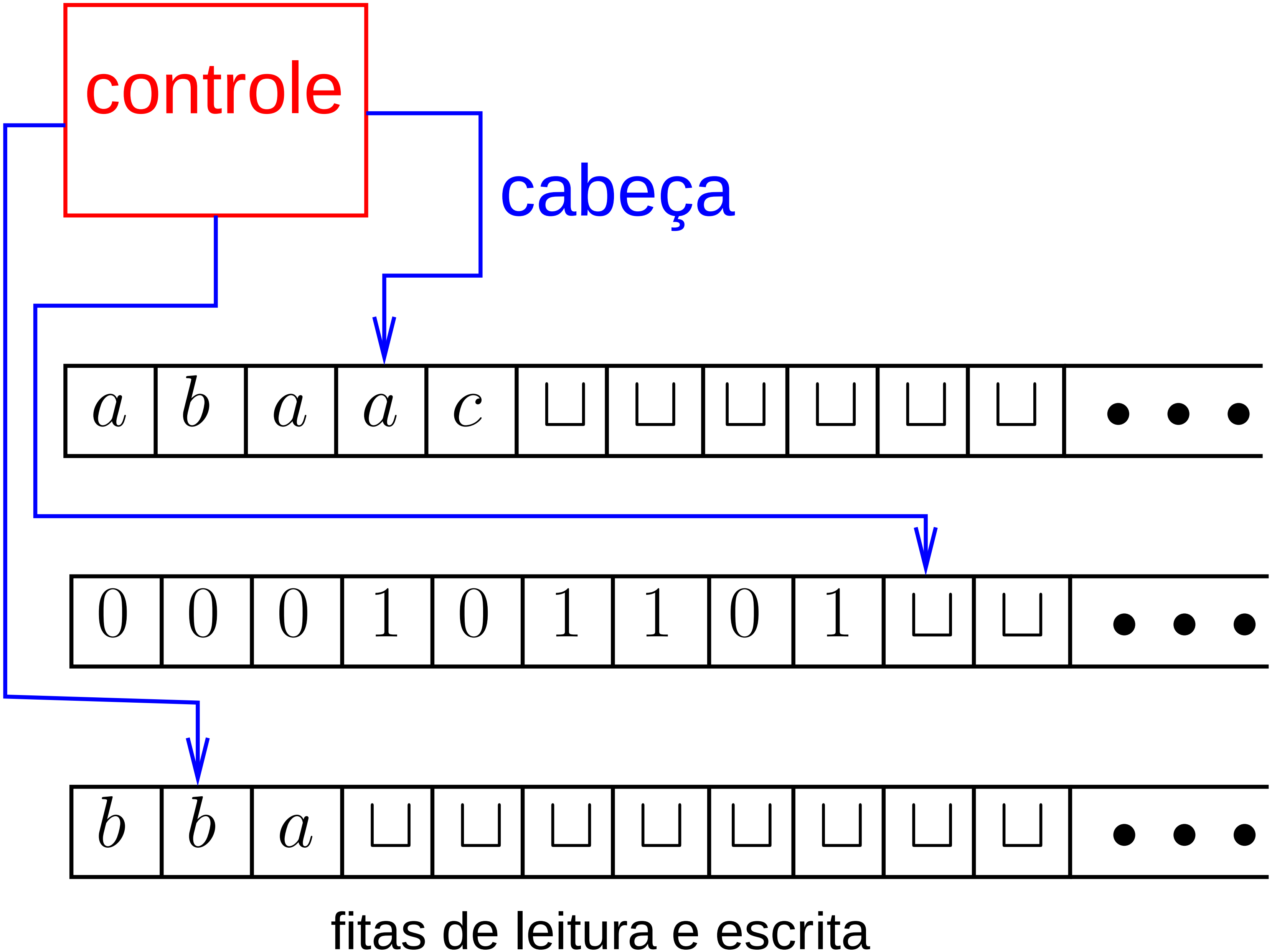
Melhores momentos

AULA PASSADA

Máquinas de Turing multifita



Exemplo de MT multifita

Descrição alto nível de uma máquina de Turing que decide se uma dada cadeia w está na linguagem

$$\{z : z \in \{0,1\}^*,\ z = z^R\}.$$

$M_1 =$ “Com entrada w:

