ACH2043
INTRODUÇÃO À TEORIA DA COMPUTAÇÃO

Aula 18

Cap 3.2 – Variantes de Máquinas de Turing

Profa. Ariane Machado Lima
ariane.machado@usp.br

Adiaremos nossa prova para 3/6? **Não!**

12		15/04/24	PROVA 1		
13	12	17/04/24	Forma Normal de Chomsky		
14	13	22/04/24	Autômato a Pilha		
15	14	24/04/24	Equivalência		
16	15	29/04/24	Linguagens não LC		
		01/05/24	FERIADO – DIA DO TRABALHO		
17	16	06/05/24	MT		
18	17	08/05/24	Linguagens sensíveis ao contexto		
19		13/05/24	WORKSHOP PPGSI		
20		15/05/24	Correção da P1		
21	18	20/05/24	Variantes de Máquina de Turing		
22	19	22/05/24	Variantes de MT		
23		27/05/24	PROVA 2		
24		29/05/24	Def. de Algoritmo		
25		03/06/24	Ling decidíveis		
26		05/06/24	Ling não-Turing reconhecíveis; o problema da parada; Ling co-Turing-reconhecíveis		
27		10/06/24	Problemas Indecidíveis		
28		12/06/24	Problemas Indecidíveis – exercícios		
29		17/06/24	Redutibilidade por mapeamento		
30		19/06/24	Complexidade		
31		24/06/24	PROVA 3		
32		26/06/24	PROVA SUB		
33		08/07/24	PROVA REC		

Aulas anteriores

Máquinas de Turing – Definição formal

Máquina pára IMEDIATAMENTE

Uma ***máquina de Turing*** é uma 7-upla, $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{aceita}}, q_{\text{rejeita}})$, onde Q, Σ, Γ são todos conjuntos finitos e

1. Q é o conjunto de estados,
2. Σ é o alfabeto de entrada sem o ***símbolo em branco*** $\sqcup$,
3. Γ é o alfabeto de fita, onde $\sqcup \in \Gamma$ e $\Sigma \subseteq \Gamma$,
4. $\delta: Q \times \Gamma \longrightarrow Q \times \Gamma \times \{\text{E}, \text{D}\}$ é a função de transição,
5. $q_0 \in Q$ é o estado inicial,
6. $q_{\text{aceita}} \in Q$ é o estado de aceitação, e
7. $q_{\text{rejeita}} \in Q$ é o estado de rejeição, onde $q_{\text{rejeita}} \neq q_{\text{aceita}}$.

Mais precisamente:

$\delta: Q' \times \Gamma \longrightarrow Q \times \Gamma \times \{\text{E}, \text{D}\}$, onde Q' é Q sem q_{aceita} e q_{rejeita}

Máquinas de Turing – Definição formal

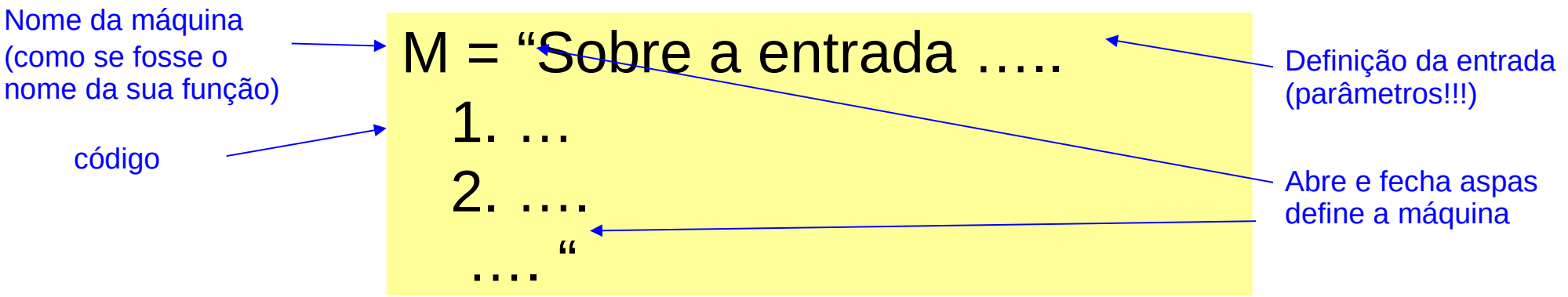
Nesta definição ela é determinística ou não determinística? **DETERMINÍSTICA!!!**

