios;
- ▶ Desvio também para o caso de  $X$  ou  $Y_i$  conter a cadeia vazia ( $\epsilon$ ).

# Definição

Componente “Empilha”:



- ▶ Insere o símbolo  $\mu$  no topo da pilha  $Y_i$ ;
- ▶  $\mu \in \Sigma$ .

Exemplo —  $a^n b^n$ 

Estratégia:

- ▶ Usa uma única pilha;
- ▶ Ler os símbolos  $a$  da entrada ( $X$ ), empilhando os mesmos em seguida ( $Y$ );
- ▶ Quando encontrar o primeiro  $b$  na entrada, começar a desempilhar os símbolos  $a$ , garantindo que para cada  $b$  em  $X$  existe um  $a$  em  $Y$ ;
- ▶ Se a seqüência de símbolos  $b$  da entrada ( $X$ ) acabar juntamente com a seqüência de símbolos  $a$  da pilha ( $Y$ ), então ACEITA; senão, REJEITA.



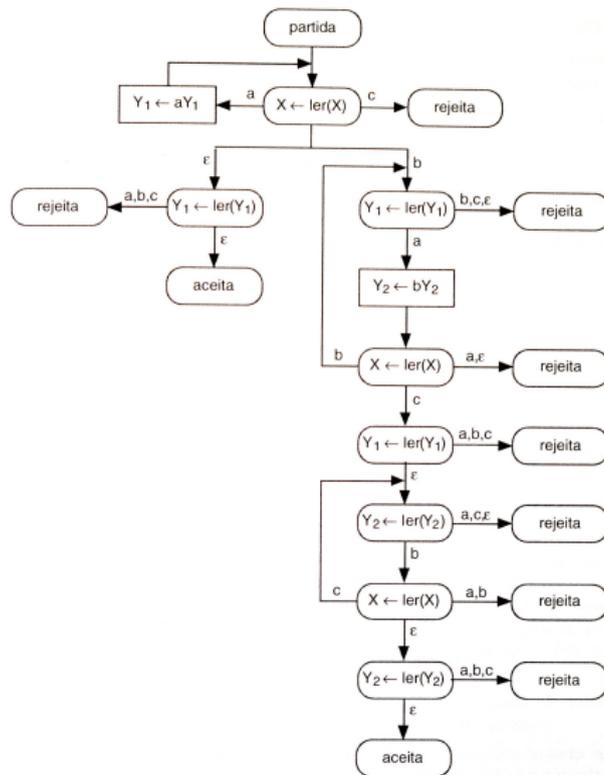
Exemplo —  $a^n b^n$ 

$X$	$Y$
$aaabbb$	$\epsilon$
$aabbb$	$\epsilon$
$aabbb$	$a$
$abbb$	$a$
$abbb$	$aa$
$bbb$	$aa$
$bbb$	$aaa$
$bb$	$aaa$
$bb$	$aa$
$b$	$aa$
$b$	$a$
$\epsilon$	$a$
$\epsilon$	$\epsilon$

Exemplo —  $a^n b^n c^n$ 

Estratégia:

- ▶ Usa duas pilhas,  $Y_1$  e  $Y_2$ ;
- ▶ Remover símbolos  $a$  da entrada, inserindo-os na pilha  $Y_1$ ;
- ▶ Para cada símbolo  $b$  da entrada, remover o mesmo de  $X$ , remover um símbolo  $a$  de  $Y_1$  e inserir um símbolo  $b$  em  $Y_2$ ;
- ▶ Deve-se garantir que as quantidades de  $a$  em  $Y_1$  e  $b$  na entrada sejam idênticas;
- ▶ Para cada símbolo  $c$  da entrada, remover o mesmo de  $X$  e remover um símbolo  $b$  da  $Y_2$ ;
- ▶ Se as quantidade de  $b$  em  $Y_2$  e  $c$  em  $X$  forem iguais, ACEITA; senão REJEITA.

Exemplo —  $a^n b^n c^n$ 

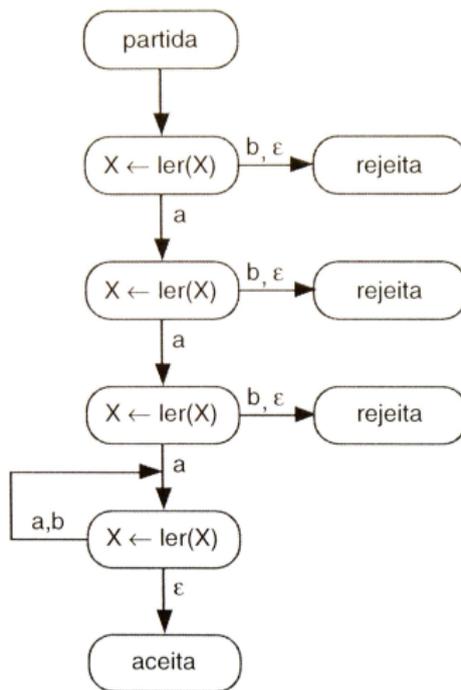
Exemplo —  $a^n b^n c^n$ 

$X$	$Y_1$	$Y_2$
$aabbcc$	$\epsilon$	$\epsilon$
$abbcc$	$\epsilon$	$\epsilon$
$abbcc$	$a$	$\epsilon$
$bbcc$	$a$	$\epsilon$
$bbcc$	$aa$	$\epsilon$
$bcc$	$aa$	$\epsilon$
$bcc$	$a$	$\epsilon$
$bcc$	$a$	$b$
$cc$	$a$	$b$
$cc$	$\epsilon$	$b$
$cc$	$\epsilon$	$bb$
$c$	$\epsilon$	$bb$
$c$	$\epsilon$	$b$
$\epsilon$	$\epsilon$	$b$
$\epsilon$	$\epsilon$	$\epsilon$

Exemplo —  $aaa(a|b)^*$ 

Estratégia:

- ▶ Não usa pilha;
- ▶ Verifica se os três primeiros símbolos da entrada  $X$  são  $a$ ;
- ▶ Consome os demais símbolos da entrada;
- ▶ Quando esgotar a cadeia de entrada, ACEITA. Se as condições anteriores não forem verificadas, REJEITA.

Exemplo —  $aaa(a|b)^*$ 

Exemplo —  $aaa(a|b)^*$ 

$$\begin{array}{l} X \\ \hline aaabc \\ aabc \\ abc \\ bc \\ c \\ \epsilon \end{array}$$

# Quantidade de pilhas

A classe de linguagens representadas por Máquinas de Pilhas depende da quantidade de pilhas que ela possui:

- ▶ Nenhuma pilha: corresponde ao autômato finito, capaz de reconhecer a classe das linguagens regulares;
- ▶ Uma pilha: corresponde ao autômato de pilha, capaz de reconhecer a classe das linguagens livres de contexto;
- ▶ Duas pilhas: corresponde à Máquina de Turing, capaz de aceitar a classe das linguagens recursivamente enumeráveis;
- ▶ Três ou mais pilhas: podem sempre ser simuladas por uma máquina com apenas duas pilhas.

# Generalidades

- ▶ Similar à Máquina com Duas Pilhas;
- ▶ No lugar de um diagrama de fluxos usa-se um diagrama de estados;
- ▶ Componentes:
  - ▶ Fita de entrada;
  - ▶ Duas pilhas;
  - ▶ Máquina de estados.
- ▶ Máquina Universal.

# Fita de entrada

- ▶ Finita;
- ▶ Contém a cadeia a ser analisada;
- ▶ Leitura apenas;
- ▶ Deslocamento do cursor para a direita apenas;
- ▶ A leitura provoca o deslocamento do cursor;
- ▶ Teste se a entrada foi esgotada;
- ▶ Leitura opcional.

# Pilhas

- ▶ Tamanho ilimitado;
- ▶ Usadas como memória auxiliar;
- ▶ Leitura/escrita;
- ▶ A leitura remove o símbolo consultado (topo da pilha);
- ▶ Cada pilha é acessada por uma cabeça de leitura/escrita independente;
- ▶ Teste se a pilha está vazia;
- ▶ Leitura opcional.

# Definição

Um Autômato com Duas Pilhas é uma sextupla:

$$M = (\Sigma, Q, \Pi, q_0, F, V)$$

onde:

- ▶  $\Sigma$  é o alfabeto de entrada;
- ▶  $Q$  é o conjunto de estados;
- ▶  $\Pi$  é a função de transição:

$$\Pi : Q \times (\Sigma \cup \{\epsilon, ?\}) \times (V \cup \{\epsilon, ?\}) \times (V \cup \{\epsilon, ?\}) \rightarrow Q \times (V \cup \{\epsilon\}) \times (V \cup \{\epsilon\})$$

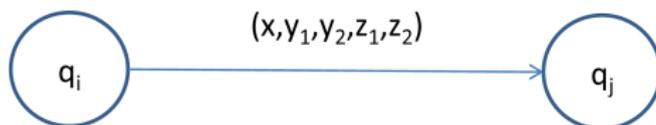
- ▶  $q_0 \in Q$  é o estado inicial;
- ▶  $F \subseteq Q$  é o conjunto de estados finais;
- ▶  $V$  é o alfabeto auxiliar.