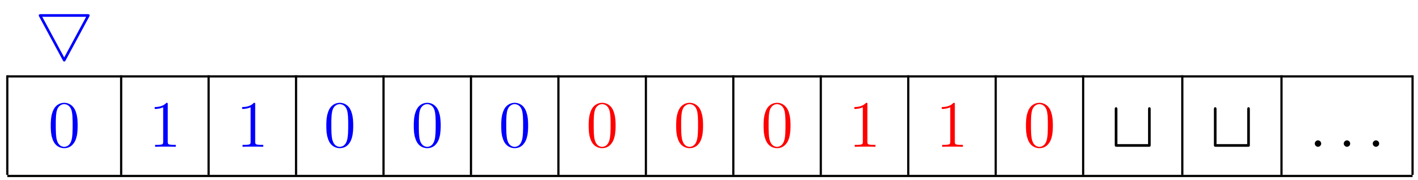
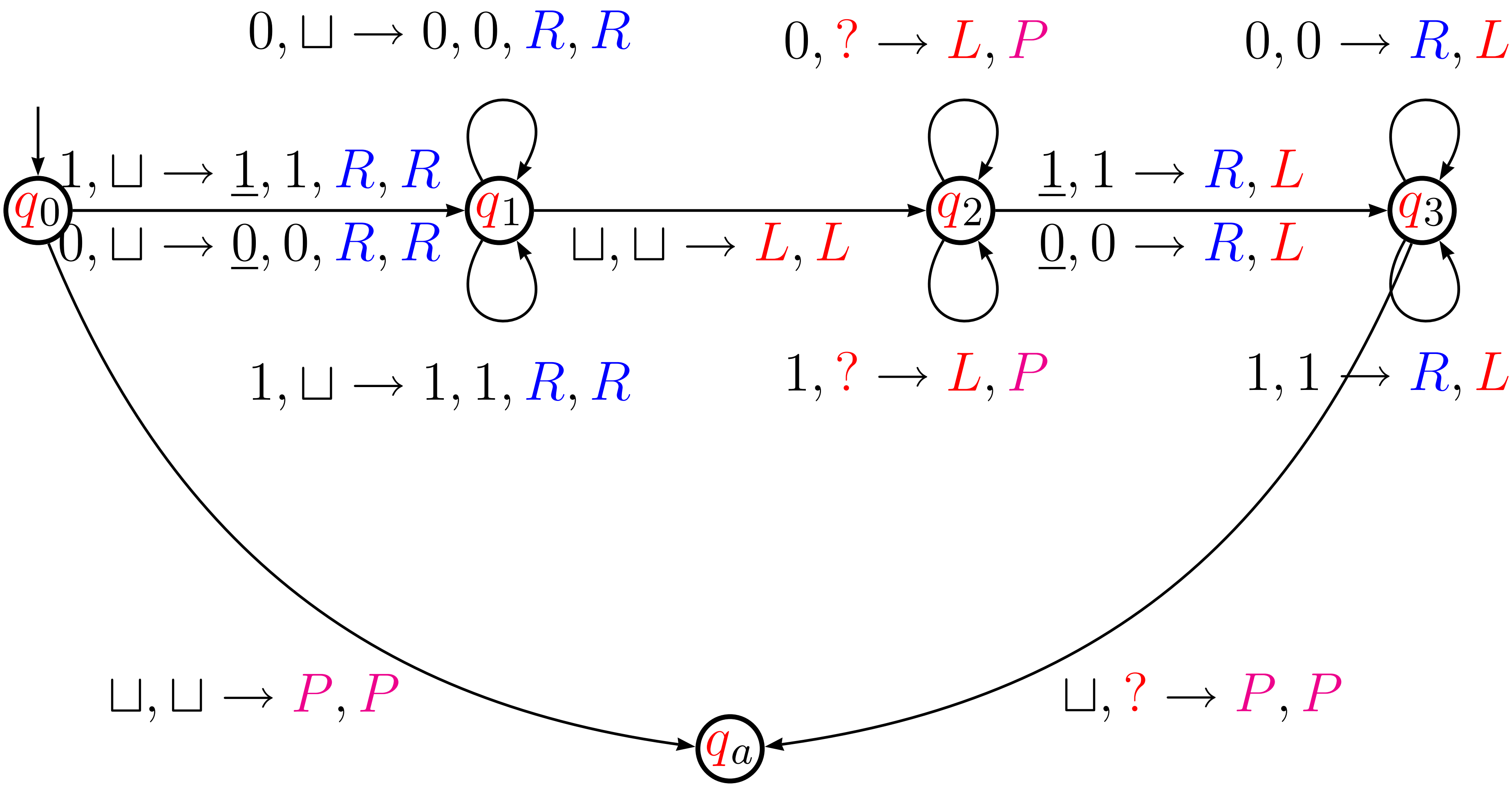


Diagrama de estados para M_5



Transições ausentes levam para $q_{\text{rejeição}}$

Número de passos

Se a cadeia de entrada tem comprimento n então a máquina M_5 faz não mais do que $3n + 1$ passos.

MT multifita por MT fita única

Duas máquinas são **equivalentes** se elas reconhecem a mesma linguagem.

Teorema. Dada uma máquina de Turing multifita M, podemos construir uma uma máquina de Turing S com uma única fita e equivalente a M. Além disso, se tendo como entrada uma cadeia de comprimento n a máquina M faz não mais do que $t(n) \geq n$ passos, então S faz $\mathrm{O}(t^2(n))$ passos.

AULA 5

Máquinas de Turing não-determinísticas

MS 3.2, MS 7.1

Máquinas de Turing não-determinísticas

Para cada estado q e símbolo a, $\delta(q,a)$ é um **conjunto finito de ternos**:

$$(q_1, a_1, L), (q_2, a_2, R), \ldots, (q_b, a_b, R)$$

A função de transição é da forma

$$\delta : (Q \setminus \{q_{\text{aceitação}}, q_{\text{rejeição}}\}) \times \Gamma \to \mathcal{P}(Q \times \Gamma \times \{L, P, R\}).$$

A computação da MT se ramifica dependendo das possibilidades da função δ.

Se algum ramo atinge o estado de aceitação, a máquina aceita a entrada.