Uma ***máquina de Turing*** é uma 7-upla, $(Q, \Sigma, \Gamma, \delta, q_0, q_{\text{aceita}}, q_{\text{rejeita}})$, onde Q, Σ, Γ são todos conjuntos finitos e

1. Q é o conjunto de estados,
2. Σ é o alfabeto de entrada sem o ***símbolo em branco*** $\sqcup$,
3. Γ é o alfabeto de fita, onde $\sqcup \in \Gamma$ e $\Sigma \subseteq \Gamma$,
4. $\delta: Q \times \Gamma \longrightarrow Q \times \Gamma \times \{\text{E}, \text{D}\}$ é a função de transição,
5. $q_0 \in Q$ é o estado inicial,
6. $q_{\text{aceita}} \in Q$ é o estado de aceitação, e
7. $q_{\text{rejeita}} \in Q$ é o estado de rejeição, onde $q_{\text{rejeita}} \neq q_{\text{aceita}}$.

$\delta: Q' \times \Gamma \longrightarrow Q \times \Gamma \times \{\text{E}, \text{D}\}$, onde Q' é Q sem q_{aceita} e q_{rejeita}

Máquinas de Turing - Tipos de descrições



2-) Descrição de implementação (descrição em língua natural da manipulação da máquina)

3-) Descrição de alto-nível (algoritmo)

São como funções booleanas:

```
bool M(...){
  ...
}
```

que dentro do código precisam retornar *true* ou *false* dependendo da entrada

Isto é: *aceitar* ou *rejeitar* a entrada

Máquinas de Turing

A coleção de cadeias que M aceita é ***a linguagem de M***, ou ***a linguagem reconhecida por M***, denotada $L(M)$.

DEFINIÇÃO 3.5

Chame uma linguagem de ***Turing-reconhecível***, se alguma máquina de Turing a reconhece.[1]

1 - Ou linguagem **recursivamente enumerável** ou linguagem **irrestrita**

A máquina de Turing pode NÃO PARAR para algumas cadeias (neste caso a cadeia não pertence à linguagem reconhecida pela máquina)

Máquinas de Turing (MT) Decisoras

Uma MT é decisora se ela nunca entra em loop (isto é, sempre pára em um estado de aceitação ou de rejeição).

Dizemos que um decisor que reconhece uma linguagem decide essa linguagem.