## Diagrama de estados

Se:

$$\Pi(q_i, x, y_1, z_1) = (q_j, y_2, z_2)$$

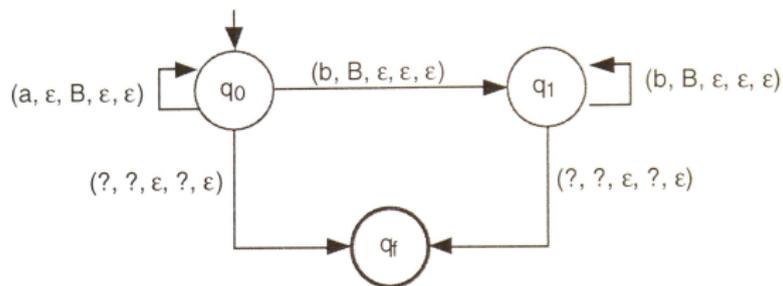
então:



# ? e $\epsilon$ em $\Pi$

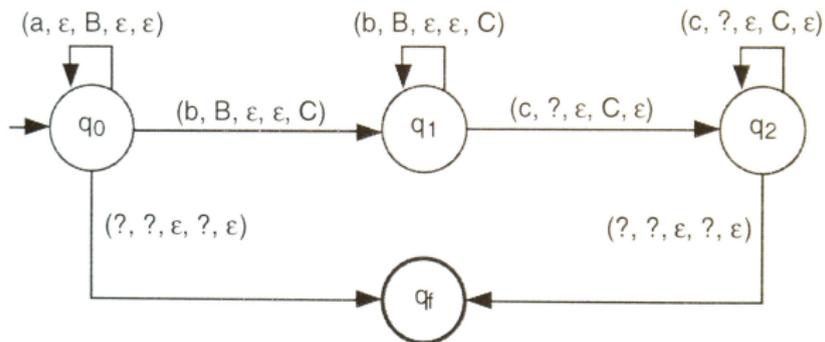
Seja  $\Pi(q_i, x, y_1, z_1) = (q_j, y_2, z_2)$ . Então:

- ▶ Se  $x = \epsilon$ , não lê símbolo da fita de entrada e não desloca o cursor;
- ▶ Se  $x = ?$ , testa se a cadeia de entrada está esgotada;
- ▶ Se  $y_1 = \epsilon$ , não lê símbolo da primeira pilha e não desempilha símbolo;
- ▶ Se  $y_1 = ?$ , testa se a primeira pilha está vazia;
- ▶ Se  $z_1 = \epsilon$ , não lê símbolo da segunda pilha e não desempilha símbolo;
- ▶ Se  $z_1 = ?$ , testa se a segunda pilha está vazia;
- ▶ Se  $y_2 = \epsilon$ , mantém a primeira pilha inalterada;
- ▶ Se  $z_2 = \epsilon$ , mantém a segunda pilha inalterada.

Exemplo —  $a^n b^n$ 

Exemplo —  $a^n b^n$ 

Estado	Entrada	Primeira pilha
$q_0$	$aaabbb$	$\epsilon$
$q_0$	$aabbb$	$B$
$q_0$	$abbb$	$BB$
$q_0$	$bbb$	$BBB$
$q_1$	$bb$	$BB$
$q_1$	$b$	$B$
$q_1$	$\epsilon$	$\epsilon$
$q_f$		

Exemplo —  $a^n b^n c^n$ 

Exemplo —  $a^n b^n c^n$ 

Estado	Entrada	Primeira pilha	Segunda pilha
$q_0$	$aabbcc$	$\epsilon$	$\epsilon$
$q_0$	$abbcc$	$B$	$\epsilon$
$q_0$	$bbcc$	$BB$	$\epsilon$
$q_1$	$bcc$	$B$	$C$
$q_1$	$cc$	$\epsilon$	$CC$
$q_2$	$c$	$\epsilon$	$C$
$q_2$	$\epsilon$	$\epsilon$	$\epsilon$
$q_f$			

# Teorema 5

Máquina de Turing  $\leq$  Autômato com Duas Pilhas

Toda Máquina de Turing pode ser simulada por algum Autômato com Duas Pilhas.

- ▶ A primeira pilha ( $P_1$ ) simula o conteúdo da fita de entrada situado à esquerda da cabeça de leitura/escrita;
- ▶ A segunda pilha ( $P_2$ ) simula o conteúdo da fita de entrada situado à direita da cabeça de leitura/escrita, incluindo o símbolo corrente;
- ▶ A cadeia é copiada da fita de entrada inicialmente para  $P_1$ , e depois desta para  $P_2$ ;
- ▶ Marcadores de fundo de pilha \$ em  $P_2$  ( $P_1$ ) são usados para simular células em branco à direita (esquerda) do último (primeiro) símbolo diferente de branco presente na fita.

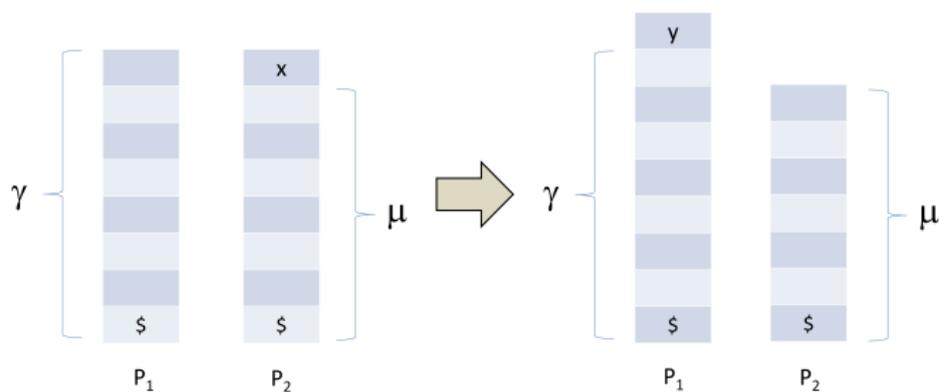
# Teorema 5

## Deslocamento à direita

- ▶ Considere  $\Pi(q_i, x) = (q_j, y, D)$ ;
- ▶ Considere  $P_1 = \gamma$  e  $P_2 = \mu x$  (topo à direita);
- ▶ O autômato deve executar os movimentos necessários para que  $P_1 = \gamma y$  e  $P_2 = \mu$ ;
- ▶ Portanto,  $(\gamma, q_i, x\mu^R) \vdash (\gamma y, q_j, \mu^R)$

## Teorema 5

## Deslocamento à direita



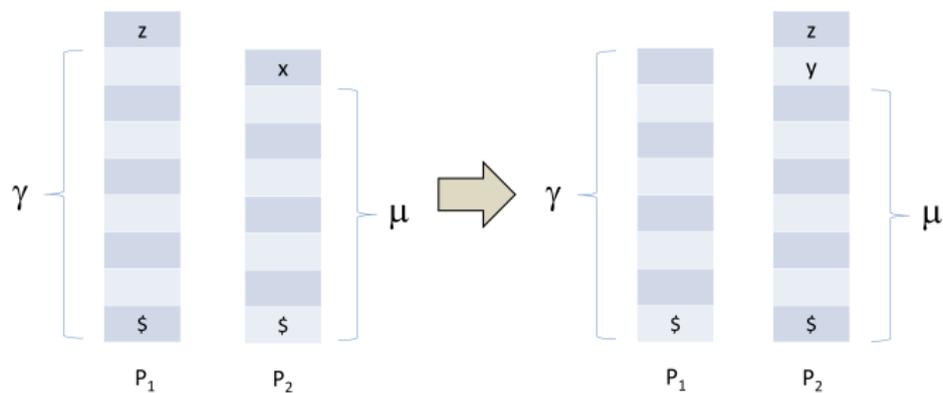
## Teorema 5

## Deslocamento à esquerda

- ▶ Considere  $\Pi(q_i, x) = (q_j, y, E)$ ;
- ▶ Considere  $P_1 = \gamma z$  e  $P_2 = \mu x$  (topo à direita);
- ▶ O autômato deve executar os movimentos necessários para que  $P_1 = \gamma$  e  $P_2 = \mu y z$ ;
- ▶ Portanto,  $(\gamma z, q_i, x \mu^R) \vdash (\gamma, q_j, z y \mu^R)$

## Teorema 5

## Deslocamento à esquerda



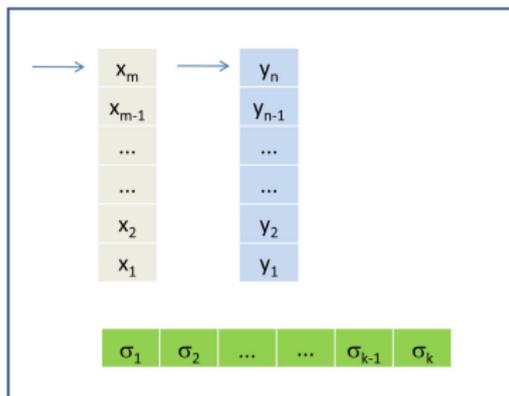
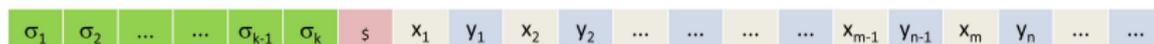
# Teorema 6

Autômato com Duas Pilhas  $\leq$  Máquina de Turing

Todo Autômato com Duas Pilhas pode ser simulado por alguma Máquina de Turing.

- ▶ A cadeia de entrada ocupa as primeiras posições da fita da Máquina de Turing
- ▶ A primeira pilha ( $P_1$ ) é simulada nas posições ímpares da fita da Máquina de Turing, após a cadeia de entrada;
- ▶ A segunda pilha ( $P_2$ ) é simulada nas posições pares da fita da Máquina de Turing, após a cadeia de entrada.

## Teorema 6

Autômato com Duas Pilhas  $\leq$  Máquina de Turing(se  $k$  ímpar)

# Conceito

Iremos explorar variações sobre as Máquinas de Turing, mostrando que todas elas podem ser simuladas pela Máquina de Turing básica:

- ▶ Fita de entrada com múltiplas trilhas;
- ▶ Não-determinismo;
- ▶ Múltiplas fitas de entrada;
- ▶ Fita limitada à esquerda.

Além de reforçar a noção da Máquina de Turing como uma Máquina Universal, tais variações serão úteis na demonstração de alguns teoremas que serão vistos mais adiante.