Exemplo de MT não-determinística

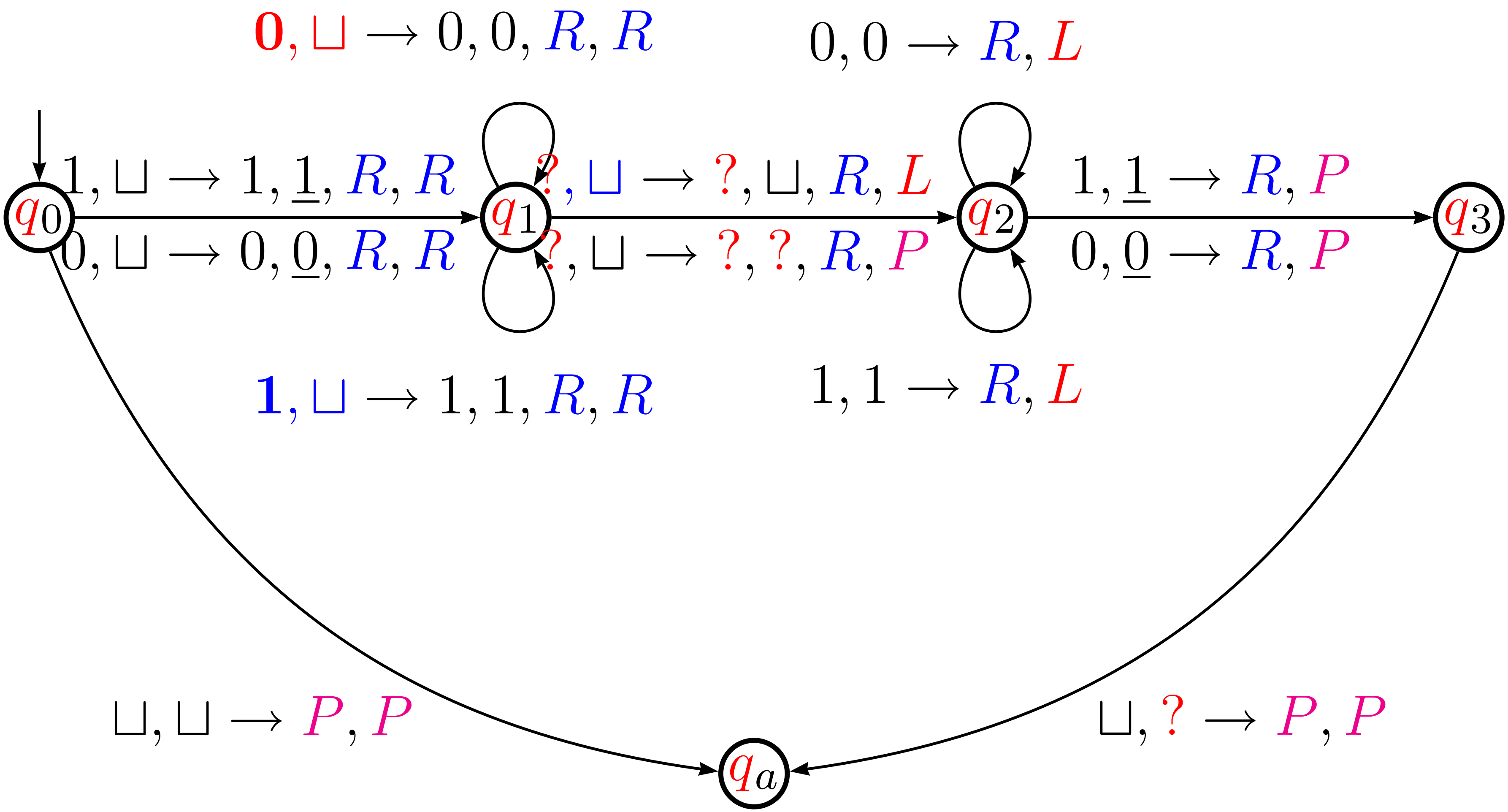
Máquina de Turing não-determinística com duas fitas que decide se uma dada cadeia w está na linguagem

$$\{z : z \in \{0,1\}^*, z = z^R\}.$$

$N_0 =$ "Com entrada w:

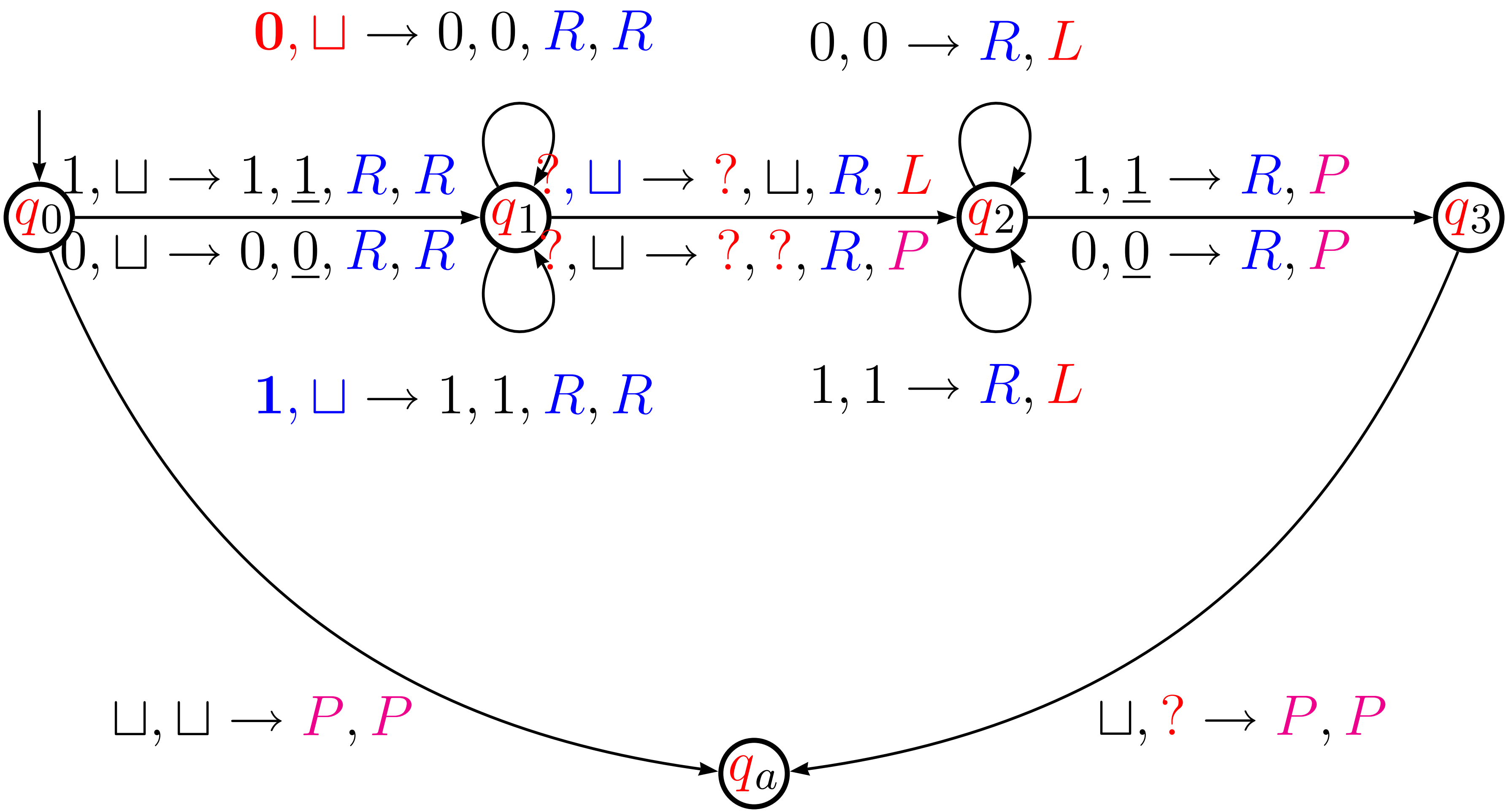
$\bigtriangledown$														
0	1	1	0	0	0	0	0	0	1	1	0	$\sqcup$	$\sqcup$	...