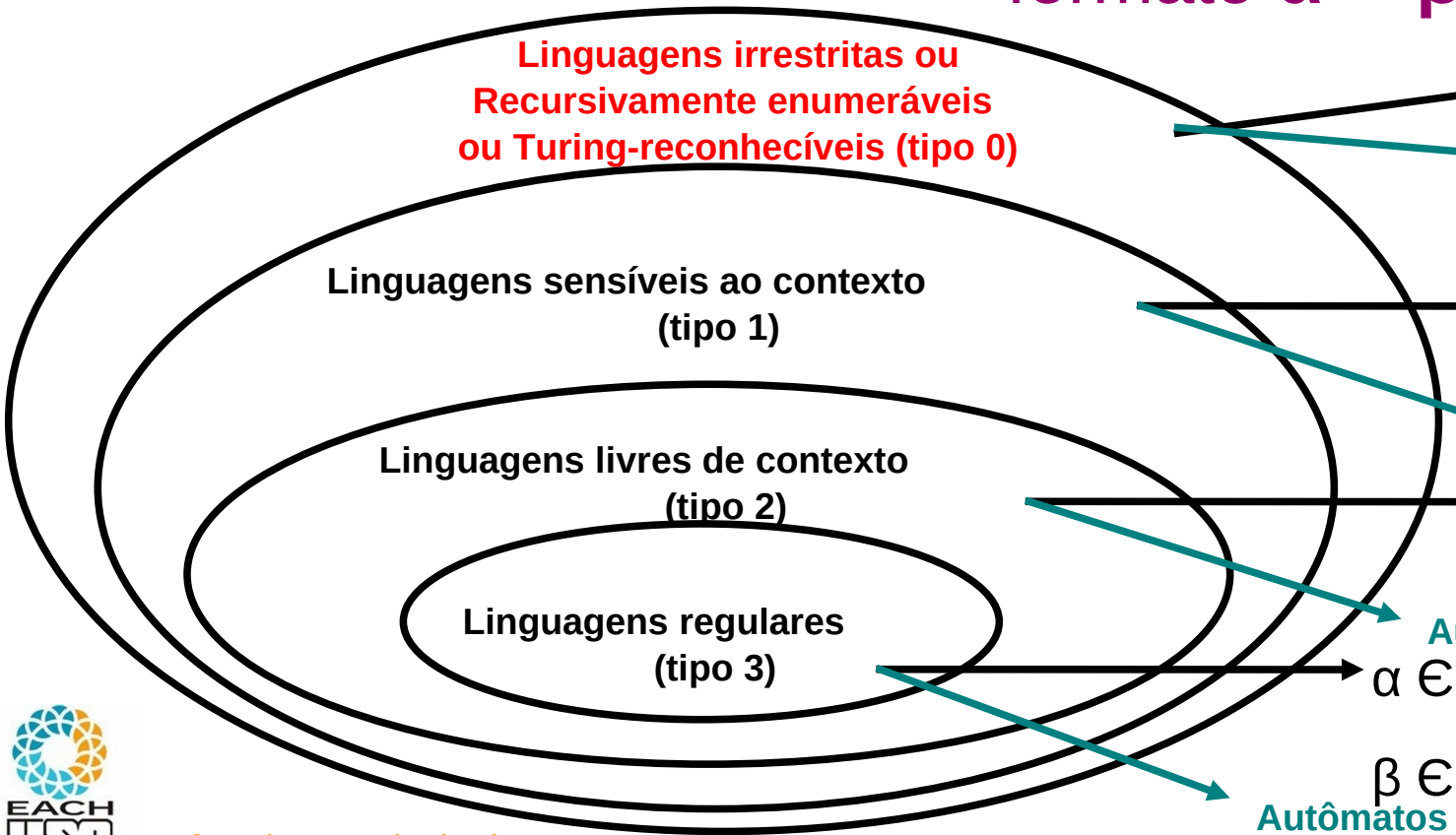
DEFINIÇÃO 3.6

Chame uma linguagem de ***Turing-decidível*** ou simplesmente ***decidível*** se alguma máquina de Turing a decide.[2]

2 - Ou linguagem **recursiva**

Hierarquia de Chomsky

Gramáticas com produções no formato $\alpha \rightarrow \beta$



$\alpha \in (V \cup \Sigma)^* V (V \cup \Sigma)^*$

$\beta \in (V \cup \Sigma)^*$

Máquina de Turing (fita ilimitada)