# Definição

Adotaremos uma definição um pouco diferente para Máquina de Turing, porém equivalente à anteriormente vista. Ela corresponde à definição utilizada em Hopcroft07:

- ▶ A fita é infinita em ambos os sentidos:  
Esse caso será analisado mais adiante.
- ▶ Não há marcador de início de fita:  
Pode ser simulado deslocando a cadeia de entrada uma posição para a direita e depois inserindo o marcador de início de fita na primeira posição.

# Definição



# Formalização

Uma Máquina de Turing (determinística) é uma 7-upla:

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

onde:

- ▶  $Q$  é o conjunto (finito) de estados;
- ▶  $\Sigma$  é o alfabeto de entrada;
- ▶  $\Gamma$  é o conjunto de símbolos da fita,  $\Sigma \subseteq \Gamma$ ;
- ▶  $\delta$  é a função de transição:

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

# Formalização

Uma Máquina de Turing (determinística) é uma 7-upla:

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

onde:

- ▶  $q_0 \in Q$  é o estado inicial;
- ▶  $B$  representa o símbolo branco, usado para preencher todas as posições da fita não inicializadas com símbolos da cadeia de entrada;  $B \in (\Gamma - \Sigma)$ ;
- ▶  $F$  é o conjunto de estados finais,  $F \subseteq Q$ .

# Complexidade no tempo

## Máquina de Turing determinística

A “complexidade no tempo” ou “tempo de execução” de uma Máquina de Turing  $M$  com uma entrada  $w$  é definida como:

- ▶ A quantidade de movimentos que  $M$  executa com a entrada  $w$  até parar (aceitando ou rejeitando);
- ▶ Se  $M$  não pára com a entrada  $w$ , o tempo é infinito.

# Complexidade no tempo

## Máquina de Turing determinística

A “complexidade no tempo” ou “tempo de execução” de uma Máquina de Turing  $M$  é definida como:

- ▶ A função  $T(n)$ ;
- ▶  $n$  representa um certo comprimento da cadeia de entrada;
- ▶  $T(n)$  é a quantidade de movimentos que são executados quando são consideradas todas as possíveis cadeias  $w$  de comprimento  $n$ .

# Complexidade no tempo

Independentemente de fatores ou coeficientes, considera-se:

- ▶ Problemas “tratáveis” são aqueles que possuem tempo de execução polinomial, ou seja,  $T(n) = O(n^k)$ , para algum  $k$ ;
- ▶ Problemas “intratáveis” são aqueles que possuem tempo de execução exponencial, ou seja,  $T(n) = O(k^n)$ , para algum  $k$ ;
- ▶ Exceções à parte, funções exponenciais crescem muito mais rapidamente do que funções polinomiais;
- ▶ Problemas “tratáveis” geralmente possuem soluções viáveis em computadores; problemas “intratáveis” geralmente não possuem.

# Fita de entrada com múltiplas trilhas

## Conceito

...		A							...
...		B							...
...		...							...
...		Z							...



# Fita de entrada com múltiplas trilhas

## Formalização

- ▶ Para uma fita de entrada com  $n$  trilhas:

$$\delta : Q \times \underbrace{\Gamma \times \Gamma \dots \times \Gamma}_{\Gamma^n} \rightarrow Q \times \underbrace{\Gamma \times \Gamma \dots \times \Gamma}_{\Gamma^n} \times \{L, R\}$$

Em cada estado, o controle finito consulta o símbolo armazenado em cada uma das trilhas individualmente, providencia uma substituição para cada um deles, e desloca a cabeça de leitura/escrita para a direita ou para a esquerda;

- ▶ Se  $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$  é uma Máquina de Turing com  $n$  trilhas, essa máquina pode ser simulada por  $M'$  cujo conjunto de símbolos da fita é  $\Gamma' = \Gamma^n$ ; os demais elementos de  $M$  permanecem inalterados em  $M'$ ;
- ▶ Cada elemento de  $\Gamma^n$  é considerado um novo símbolo, e dessa forma um único símbolo é consultado/gravado de cada vez, como numa máquina com apenas uma trilha.

# Fita de entrada com múltiplas trilhas

## Exemplo

Suponha uma Máquina de Turing com 2 trilhas e:

$$\Gamma = \{a, X, B\}$$

Então:

$$\Gamma' = \{(a, a), (a, X), (a, B), (X, a), (X, X), (X, B), (B, a), (B, X), (B, B)\}$$

# Não-determinismo

## Definição

Uma Máquina de Turing  $M$  é dita “não-determinística” se existir mais de uma possibilidade de movimentação a partir de uma mesma configuração. Formalmente:

$$\delta : Q \times \Gamma \rightarrow 2^{Q \times \Gamma \times \{L,R\}}$$

# Não-determinismo

## Linguagem definida

Seja  $M$  uma Máquina de Turing  $M$  não-determinística e  $w \in \Sigma^*$ . São considerados três casos, que cobrem todas as situações possíveis:

- ▶  $w \in ACEITA(M)$  se e somente se existe pelo menos uma seqüência de movimentos que conduz  $M$  a um estado final com a cadeia  $w$ ;
- ▶  $w \in REJEITA(M)$  se e somente se todas as seqüências de movimentos de  $M$  com a cadeia  $w$  conduzem à configurações de parada não-finais;
- ▶  $w \in LOOP(M)$  se e somente se:
  - ▶ Não existe nenhuma seqüência de movimentos que conduza  $M$  a um estado final com a cadeia  $w$ ;
  - ▶ Existe pelo menos uma seqüência de movimentos que fazem com que  $M$  entre em loop com a cadeia  $w$ .

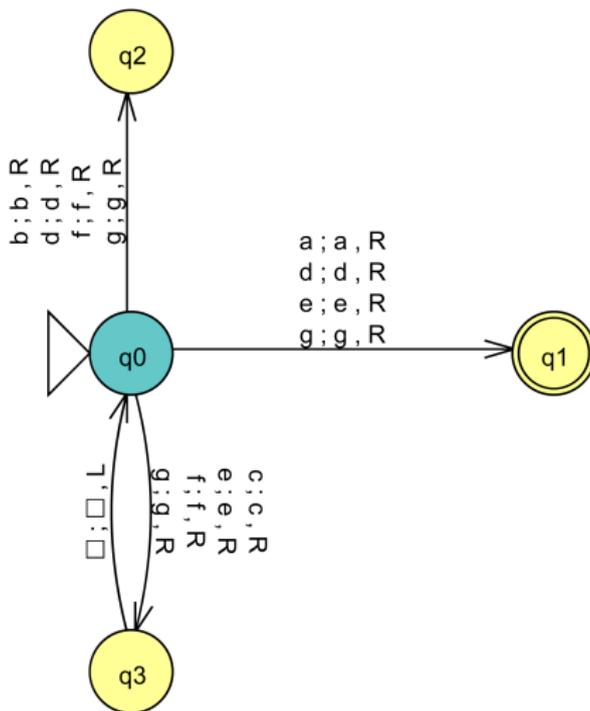
# Não-determinismo

## Exemplo

- ▶ A Máquina de Turing da figura seguinte é não-determinística e possui  $\Sigma = \{a, b, c, d, e, f, g\}$ ;
- ▶ São consideradas cadeias de entrada que provocam todas as combinações possíveis entre as situações de aceitação, rejeição e loop, inclusive combinações duas a duas e as três simultaneamente;
- ▶ O resultado serve para ilustrar a determinação de  $ACEITA(M)$ ,  $REJEITA(M)$  e  $LOOP(M)$  em Máquinas de Turing não-determinísticas.

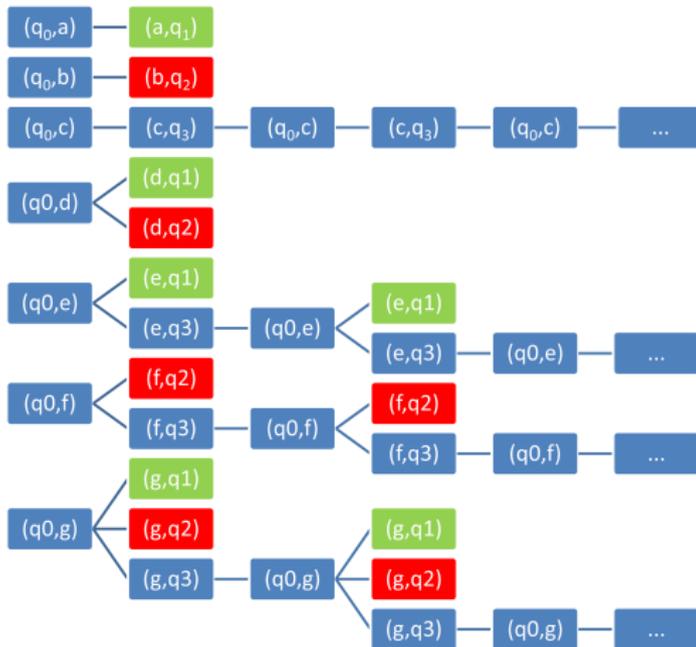
# Não-determinismo

## Exemplo



# Não-determinismo

## Exemplo



# Não-determinismo

## Exemplo

	ACEITA	REJEITA	LOOP	
<i>a</i>	✓			ACEITA
<i>b</i>		✓		REJEITA
<i>c</i>			✓	LOOP
<i>d</i>	✓	✓		ACEITA
<i>e</i>	✓		✓	ACEITA
<i>f</i>		✓	✓	LOOP
<i>g</i>	✓	✓	✓	ACEITA

# Não-determinismo

## Exemplo

Portanto,  $M$  particiona  $\Sigma^*$  nos seguintes conjuntos:

- ▶  $ACEITA(M) = \{a, d, e, g, \dots\}$ ;
- ▶  $REJEITA(M) = \{b, \dots\}$ ;
- ▶  $LOOP(M) = \{c, f, \dots\}$ ;

# Não-determinismo

## Combinações de casos

Seja  $M$  e  $w$ . As diversas seqüências de movimentação de  $M$  com  $w$  podem ser classificadas em *ACEITA*, *REJEITA* e *LOOP*. Considere a quantidade de ocorrências de cada uma delas no conjunto de todas as ocorrências como sendo:

- ▶ “pelo menos um” ( $\geq 1$ );
- ▶ “nenhuma” (0), ou
- ▶ “todas” (*all*).