Diagrama de estados para N_0



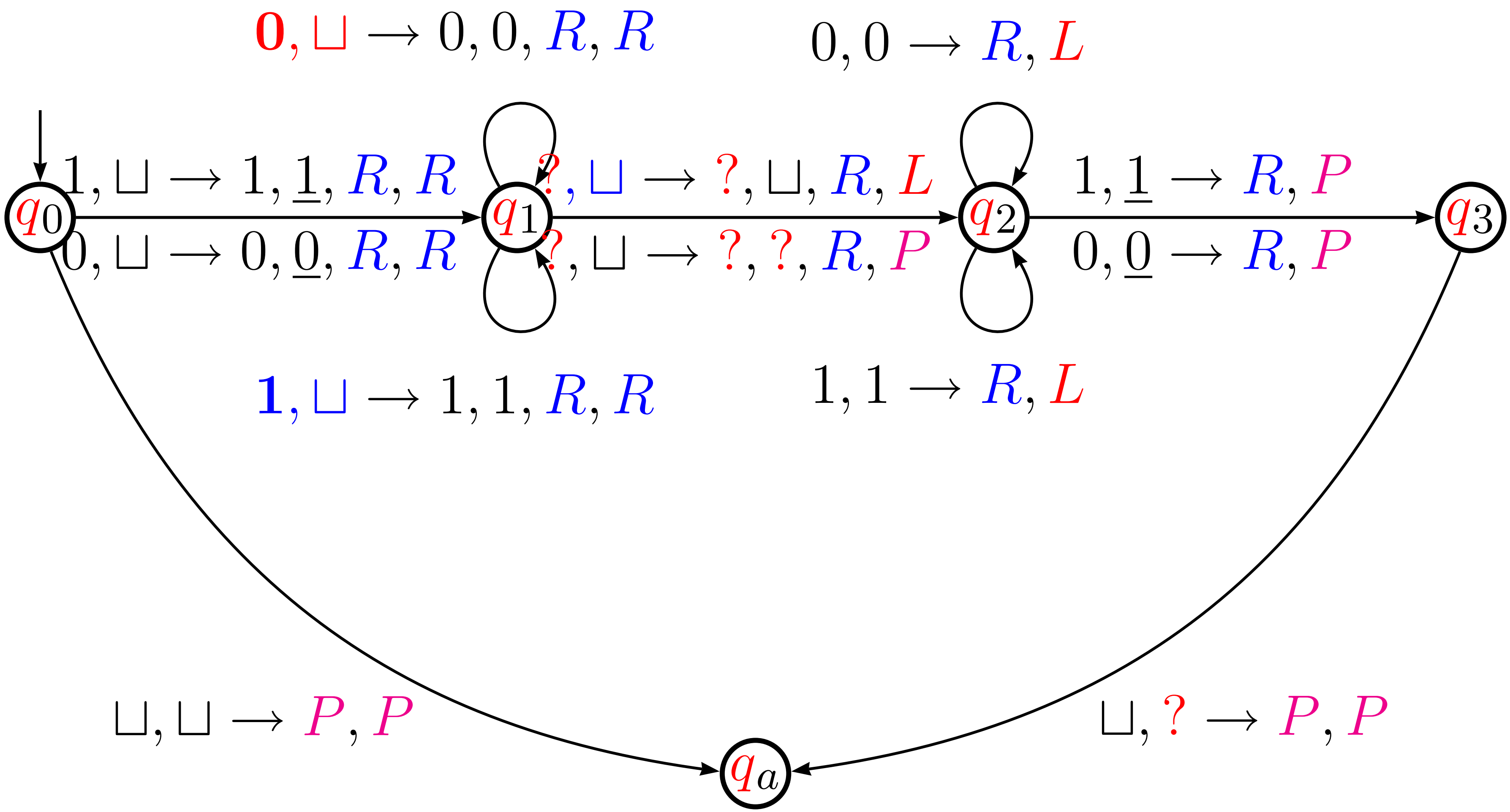
Corrigi o diagrama para aceitar palíndromos ímpares. Isto foi feito fazendo com que a MT, não-deterministicamente, ignorasse o símbolo do meio de cadeias de comprimento ímpar.