$\alpha \in (V \cup \Sigma)^* V (V \cup \Sigma)^*$

$\beta \in (V \cup \Sigma)^*$

$|\alpha| <= |\beta|$

Máquina de Turing com fita limitada

$\alpha \in V$

$\beta \in (V \cup \Sigma)^*$

Autômatos com pilha

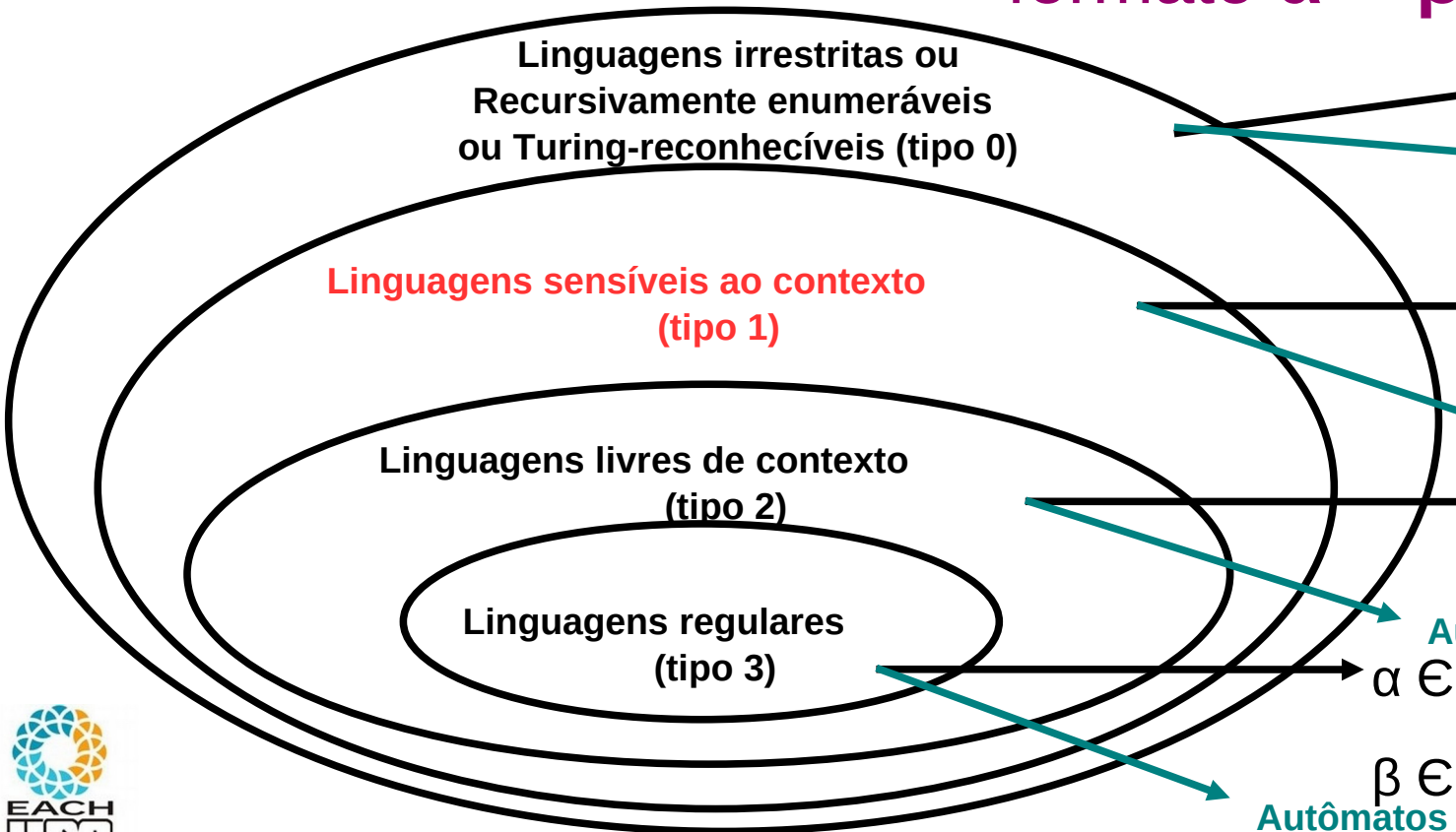
$\alpha \in V$

$\beta \in \Sigma_\varepsilon$, $\beta \in V$, $\beta \in (V\Sigma$ ou $\Sigma V)$

Autômatos finitos

Hierarquia de Chomsky

Gramáticas com produções no formato **α → β**



α Є (VUΣ)*V(VUΣ)*

β Є (VUΣ)*

Máquina de Turing (fita ilimitada)