As tabelas seguintes mostram as várias combinações possíveis para os valores dessas três variáveis, e como a cadeia  $w$  deve ser considerada do ponto de vista de aceitação, rejeição ou loop. Das 27 combinações possíveis ( $3^3$ ), apenas 10 são válidas.

# Não-determinismo

## Combinações de casos

ACEITA	REJEITA	LOOP	
$\geq 1$	$\geq 1$	$\geq 1$	ACEITA
$\geq 1$	$\geq 1$	0	ACEITA
$\geq 1$	$\geq 1$	all	-
$\geq 1$	0	$\geq 1$	ACEITA
$\geq 1$	0	0	ACEITA
$\geq 1$	0	all	-
$\geq 1$	all	$\geq 1$	-
$\geq 1$	all	0	-
$\geq 1$	all	all	-

# Não-determinismo

## Combinações de casos

ACEITA	REJEITA	LOOP	
0	$\geq 1$	$\geq 1$	LOOP
0	$\geq 1$	0	REJEITA
0	$\geq 1$	all	-
0	0	$\geq 1$	LOOP
0	0	0	-
0	0	all	LOOP
0	all	$\geq 1$	-
0	all	0	REJEITA
0	all	all	-

# Não-determinismo

## Combinações de casos

ACEITA	REJEITA	LOOP	
all	$\geq 1$	$\geq 1$	-
all	$\geq 1$	0	-
all	$\geq 1$	all	-
all	0	$\geq 1$	-
all	0	0	ACEITA
all	0	all	-
all	all	$\geq 1$	-
all	all	0	-
all	all	all	-

# Não-determinismo

## Equivalência

### Teorema:

Toda Máquina de Turing não-determinística  $M$  pode ser simulada por uma Máquina de Turing determinística  $M'$  equivalente. Ou seja:

- ▶  $ACEITA(M') = ACEITA(M)$ ;
- ▶  $REJEITA(M') = REJEITA(M)$ ;
- ▶  $LOOP(M') = LOOP(M)$ .

# Não-determinismo

## Equivalência

### Método:

- ▶ Simular as configurações de  $M$ , representando-as na fita de  $M'$ ;
- ▶ As configurações são delimitadas pelo símbolo especial  $*$ ;
- ▶ A configuração corrente é marcada pelo símbolo especial  $X$ , que ocupa o lugar do  $*$  situado à esquerda da mesma;
- ▶ A função de transição  $\delta$  de  $M$  está armazenada no controle de  $M'$ ;
- ▶ Considerar a árvore de todos os caminhos possíveis;
- ▶ Fazer uma busca em largura para determinar se alguma configuração é final e aceitar quando encontrar;
- ▶ Parar e rejeitar quando não houverem novas configurações a serem consideradas;
- ▶ Parar e aceitar quando o estado corrente for final.

# Não-determinismo

## Equivalência

### Algoritmo:

1. A fita de  $M'$  contém, inicialmente, a configuração inicial de  $M$  com a cadeia  $w$ ;
2. Essa configuração é marcada como sendo a configuração corrente;
3.  $M'$  analisa a configuração corrente para determinar se o estado corrente é final;
4. Em caso afirmativo,  $M'$  pára e aceita  $w$ ; em caso negativo,  $M'$  analisa a configuração corrente para determinar o estado corrente  $q_i$  e o símbolo corrente  $x$ ;
5.  $M'$  insere, no final da cadeia de entrada, tantas novas configurações quantos sejam os elementos de  $\delta(q_i, x)$ ;

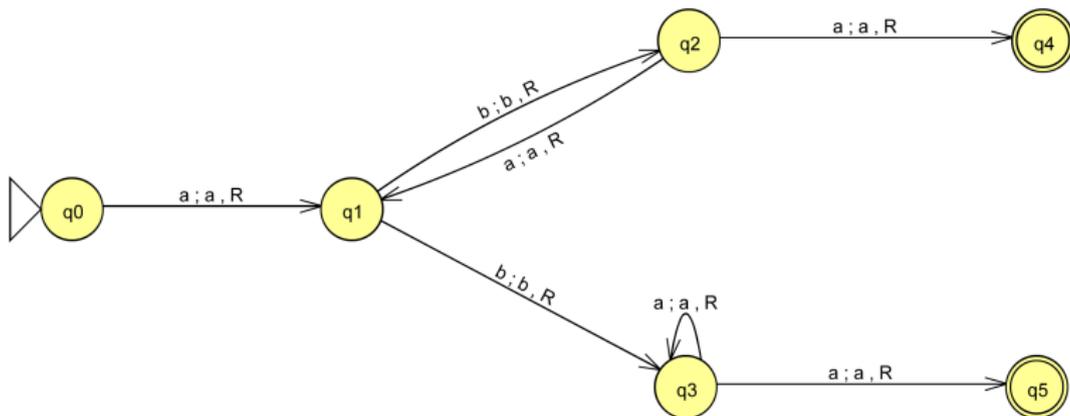
# Não-determinismo

## Equivalência

6. Cada uma dessas configurações é modificada para refletir a aplicação de uma particular transição:
  - ▶ Suponha que  $(\alpha, q_i, x\gamma)$  seja a configuração corrente e que  $\delta(q_i, x) = \{(q_j, y, R), \dots, (q_m, z, R)\}$ ;
  - ▶ As novas configurações são  $(\alpha y, q_j, \gamma), \dots, (\alpha z, q_m, \gamma)$ ;
  - ▶ De maneira análoga se os deslocamentos forem à esquerda.
7.  $M'$  procura a próxima configuração na fita de entrada;
8. Caso não exista,  $M'$  pára e rejeita  $w$ ; caso exista, vá para 2.

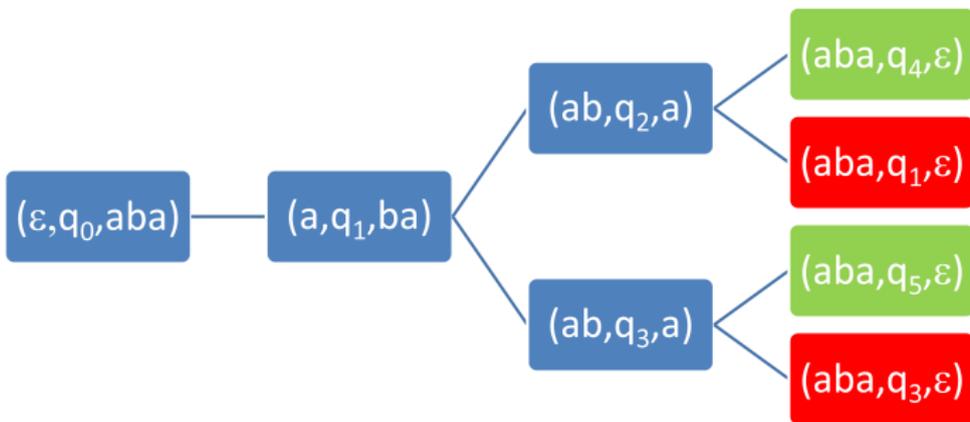
# Não-determinismo

## Exemplo



# Não-determinismo

## Exemplo



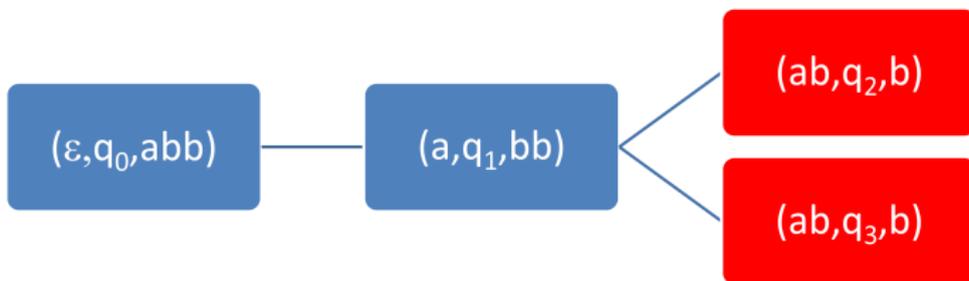
# Não-determinismo

## Exemplo

X	q <sub>0</sub>	a	b	a	*															
*	q <sub>0</sub>	a	b	a	X	a	q <sub>1</sub>	b	a	*										
*	q <sub>0</sub>	a	b	a	*	a	q <sub>1</sub>	b	a	X	a	b	q <sub>2</sub>	a	*	a	b	q <sub>3</sub>	a	
*																				
*	q <sub>0</sub>	a	b	a	*	a	q <sub>1</sub>	b	a	*	a	b	q <sub>2</sub>	a	X	a	b	q <sub>3</sub>	a	
*	a	b	a	q <sub>4</sub>	*	a	b	a	q <sub>1</sub>	*										
*	q <sub>0</sub>	a	b	a	*	a	q <sub>1</sub>	b	a	*	a	b	q <sub>2</sub>	a	*	a	b	q <sub>3</sub>	a	
X	a	b	a	q <sub>4</sub>	*	a	b	a	q <sub>1</sub>	*	a	b	a	q <sub>5</sub>	*	a	b	a	q <sub>3</sub>	