Diagrama de estados para N_0



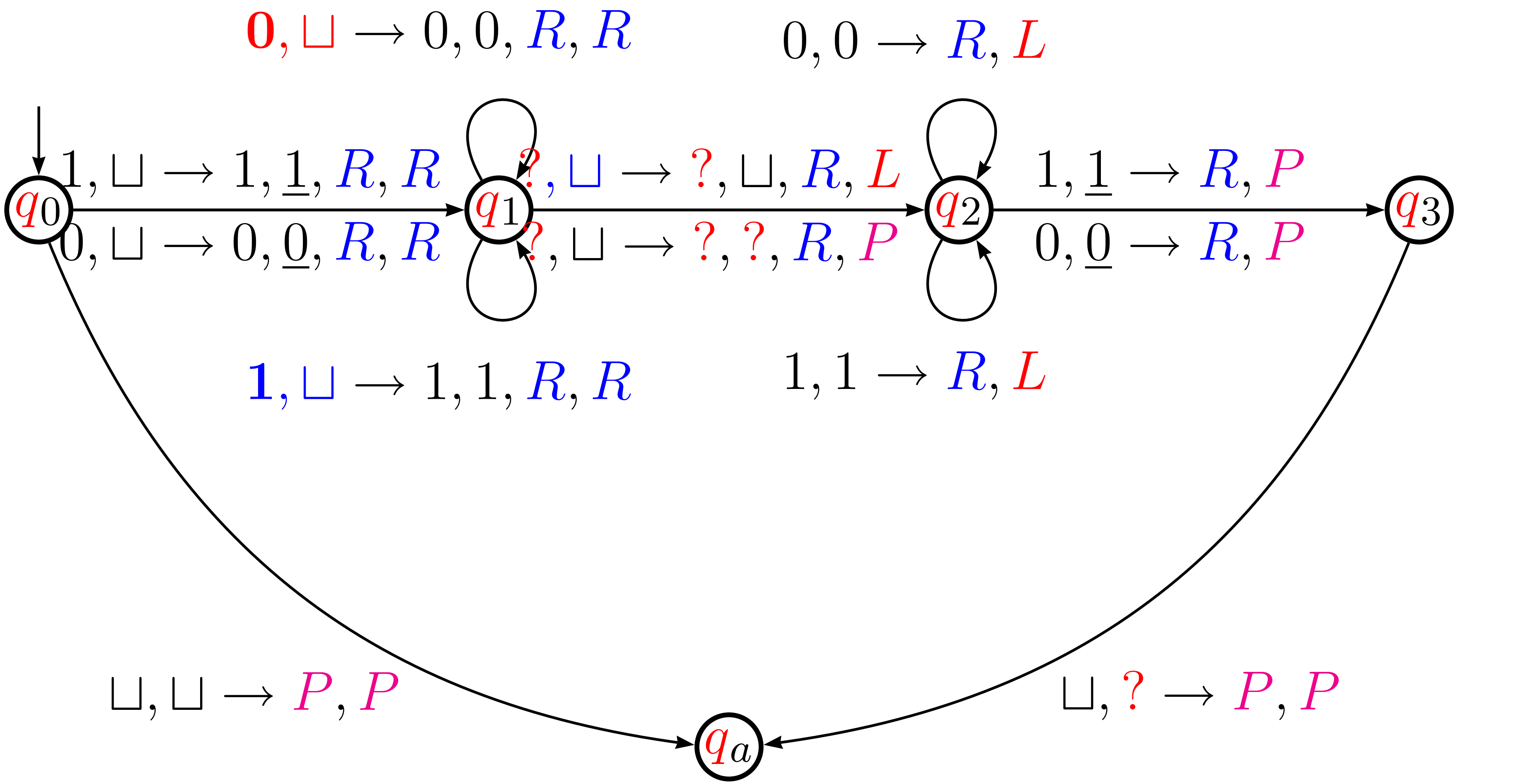
Em uma mesma transição, $? = 0$ ou $? = 1$, **não ambos**.
Transições ausentes levam para $q_{\text{rejeição}}$.