α Є (VUΣ)*V(VUΣ)*

β Є (VUΣ)*

|α| <= | β |

Máquina de Turing com fita limitada

α Є V

β Є (VUΣ)*

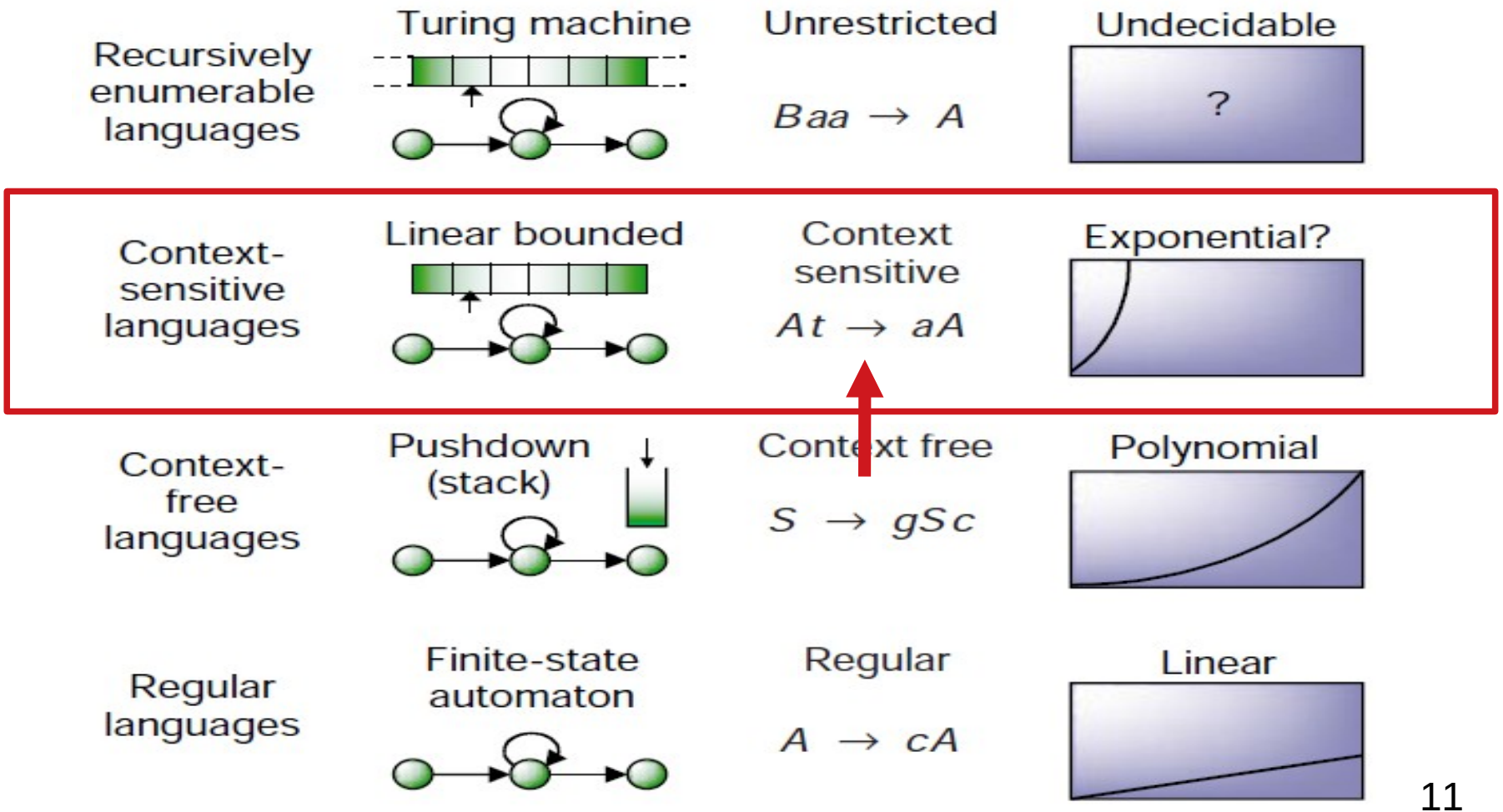
Autômatos com pilha

α Є V

β Є Σ_ε, β Є V, β Є (VΣ ou ΣV)