# Não-determinismo

## Exemplo



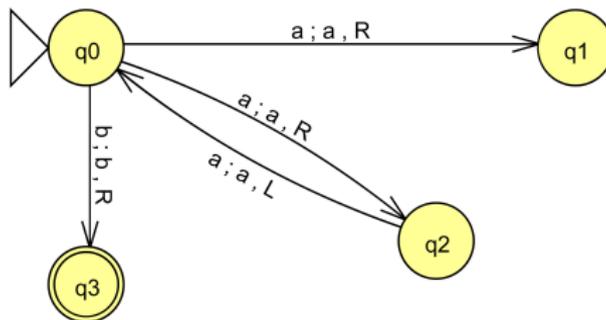
# Não-determinismo

## Exemplo

X	q <sub>0</sub>	a	b	b	*														
*	q <sub>0</sub>	a	b	a	X	a	q <sub>1</sub>	b	b	*									
*	q <sub>0</sub>	a	b	a	*	a	q <sub>1</sub>	b	a	X	a	b	q <sub>2</sub>	b	*	a	b	q <sub>3</sub>	b
*																			

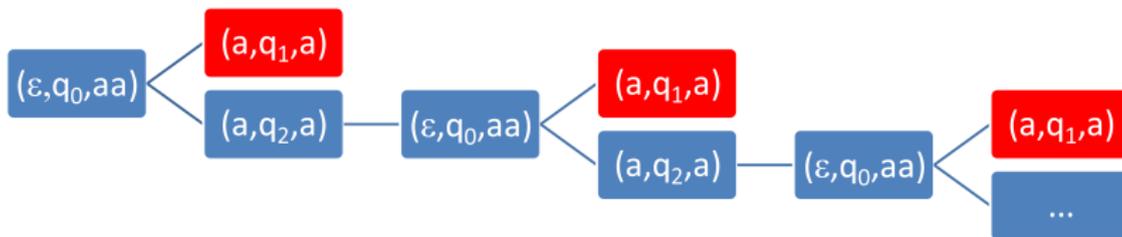
# Não-determinismo

## Exemplo



# Não-determinismo

## Exemplo



# Não-determinismo

## Exemplo

X	q <sub>0</sub>	a	a	*															
*	q <sub>0</sub>	a	a	X	a	q <sub>1</sub>	a	*	a	q <sub>2</sub>	a	*							
*	q <sub>0</sub>	a	a	*	a	q <sub>1</sub>	a	X	a	q <sub>2</sub>	a	*	q <sub>0</sub>	a	a	*			
*	q <sub>0</sub>	a	a	*	a	q <sub>1</sub>	a	*	a	q <sub>2</sub>	a	X	q <sub>0</sub>	a	a	*	a	q <sub>1</sub>	a
*	a	q <sub>2</sub>	a	*															
*	q <sub>0</sub>	a	a	*	a	q <sub>1</sub>	a	*	a	q <sub>2</sub>	a	*	q <sub>0</sub>	a	a	X	a	q <sub>1</sub>	a
*	a	q <sub>2</sub>	a	*	q <sub>0</sub>	a	a	*											
*	q <sub>0</sub>	a	a	*	a	q <sub>1</sub>	a	*	a	q <sub>2</sub>	a	*	q <sub>0</sub>	a	a	*	a	q <sub>1</sub>	a
X	a	q <sub>2</sub>	a	*	q <sub>0</sub>	a	a	*	a	q <sub>1</sub>	a	*	a	q <sub>2</sub>	a	*			

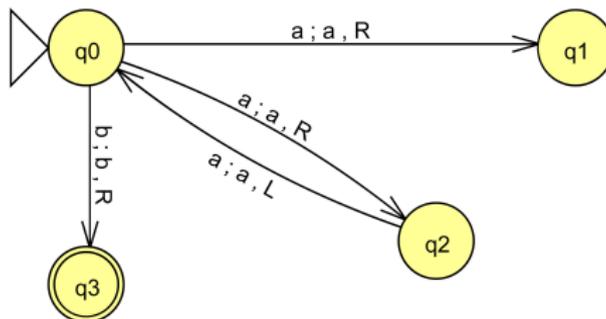
# Não-determinismo

Conversão — início

- ▶ Seja  $M$  especificada no slide seguinte;
- ▶ Seja  $w = aa$ ;
- ▶ A configuração inicial de  $M$  é  $(\epsilon, q_0, aa)$ ;
- ▶ Os estados  $q_0, q_1, q_2$  e  $q_3$  de  $M$  são denotados respectivamente 0, 1, 2 e 3 em  $M'$ ;
- ▶ A configuração inicial de  $M'$  é  $(\epsilon, q_0, X0aa*)$ ;
- ▶  $M'$  procura a configuração à direita do símbolo  $X$ ;
- ▶  $M'$  determina o estado corrente e o símbolo corrente de  $M$ .

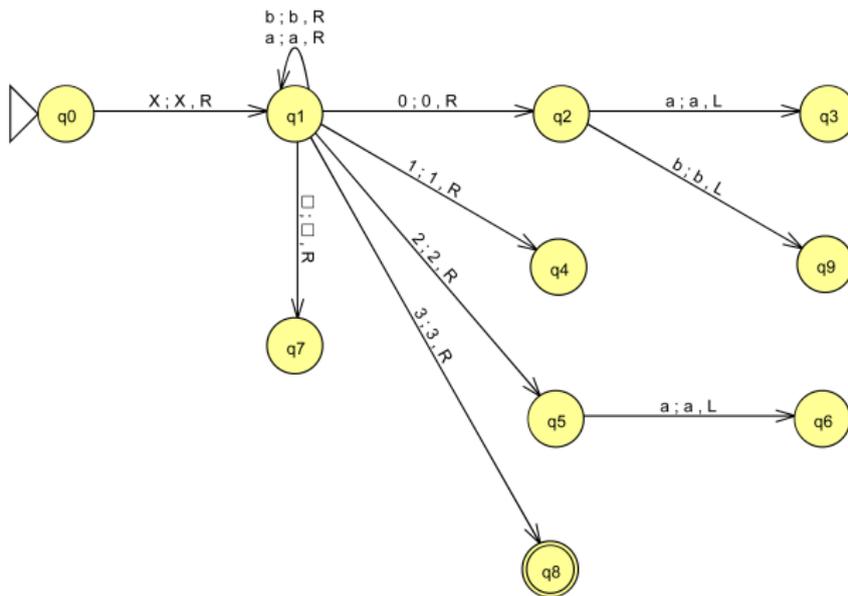
# Não-determinismo

Versão não-determinística



# Não-determinismo

Versão não-determinística equivalente (parcial)



# Não-determinismo

Conversão — término

$M'$  está especificado parcialmente:

- a) A partir de  $q_3$ , gravar duas novas configurações no final da fita, uma substituindo  $0a$  na configuração corrente por  $a1$  e outra substituindo  $0a$  por  $a2$ ; terminar ambas com  $*$ ; ir para (d);
- b) A partir de  $q_9$ , gravar uma nova configuração no final da fita, substituindo  $0b$  na configuração corrente por  $b3$ ; terminar com  $*$ ; ir para (d);
- c) A partir de  $q_6$ , gravar uma nova configuração no final da fita, substituindo  $a2$  na configuração corrente por  $0a$ ; terminar com  $*$ ; ir para (d);

# Não-determinismo

## Conversão — término

- d) Procurar o  $X$  à esquerda e substituir por  $*$ ; depois disso, procurar o primeiro  $*$  à direita e substituir por  $X$ ; ir para (e);
- e) Deslocar a cabeça até o  $X$  e ir para o estado  $q_0$ ;
- f) Nos estados  $q_2$  e  $q_5$ , se a entrada corrente for  $*$  ou branco, executar os passos (d) e (e).

# Não-determinismo

## Conclusões

- ▶ Se  $M$  alcança alguma configuração de aceitação para  $w$ ,  $M'$  aceita  $w$  e pára;
- ▶ Se  $M$  pára em configurações não-finais em todos os caminhos,  $M'$  pára e rejeita  $w$ ;
- ▶ Se  $M$  entra em loop em algum caminho,  $M'$  entra em loop.
  
- ▶ Se  $M'$  alcança uma configuração de aceitação para  $w$ , existe um caminho de aceitação para  $w$  em  $M$ ;
- ▶ Se  $M'$  pára e rejeita  $w$ ,  $M$  pára em configurações não-finais em todos os caminhos com a entrada  $w$ ;
- ▶ Se  $M'$  entra em loop em algum caminho,  $M$  entra em loop em algum caminho.

# Complexidade no tempo

## Máquina de Turing não-determinística

A “complexidade no tempo” ou “tempo de execução” de uma Máquina de Turing  $M$  com uma entrada  $w$  é definida como:

- ▶ A quantidade máxima de movimentos que  $M$  executa com a entrada  $w$  até parar (aceitando ou rejeitando), considerando todas as seqüências possíveis de movimentação;
- ▶ Se  $M$  não pára com a entrada  $w$ , o tempo é infinito.

# Complexidade no tempo

## Máquina de Turing não-determinística

A “complexidade no tempo” ou “tempo de execução” de uma Máquina de Turing  $M$  é definida como:

- ▶ A função  $T(n)$ ;
- ▶  $n$  representa um certo comprimento da cadeia de entrada;
- ▶  $T(n)$  é a quantidade máxima de movimentos executados quando são consideradas todas as possíveis cadeias  $w$  de comprimento  $n$  e, para cada uma delas, todas as possíveis seqüências de movimentação.