Diagrama de estados para N_0



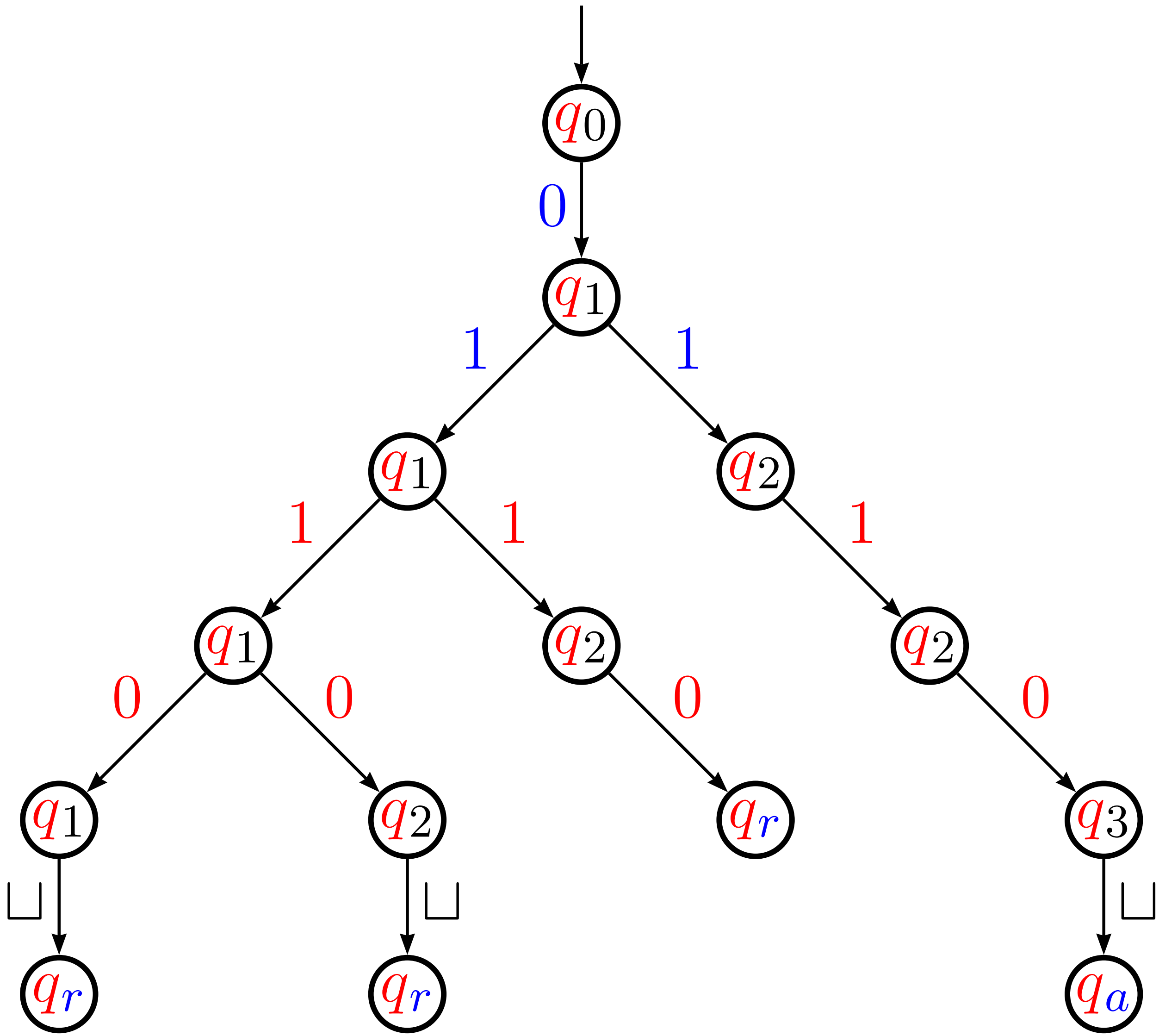
$$\delta(q_1, 0, \sqcup) = \{(q_1, 0, 0, R, R), (q_2, 0, 0, R, P), (q_2, 0, \sqcup, R, L)\}$$
$$\delta(q_1, 1, \sqcup) = \{(q_1, 1, 1, R, R), (q_2, 1, 1, R, P), (q_2, 1, \sqcup, R, L)\}$$

Diagrama de estados para N_0



Árvore da computação de N_0

$w = 0\,1\,1\,0$. O rótulo no arco é o símbolo sob a cabeça da fita 1.



Não-determinístico por determinísmo

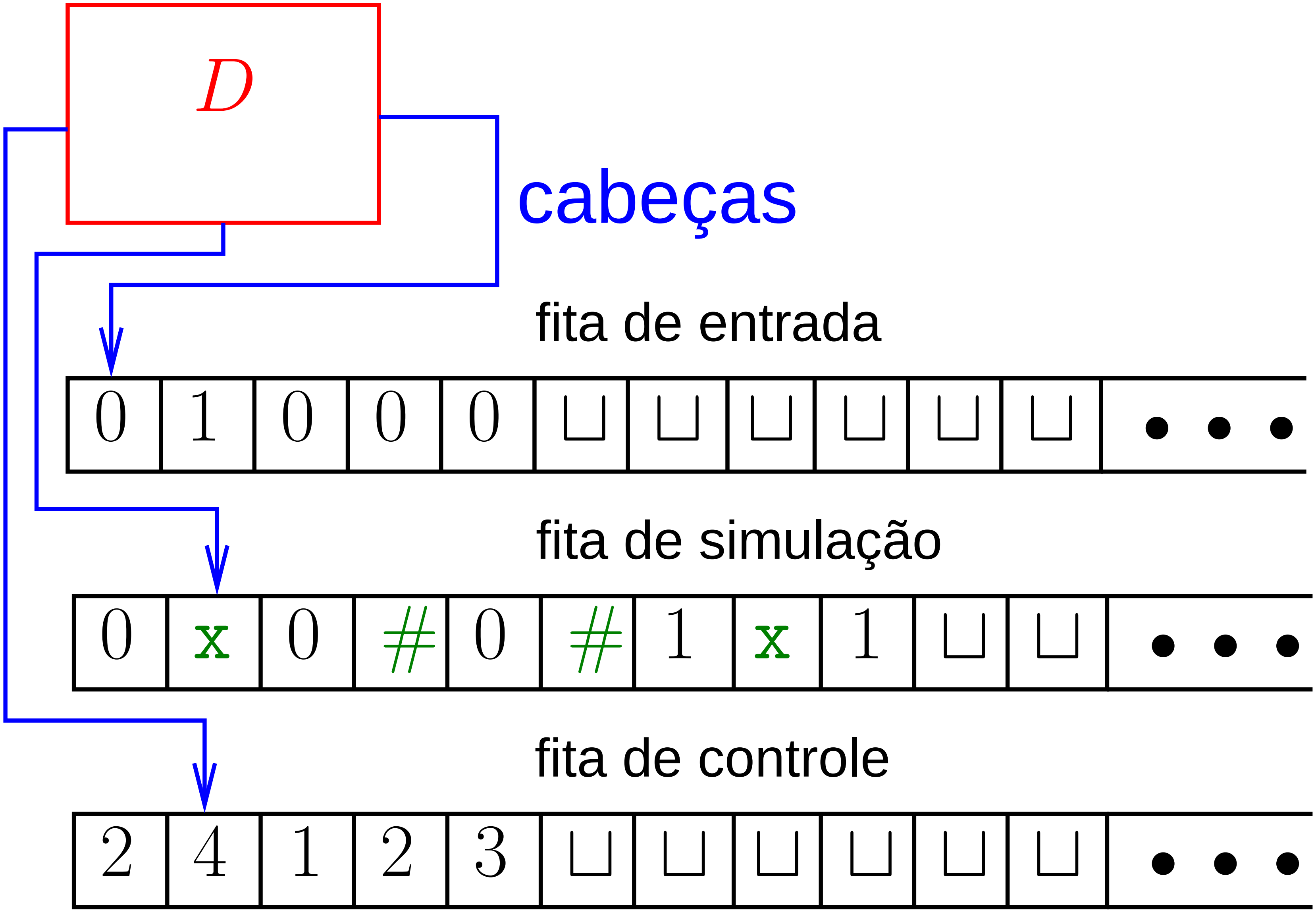
Teorema. Dada uma máquina de Turing não-determinística N, podemos construir uma máquina de Turing determinística D equivalente a N.