Autômatos finitos

Linguagens, modelos computacionais (dispositivos, gramáticas) e suas complexidades



Aula de hoje

3.2 – Variantes de Máquinas de Turing (parte 1)

Motivação

- Uma Máquina de Turing é:
  - Um dispositivo de reconhecimento de linguagens irrestritas
  - Um modelo teórico com o qual podemos estudar computabilidade
    - Tudo o que é computável por uma MT também o é por um computador moderno e vice-versa
- Saber variantes de MT nos dá opções para estudar cada problema
  - Dependendo do problema é mais fácil usar certas variantes

Variantes de Máquinas de Turing

Máquina de Turing é um modelo robusto: ela e suas variações reconhecem a mesma classe de linguagens

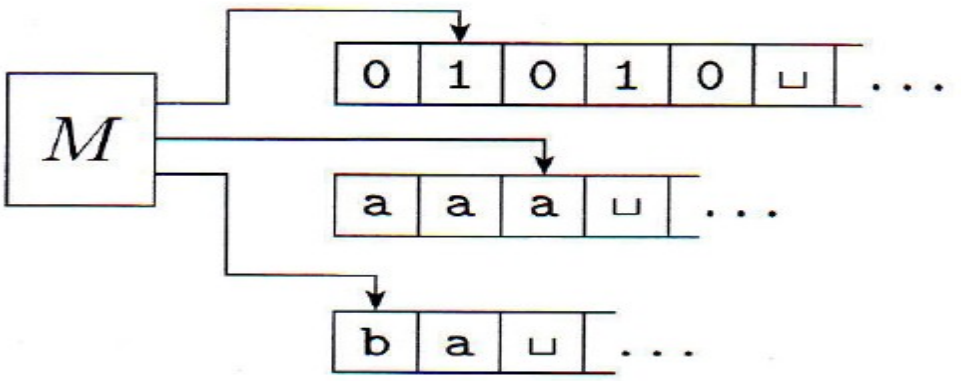
Máquinas de Turing Multifita

- K fitas
- Cada fita tem sua própria cabeça para leitura e escrita que se move independentemente das outras
- Inicialmente, a cadeia de entrada fica na fita 1, e as demais fitas com branco, todas com a cabeça de fita na primeira posição

$$\delta : Q \times \Gamma^k \longrightarrow Q \times \Gamma^k \times \{\mathrm{E}, \mathrm{D}, \mathrm{P}\}^k$$

$$\delta(q_i, a_1, \ldots, a_k) = (q_j, b_1, \ldots, b_k, \mathrm{E}, \mathrm{D}, \ldots, \mathrm{E})$$

Ex: k = 3



TEOREMA 3.13

Toda máquina de Turing multifita tem uma máquina de Turing de uma única fita que lhe é equivalente.

Similar a termos computador com vários ou apenas um processador...

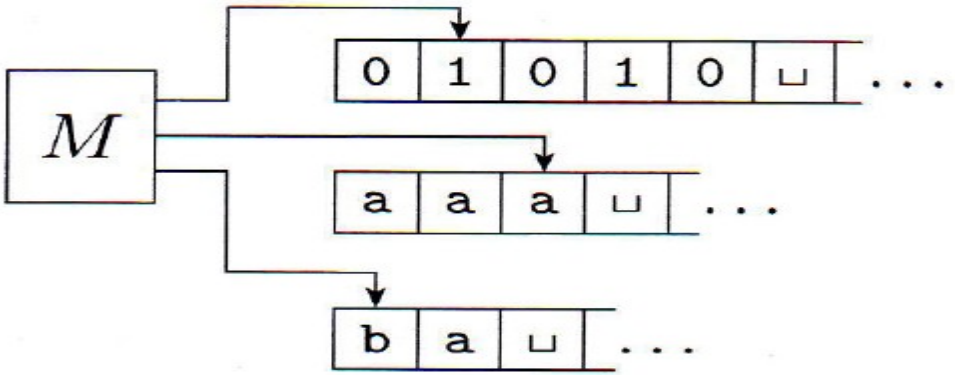
TEOREMA 3.13

Toda máquina de Turing multifita tem uma máquina de Turing de uma única fita que lhe é equivalente.