# Não-determinismo

## Conclusões

- ▶ Considere que  $M$  executa  $n$  movimentos na seqüência mais longa;
- ▶ Considere que o maior número de transições em qualquer configuração de  $M$  é  $m$ ;
- ▶ Após a execução do primeiro movimento de  $M$  (a partir da configuração inicial) haverão, no máximo,  $m$  configurações seguintes;
- ▶ Após a execução do segundo movimento de  $M$ , haverão, no máximo,  $m * m$  configurações seguintes;
- ▶ Após a execução do  $n$ -ésimo movimento de  $M$ , haverão, no máximo,  $m^n$  configurações seguintes;

# Não-determinismo

## Conclusões

- ▶ Portanto, o total de configurações alcançadas por  $M$  é  $1 + m + m^2 + \dots + m^n$ ;
- ▶  $1 + m + m^2 + \dots + m^n \leq 1 + n * m^n, \forall m \geq 0, \forall n \geq 0$ ;
- ▶ Cada caminho de  $M$  analisa no máximo  $1 + n$  configurações;
- ▶  $M'$  precisa analisar, sozinha,  $1 + n * m^n$  configurações;
- ▶ O tempo de execução de  $M'$  é exponencial;
- ▶ Se  $M$  é  $O(n)$ , então  $M'$  é  $O(m^n)$ ;
- ▶ Se  $M$  é  $O(t(n))$ , então  $M'$  é  $O(m^{t(n)})$ .

# Múltiplas fitas de entrada

## Conceito

Ao invés de uma única fita, a Máquina de Turing possui uma quantidade finita de fitas:

- ▶ A cadeia de entrada é posicionada na primeira fita;
- ▶ Cada fita possui uma cabeça de leitura/escrita independente das demais;
- ▶ As transições controlam as leituras, as escritas e as movimentações de todas as cabeças;
- ▶ A cabeça pode permanecer no lugar de origem, sem se deslocar.

# Múltiplas fitas de entrada

## Definição

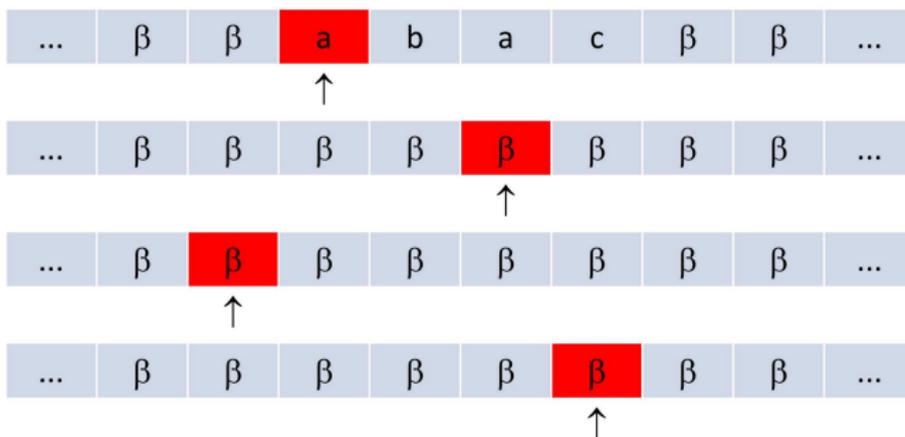
Máquina de Turing com  $n$  fitas:

$$\delta : Q \times \underbrace{\Gamma}_{\text{Fita 1}} \times \underbrace{\Gamma}_{\text{Fita 2}} \times \dots \times \underbrace{\Gamma}_{\text{Fita } n} \rightarrow$$

$$Q \times \underbrace{(\Gamma \times \{L, R, S\})}_{\text{Fita 1}} \times \underbrace{(\Gamma \times \{L, R, S\})}_{\text{Fita 2}} \dots \times \underbrace{(\Gamma \times \{L, R, S\})}_{\text{Fita } n}$$

# Múltiplas fitas de entrada

## Representação



# Múltiplas fitas de entrada

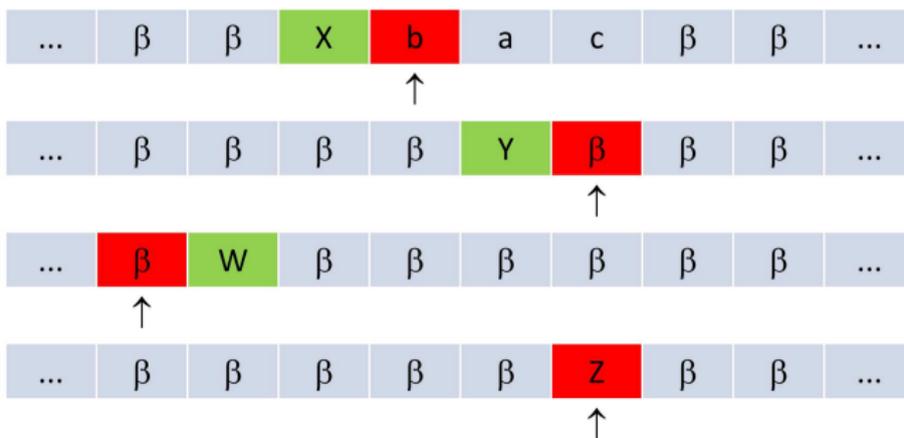
## Representação

Suponha:

$$\delta(q_0, a, \beta, \beta, \beta) = (q_1, (X, R), (Y, R), (W, L), (Z, S))$$

# Múltiplas fitas de entrada

## Representação



# Múltiplas fitas de entrada

## Equivalência

Como toda Máquina de Turing é uma Máquina de Turing com múltiplas fitas, é fato que Máquinas de Turing com múltiplas fitas aceitam todas as linguagens recursivamente enumeráveis. No entanto, cabe questionar se existem linguagens que não são recursivamente enumeráveis e que são aceitas por alguma Máquina de Turing com duas ou mais fitas.

# Múltiplas fitas de entrada

## Equivalência

Teorema: A classe das linguagens aceitas por Máquinas de Turing com múltiplas fitas corresponde exatamente à classe das linguagens aceitas por Máquinas de Turing com uma única fita.

- ▶ MT com uma única fita simula MT com múltiplas fitas, independentemente da quantidade de fitas.

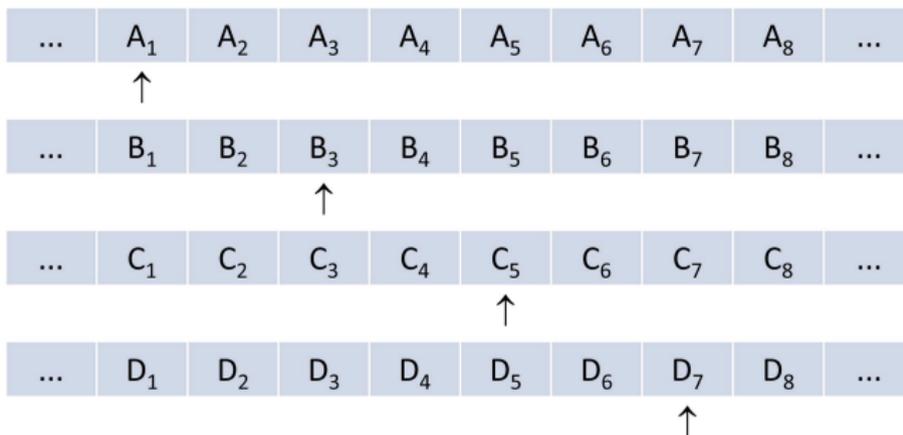
# Múltiplas fitas de entrada

## Convenção

- ▶ Uma MT com  $n$  fitas será representada por uma MT com uma única fita e  $2 * n$  trilhas;
- ▶ A trilha  $2 * i - 1$  representa o conteúdo da fita  $i$ ,  $1 \leq i \leq n$ ;
- ▶ A trilha  $2 * i$  representa (símbolo  $X$ ) a posição corrente da cabeça de leitura/escrita na fita  $2 * i - 1$ ,  $1 \leq i \leq n$ ;
- ▶ Exemplo para  $n = 4$ .

# Múltiplas fitas de entrada

## Exemplo



# Múltiplas fitas de entrada

## Exemplo

...	A <sub>1</sub>	A <sub>2</sub>	A <sub>3</sub>	A <sub>4</sub>	A <sub>5</sub>	A <sub>6</sub>	A <sub>7</sub>	A <sub>8</sub>	...
	X								
...	B <sub>1</sub>	B <sub>2</sub>	B <sub>3</sub>	B <sub>4</sub>	B <sub>5</sub>	B <sub>6</sub>	B <sub>7</sub>	B <sub>8</sub>	...
			X						
...	C <sub>1</sub>	C <sub>2</sub>	C <sub>3</sub>	C <sub>4</sub>	C <sub>5</sub>	C <sub>6</sub>	C <sub>7</sub>	C <sub>8</sub>	...
					X				
...	D <sub>1</sub>	D <sub>2</sub>	D <sub>3</sub>	D <sub>4</sub>	D <sub>5</sub>	D <sub>6</sub>	D <sub>7</sub>	D <sub>8</sub>	...
							X		

↑

# Múltiplas fitas de entrada

## Equivalência

Método:  $M_2$  simula  $M_1$

1.  $M_2$  precisa localizar as posições onde estão os marcadores das  $n$  cabeças de leitura/escrita de  $M_1$  na sua fita de entrada;
2. Para não se perder,  $M_2$  deve manter sempre, armazenado no seu conjunto de estados, a quantidade de marcadores que estão à esquerda e à direita da posição corrente de leitura/escrita;
3. Após a localização de cada marcador,  $M_2$  deve armazenar, no seu conjunto de estados, o símbolo lido em cada uma das posições correspondentes;
4. O estado de  $M_1$  deve estar armazenado também no conjunto de estados de  $M_2$ ;

# Múltiplas fitas de entrada

## Equivalência

Método:  $M_2$  simula  $M_1$

5.  $M_2$  determina a transição a ser aplicada;
6.  $M_2$  revisita cada um dos marcadores, desloca os mesmos de posição (se for o caso), substitui os símbolos correspondentes e registra uma eventual mudança de estado de  $M_1$  no seu próprio conjunto de estados;
7. Os estados de aceitação de  $M_2$  são aqueles que representam estados de aceitação de  $M_1$ .