Idéia da prova.

D tenta todas as possíveis computações de N buscando atingir o estado de aceitação.

Não pode fazer *busca em profundidade*, mas fará implicitamente *busca em largura* na árvore da computação.

Não-determinismo por determinismo



Prova: A máquina de Turing D que simula N tem 3 fitas.

Sabemos que D é equivalente a uma máquina de Turing comum.

- A fita 1 contém a entrada e nunca é alterada.
- A fita 2 mantém uma cópia da fita de N para alguma de suas possíveis computações.
- A fita 3 controla qual a busca em largura das possíveis computações de N.

Seja b o tamanho do maior conjunto de escolhas da função δ de N. No exemplo de MT não-determinística com 2-fitas $b = 3$.

A cada nó na árvore de busca podemos associar um endereço que é uma cadeia sobre o alfabeto $\Sigma_b = \{1, 2, \ldots, b\}$.

Por exemplo, o endereço $2\,3\,1$ corresponde ao nó que se atinge começando da raiz, seguindo o segundo filho, depois o terceiro filho e depois o primeiro filho.

Se um nó não tem o b-ésimo filho, então o endereço correspondente não corresponde a nenhum nó. A fita 3 contém uma cadeia sobre Σ_b representando um ramo da computação.

Simulação de N por D

$D =$ "Com entrada w:

1. Copie fita 1 para fita 2.
2. Use fita 2 para simular N com entrada w em um ramo de sua computação não-determinística. Antes de cada passo de N consulte o próximo símbolo da fita 3. Se não existirem mais símbolos na fita 3, ou se a escolha indicada pela fita 2 não for válida, aborte esse ramo e vá para o passo 3. Se o estado de rejeição é atingido, também vá ao passo 3. Se o estado de aceitação é atingido, *aceite*.
3. Troque o conteúdo da fita 3 pela próxima cadeia em ordem lexicográfica. Volte ao passo 1."

■

Conclusão

Corolário. Uma linguagem é Turing-reconhecível se e somente se alguma máquina de Turing não-determinística a reconhece.

Prova:

($\Rightarrow$) Uma máquina de Turing comum é um caso particular de uma máquina de Turing não-determinística.

($\Leftarrow$) Segue do teorema anterior.

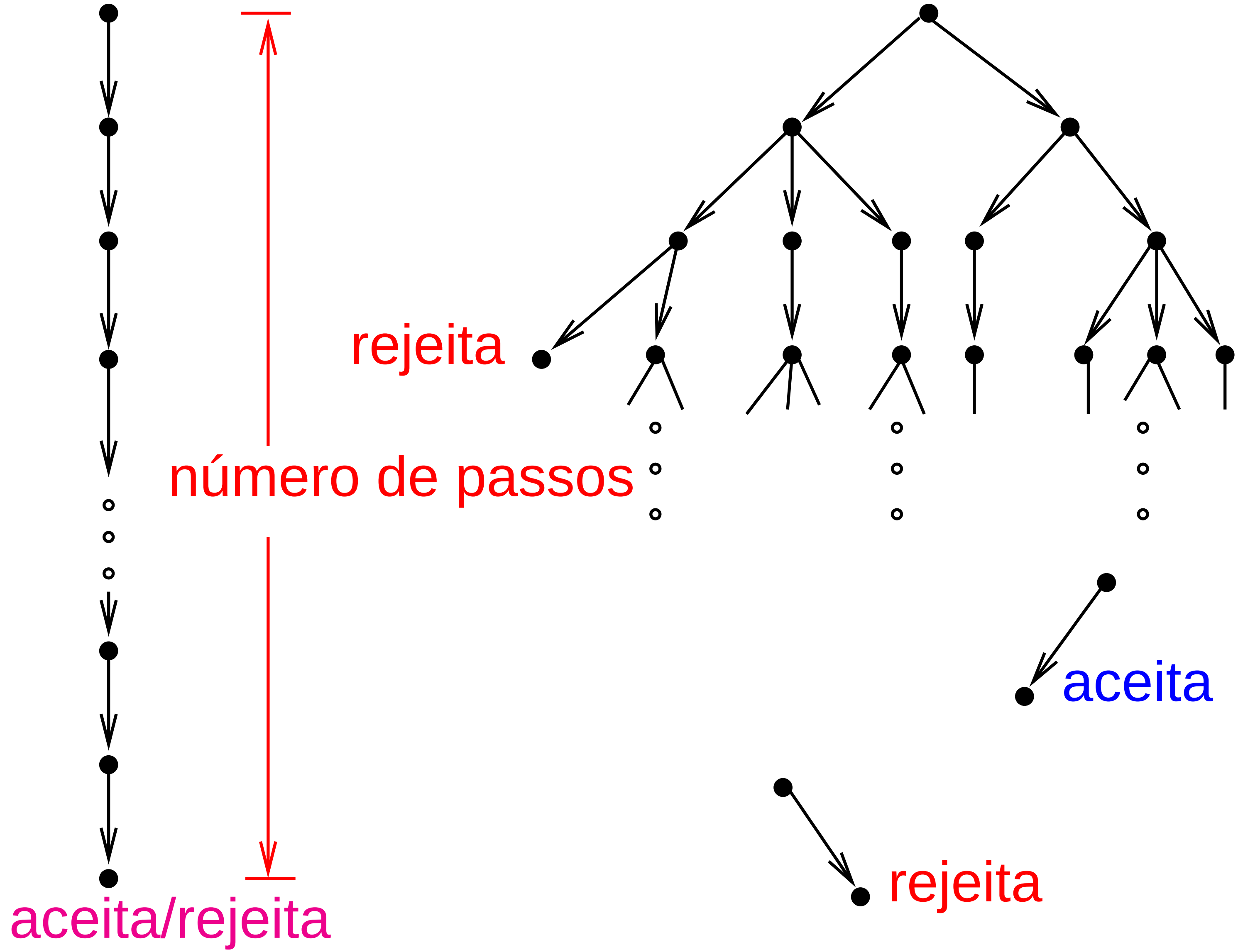
Chamamos uma máquina de Turing não-determinística de **decisora** se todos os possíveis ramos de computação param em todas as entradas.