PROVA

Seja S uma MT de fita única e M uma MT de k fitas.

S pode simular M:



COMO?

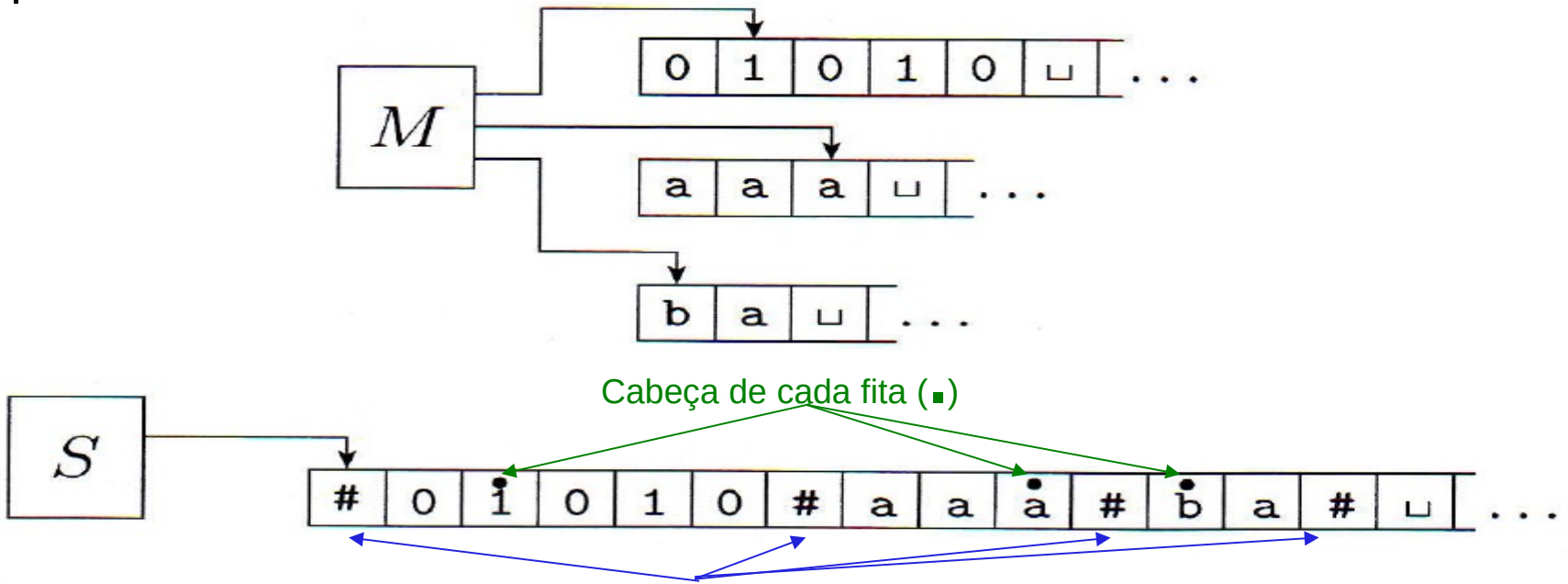
TEOREMA 3.13

Toda máquina de Turing multifita tem uma máquina de Turing de uma única fita que lhe é equivalente.