# Múltiplas fitas de entrada

## Equivalência

Teorema: O tempo que  $M_2$  (com uma única fita) leva para simular  $n$  movimentos de  $M_1$  (com múltiplas fitas) é  $O(n^2)$ .

# Múltiplas fitas de entrada

## Equivalência

- ▶ Após 1 movimento de  $M_1$ , os marcadores de  $M_2$  estarão separados por no máximo 2 posições;
- ▶ Após 2 movimentos de  $M_1$ , os marcadores de  $M_2$  estarão separados por no máximo 4 posições;
- ▶ Após  $n$  movimentos de  $M_1$ , os marcadores de  $M_2$  estarão separados por no máximo  $2 * n$  posições;
- ▶ Para localizar todos os marcadores,  $M_2$  deve executar, no máximo,  $2 * i$  movimentos para a direita, onde  $i$  é o número de movimentos executados por  $M_1$  até o momento;
- ▶ Uma vez determinada a transição a ser aplicada,  $M_2$  deve substituir os símbolos nas trilhas ímpares na fita de entrada; para isso são requeridos, no máximo,  $2 * i$  movimentos para a esquerda;

# Múltiplas fitas de entrada

## Equivalência

- ▶ Também é necessário deslocar os marcadores das trilhas pares para a esquerda ou para a direita; para isso serão necessários outros 2 movimentos por marcador (um em cada sentido), num total de  $2 * k$  movimentos, onde  $k$  é o número de fitas sendo simuladas. Esse cálculo independe do valor de  $i$ .

Movimentos de $M_1$	Distância máxima em $M_2$	Movimentos de $M_2$
1	2	$4 + 2 * k$
2	4	$8 + 2 * k$
...	...	...
$n$	$2 * n$	$4 * n + 2 * k$

# Múltiplas fitas de entrada

## Equivalência

- ▶ Portanto, para simular  $n$  movimento de  $M_1$  são requeridos
 
$$\sum_{i=1}^n (4 * i + 2 * k) \leq n * (4 * n + 2 * k) = 4 * n^2 + 2 * k * n$$
 movimentos;
- ▶ Logo, para simular  $n$  movimentos de  $M_1$  serão requeridos, no máximo,  $O(n^2)$  movimentos.

# Fita limitada à esquerda

## Conceito

Ao invés de uma fita com tamanho ilimitado em ambos os sentidos, a Máquina de Turing possui uma fita limitada à esquerda e sem limitação à direita:

- ▶ A fita possui duas trilhas;
- ▶ A cadeia de entrada é posicionada na primeira trilha no início da fita;
- ▶ A fita é preenchida com brancos à direita do último símbolo da cadeia de entrada na primeira trilha, e integralmente na segunda trilha;
- ▶ Qualquer tentativa de movimentação da cabeça de leitura/escrita para a esquerda da primeira posição da fita gera uma condição de parada com rejeição da cadeia de entrada.

# Fita limitada à esquerda

## Equivalência

Teorema: A classe das linguagens aceitas por Máquinas de Turing com fita ilimitada em ambos os sentidos corresponde exatamente à classe das linguagens aceitas por Máquinas de Turing com fita limitada à esquerda.

- ▶ MT  $M_1$  com fita limitada à esquerda simula MT  $M_2$  com fita ilimitada;
- ▶  $M_1$  nunca escreve branco ( $B$ ) na fita de entrada;
- ▶ A cabeça de leitura/escrita nunca se desloca para a esquerda da primeira posição.



# Fita limitada à esquerda

Nunca escreve branco

## Algoritmo:

Seja  $M_2 = (Q_2, \Sigma, \Gamma_2, \delta_2, q_2, B, F_2)$ . Então:

- ▶ Fazer  $\Gamma_2 \leftarrow \Gamma_2 \cup \{B'\}$ ;
- ▶ Substituir toda regra do tipo  $\delta_2(q, X) = (p, B, D)$ ,  $D \in \{L, R\}$ , por  $\delta_2(q, X) = (p, B', D)$ ;
- ▶  $\forall q \in Q_2$ , fazer  $\delta_2(q, B') = \delta_2(q, B)$ .

# Fita limitada à esquerda

## Equivalência

### Método:

- ▶ Usar a trilha superior para representar o lado direito da fita, e a trilha inferior para representar o lado esquerdo da fita;
- ▶ Memorizar, nos estados de  $M_1$ , se a cabeça de leitura/escrita está posicionada à esquerda ou à direita da posição inicial em  $M_2$ ;
- ▶ Conforme o estado de  $M_1$ , manipular apenas a trilha superior ou inferior da fita de entrada;
- ▶ Garantir que toda movimentação para a direita da primeira posição seleciona a trilha superior, e que toda movimentação à esquerda da primeira posição seleciona a trilha inferior.

# Fita limitada à esquerda

## Equivalência

Considere  $M_2 = (Q_2, \Sigma, \Gamma_2, \delta_2, q_2, B, F_2)$  modificado para nunca escrever brancos na fita.  $M_2$  é simulado por  $M_1$  com fita limitada à esquerda,  $M_1 = (Q_1, \Sigma_1, \Gamma_1, \delta_1, q_0, (B, B), F_1)$ , onde:

- ▶  $Q_1 = \{q_0, q_1\} \cup (Q_2 \times \{U, L\})$ ;  
 $U$  indica a manipulação da trilha superior da fita,  $L$  indica a manipulação da trilha inferior;
- ▶  $\Sigma_1 = \Sigma \times \{B\}$ ;
- ▶  $\Gamma_1 = (\Gamma_2 \times \Gamma_2) \cup \{(X, *) | X \in \Gamma_2\}$ ;  
 $* \notin \Gamma_2$  é usado para indicar o início da fita.
- ▶  $(B, B)$  representa o branco de  $M_2$ ;
- ▶  $F_1 = \{(q, T) \in (F_2 \times \{U, L\}) | q \in F_2\}$ .

# Fita limitada à esquerda

## Equivalência

Obtenção de  $\delta_1$ :

1.  $\delta_1(q_0, (\sigma, B)) = (q_1, (\sigma, *), R)$ ,  $\forall \sigma \in (\Sigma \cup \{B\})$ ;  
primeiro movimento: inserção do marcador de início de fita na segunda trilha;
2.  $\delta_1(q_1, (X, B)) = ((q_2, U), (X, B), L)$ ,  $\forall X \in \Gamma_2$ ;  
segundo movimento: retornar para a posição inicial da fita, selecionar trilha superior (parte direita da fita de  $M_1$ ) e ir para o estado inicial  $q_2$ ;
3. Se  $\delta_2(q, X) = (p, Y, D)$ , então,  $\forall Z \in \Gamma_2$ :
  - ▶  $\delta_1((q, U), (X, Z)) = ((p, U), (Y, Z), D)$  e
  - ▶  $\delta_1((q, L), (Z, X)) = ((p, L), (Z, Y), \overline{D})$ .

simula  $M_2$  levando em consideração a trilha corrente, exceto se estiver na primeira posição.

# Fita limitada à esquerda

## Equivalência

Obtenção de  $\delta_1$ :

4. Se  $\delta_2(q, X) = (p, Y, R)$ , então:

$$\delta_1((q, L), (X, *)) = \delta_1((q, U), (X, *)) = ((p, U), (Y, *), R)$$

deslocamento à direita da primeira posição seleciona a trilha superior;

5. Se  $\delta_2(q, X) = (p, Y, L)$ , então:

$$\delta_1((q, L), (X, *)) = \delta_1((q, U), (X, *)) = ((p, L), (Y, *), R)$$

deslocamento à esquerda a primeira posição seleciona trilha inferior.

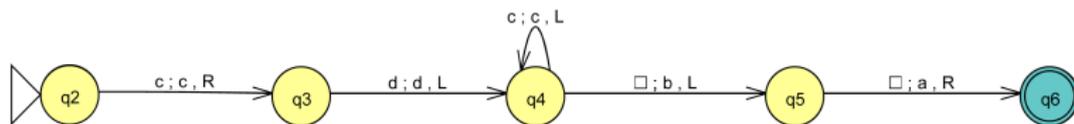
# Fita limitada à esquerda

## Conclusão

- ▶  $M_1$  reproduz as configurações de  $M_2$ ;
- ▶  $M_1$  entra em um estado de aceitação se e somente se  $M_2$  também entra;
- ▶  $L(M_1) = L(M_2)$ .