Conclusão

Corolário. Uma linguagem é decidível se e somente se alguma máquina de Turing não-determinística a decide.

Prova: Ver exercício 3.3.

Número de passos



determinismo **não-determinismo**

Número de passos

O **número de passos** realizados por uma máquina de Turing não-determinística é o número máximo de passos de um dos ramos da computação.

Número de passos

O **número de passos** realizados por uma máquina de Turing não-determinística é o número máximo de passos de um dos ramos da computação.

O número de passos feitos pela máquina N_0 tendo como entrada uma cadeia com n símbolos é

$$????$$

Número de passos

O **número de passos** realizados por uma máquina de Turing não-determinística é o número máximo de passos de um dos ramos da computação.

O número de passos feitos pela máquina N_0 tendo como entrada uma cadeia com n é

$$n + 1\,.$$

Não-determinístico por determinísmo

Teorema. Dada uma máquina de Turing não-determinística decisora N, podemos construir uma máquina de Turing determinística D equivalente a N. Além disso, se tendo como entrada uma cadeia de comprimento n a máquina N faz não mais do que $t(n) \geq n$ passos, então D faz $2^{\mathrm{O}(t(n))}$ passos.

Como sugerido na aula (Hmmm, foi isto que sugeriram?!? Agora não sei e na hora não entendi. . .) no teorema poderíamos escrever $b^{\mathrm{O}(t(n))}$, mas fica mais bonito com 2 no lugar do b: no enunciado não precisamos falar de b e $b \geq 2$.

Prova: Já vimos como converter N para D.

Hmmm. A MT D aqui é ligeiramente diferente da MT do teorema anterior. A máquina D deve parar sempre (ser decisora). Para isto podemos verificar se, em cada nível da árvore da computação, existe algum nó para o qual a computação não terminou. Isto pode ser feito com "uma variável booleana" que assim que é 0 sempre que o comprimento da cadeia da fita de controle aumenta e recebe 1 quando alguma computação com esse comprimento não termina. Esta variável booleana tem o mesmo comportamento da variával `fila_vazia` da busca em largura.

$b := \max_{q,a}\{|\delta(q,a)|\}$ é constante de N

b mede um certo "grau de não-determinísmo",

$$b = 1 \Rightarrow \text{determinísmo.}$$

Contabilização do número de passos

Preparação da fita 2 para simulação: $\mathrm{O}(n)$ passos

Como sugerido na aula, poderíamos escrever $\mathrm{O}(t(n))$, já que $t(n) \geq n$. Isto simplificaria o passo de inicialização e não faria diferença nas contas a seguir.

Simulação para cada cadeia de controle: $\mathrm{O}(t(n))$ passos

Atualização da cadeia na fita de controle: $\mathrm{O}(t(n))$ passos

Atualização $=$ computar “sucessor b-ário”

Número de cadeias diferentes na fita de controle: $\mathrm{O}(b^{t(n)})$

Total:

$$\begin{aligned}
\mathrm{O}(b^{t(n)})(\mathrm{O}(n) + 2\mathrm{O}(t(n))) &= \mathrm{O}(t(n)b^{t(n)}) \quad (n \geq t(n)) \\
&= \mathrm{O}(2^{\lg(t(n)b^{t(n)})}) \\
&= \mathrm{O}(2^{\lg t(n) + t(n)\lg b}) \\
&= 2^{\mathrm{O}(t(n))}
\end{aligned}$$

MT com 3 fitas para MT com 1 fita

Para transformar a MT determinística com 3 fitas para uma máquina com apenas uma fita o números de passos é

$$
\begin{aligned}
(2^{\mathrm{O}(t(n))})^2 &= 2^{2\mathrm{O}(t(n))} \\
&= 2^{\mathrm{O}(t(n))}
\end{aligned}
$$

Observação. $t(n)$ não precisa ser computável.

Isto significa que a MT D não usa a função $t(n)$ (como subrotina). Basta que $t(n)$ exista para que a máquina D faça o serviço dela sem entrar em loop.

■