# Fita limitada à esquerda

## Exemplo



# Fita limitada à esquerda

## Exemplo

- ▶  $Q_1 = \{q_0, q_1, (q_2, U), (q_2, L), (q_3, U), (q_3, L), (q_4, U), (q_4, L), (q_5, U), (q_5, L), (q_6, U), (q_6, L)\}$
- ▶  $\Sigma_1 = \{(a, B), (b, B), (c, B), (d, B)\}$
- ▶  $\Gamma_1 = \{(a, *), (b, *), (c, *), (d, *), (a, a), (a, b), (a, c), (a, d), (a, B), (b, a), (b, b), (b, c), (b, d), (b, B), (c, a), (c, b), (c, c), (c, d), (c, B), (d, a), (d, b), (d, c), (d, d), (d, B), (B, a), (B, b), (B, c), (B, d), (B, B)\}$
- ▶  $F_1 = \{(q_6, U), (q_6, L)\}$

## Fita limitada à esquerda

## Exemplo

Obtenção de  $\delta_1$ :

1.  $\delta_1(q_0, (a, B)) = (q_1, (a, *), R)$   
 $\delta_1(q_0, (b, B)) = (q_1, (b, *), R)$   
 $\delta_1(q_0, (c, B)) = (q_1, (c, *), R)$   
 $\delta_1(q_0, (d, B)) = (q_1, (d, *), R)$   
 $\delta_1(q_0, (B, B)) = (q_1, (B, *), R)$
2.  $\delta_1(q_1, (a, B)) = ((q_2, U), (a, B), L)$   
 $\delta_1(q_1, (b, B)) = ((q_2, U), (b, B), L)$   
 $\delta_1(q_1, (c, B)) = ((q_2, U), (c, B), L)$   
 $\delta_1(q_1, (d, B)) = ((q_2, U), (d, B), L)$   
 $\delta_1(q_1, (B, B)) = ((q_2, U), (B, B), L)$

## Fita limitada à esquerda

## Exemplo

A partir de  $\delta_2(q_2, c) = (q_3, c, R)$ :

3.  $\delta_1((q_2, U), (c, a)) = ((q_3, U), (c, a), R)$
- $\delta_1((q_2, U), (c, b)) = ((q_3, U), (c, b), R)$
- $\delta_1((q_2, U), (c, c)) = ((q_3, U), (c, c), R)$
- $\delta_1((q_2, U), (c, d)) = ((q_3, U), (c, d), R)$
- $\delta_1((q_2, U), (c, B)) = ((q_3, U), (c, B), R)$
- $\delta_1((q_2, L), (a, c)) = ((q_3, L), (a, c), L)$
- $\delta_1((q_2, L), (b, c)) = ((q_3, L), (b, c), L)$
- $\delta_1((q_2, L), (c, c)) = ((q_3, L), (c, c), L)$
- $\delta_1((q_2, L), (d, c)) = ((q_3, L), (d, c), L)$
- $\delta_1((q_2, L), (B, c)) = ((q_3, L), (B, c), L)$

# Fita limitada à esquerda

## Exemplo

A partir de  $\delta_2(q_3, d) = (q_4, d, L)$ :

3.  $\delta_1((q_3, U), (d, a)) = ((q_4, U), (d, a), L)$
- $\delta_1((q_3, U), (d, b)) = ((q_4, U), (d, b), L)$
- $\delta_1((q_3, U), (d, c)) = ((q_4, U), (d, c), L)$
- $\delta_1((q_3, U), (d, d)) = ((q_4, U), (d, d), L)$
- $\delta_1((q_3, U), (d, B)) = ((q_4, U), (d, B), L)$
- $\delta_1((q_3, L), (a, d)) = ((q_4, L), (a, d), R)$
- $\delta_1((q_3, L), (b, d)) = ((q_4, L), (b, d), R)$
- $\delta_1((q_3, L), (c, d)) = ((q_4, L), (c, d), R)$
- $\delta_1((q_3, L), (d, d)) = ((q_4, L), (d, d), R)$
- $\delta_1((q_3, L), (B, d)) = ((q_4, L), (B, d), R)$

## Fita limitada à esquerda

## Exemplo

A partir de  $\delta_2(q_4, c) = (q_4, c, L)$ :

3.  $\delta_1((q_4, U), (c, a)) = ((q_4, U), (c, a), L)$
- $\delta_1((q_4, U), (c, b)) = ((q_4, U), (c, b), L)$
- $\delta_1((q_4, U), (c, c)) = ((q_4, U), (c, c), L)$
- $\delta_1((q_4, U), (c, d)) = ((q_4, U), (c, d), L)$
- $\delta_1((q_4, U), (c, B)) = ((q_4, U), (c, B), L)$
- $\delta_1((q_4, L), (a, c)) = ((q_4, L), (a, c), R)$
- $\delta_1((q_4, L), (b, c)) = ((q_4, L), (b, c), R)$
- $\delta_1((q_4, L), (c, c)) = ((q_4, L), (c, c), R)$
- $\delta_1((q_4, L), (d, c)) = ((q_4, L), (d, c), R)$
- $\delta_1((q_4, L), (B, c)) = ((q_4, L), (B, c), R)$

# Fita limitada à esquerda

## Exemplo

A partir de  $\delta_2(q_4, B) = (q_5, b, L)$ :

3.  $\delta_1((q_4, U), (B, a)) = ((q_5, U), (b, a), L)$
- $\delta_1((q_4, U), (B, b)) = ((q_5, U), (b, b), L)$
- $\delta_1((q_4, U), (B, c)) = ((q_5, U), (b, c), L)$
- $\delta_1((q_4, U), (B, d)) = ((q_5, U), (b, d), L)$
- $\delta_1((q_4, U), (B, B)) = ((q_5, U), (b, B), L)$
- $\delta_1((q_4, L), (a, B)) = ((q_5, L), (a, b), R)$
- $\delta_1((q_4, L), (b, B)) = ((q_5, L), (b, b), R)$
- $\delta_1((q_4, L), (c, B)) = ((q_5, L), (c, b), R)$
- $\delta_1((q_4, L), (d, B)) = ((q_5, L), (d, b), R)$
- $\delta_1((q_4, L), (B, B)) = ((q_5, L), (B, b), R)$

## Fita limitada à esquerda

## Exemplo

A partir de  $\delta_2(q_4, B) = (q_5, a, R)$ :

3.  $\delta_1((q_5, U), (B, a)) = ((q_6, U), (a, a), R)$
- $\delta_1((q_5, U), (B, b)) = ((q_6, U), (a, b), R)$
- $\delta_1((q_5, U), (B, c)) = ((q_6, U), (a, c), R)$
- $\delta_1((q_5, U), (B, d)) = ((q_6, U), (a, d), R)$
- $\delta_1((q_5, U), (B, B)) = ((q_6, U), (a, B), R)$
- $\delta_1((q_5, L), (a, B)) = ((q_6, L), (a, a), L)$
- $\delta_1((q_5, L), (b, B)) = ((q_6, L), (b, a), L)$
- $\delta_1((q_5, L), (c, B)) = ((q_6, L), (c, a), L)$
- $\delta_1((q_5, L), (d, B)) = ((q_6, L), (d, a), L)$
- $\delta_1((q_5, L), (B, B)) = ((q_6, L), (B, a), L)$

# Fita limitada à esquerda

## Exemplo

A partir de  $\delta_2(q_2, c) = (q_3, c, R)$ :

$$4. \delta_1((q_2, L), (c, *)) = \delta_1((q_2, U), (c, *)) = ((q_3, U), (c, *), R)$$

A partir de  $\delta_2(q_5, B) = (q_6, a, R)$ :

$$4. \delta_1((q_5, L), (B, *)) = \delta_1((q_5, U), (B, *)) = ((q_6, U), (a, *), R)$$

# Fita limitada à esquerda

## Exemplo

A partir de  $\delta_2(q_3, d) = (q_4, d, L)$ :

$$5. \delta_1((q_3, L), (d, *)) = \delta_1((q_3, U), (d, *)) = ((q_4, L), (d, *), R)$$

A partir de  $\delta_2(q_4, c) = (q_4, c, L)$ :

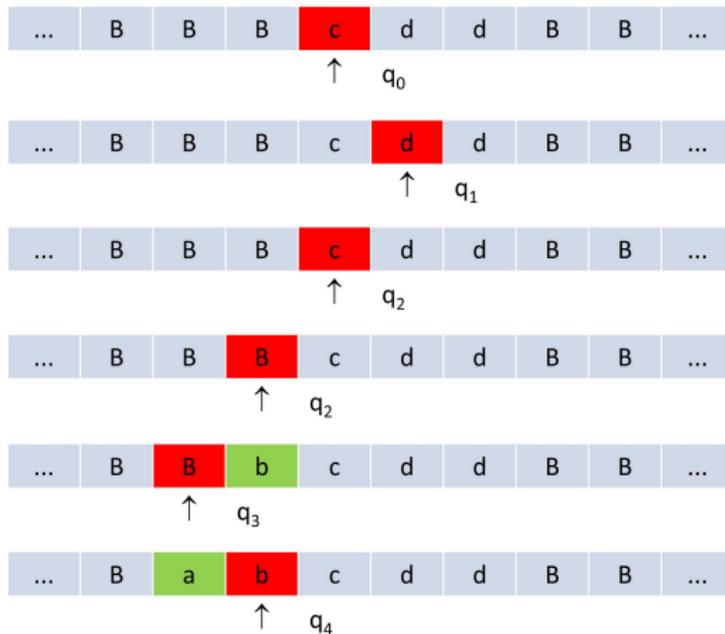
$$5. \delta_1((q_4, L), (c, *)) = \delta_1((q_4, U), (c, *)) = ((q_4, L), (c, *), R)$$

A partir de  $\delta_2(q_4, B) = (q_5, b, L)$ :

$$5. \delta_1((q_4, L), (B, *)) = \delta_1((q_4, U), (B, *)) = ((q_5, L), (b, *), R)$$

# Fita limitada à esquerda

## Exemplo



# Fita limitada à esquerda

## Exemplo

c	d	d	B	B	...
B	B	B	B	B	...

↑  $q_0$

c	d	d	B	B	...
*	B	B	B	B	...

↑  $q_1$

c	d	d	B	B	...
*	B	B	B	B	...

↑  $(q_2, U)$

c	d	d	B	B	...
*	B	B	B	B	...

↑  $(q_3, U)$

# Fita limitada à esquerda

## Exemplo

c	d	d	B	B	...
*	B	B	B	B	...

↑  $(q_4, U)$

c	d	d	B	B	...
*	B	B	B	B	...

↑  $(q_4, L)$

c	d	d	B	B	...
*	b	B	B	B	...

↑  $(q_5, L)$

c	d	d	B	B	...
*	b	a	B	B	...

↑  $(q_6, L)$