PROVA

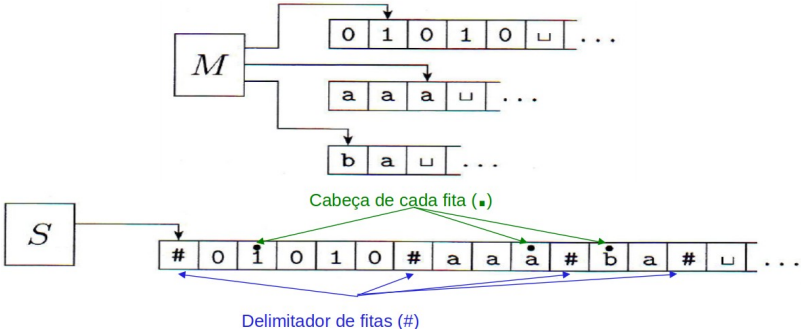
Seja S uma MT de fita única e M uma MT de k fitas.

S pode simular M:



Funcionamento de S:

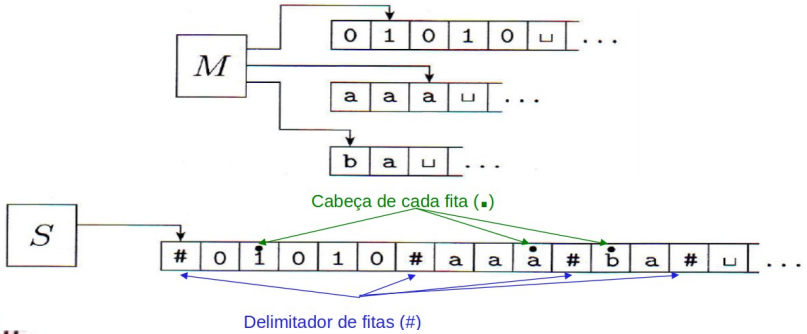
- Preparação da fita: (ex: $w = w_1 \cdots w_n$)



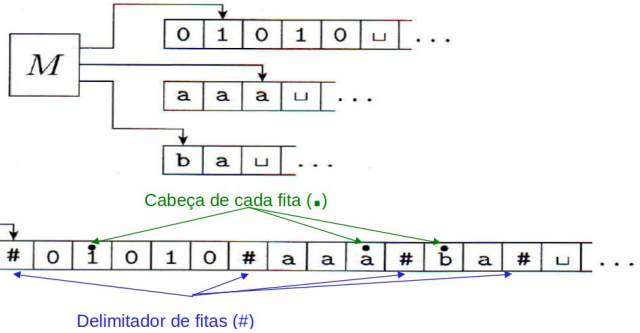
Funcionamento de S:

- Preparação da fita: (ex: $w = w_1 \cdots w_n$)

$$\#\dot{w}_1 w_2 \cdots w_n \#\dot{\sqcup}\#\dot{\sqcup}\# \cdots \#$$



Funcionamento de S:

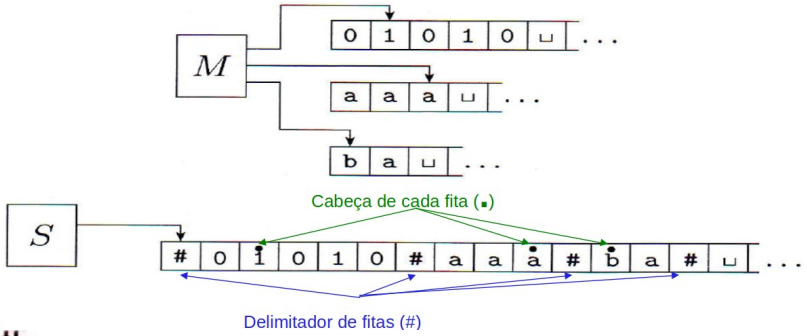


- Preparação da fita: (ex: $w = w_1 \cdots w_n$)

$$\#\dot{w}_1 w_2 \cdots w_n \#\dot{\sqcup}\#\dot{\sqcup}\# \cdots \#$$

- Leitura dos símbolos atuais:

Funcionamento de S:

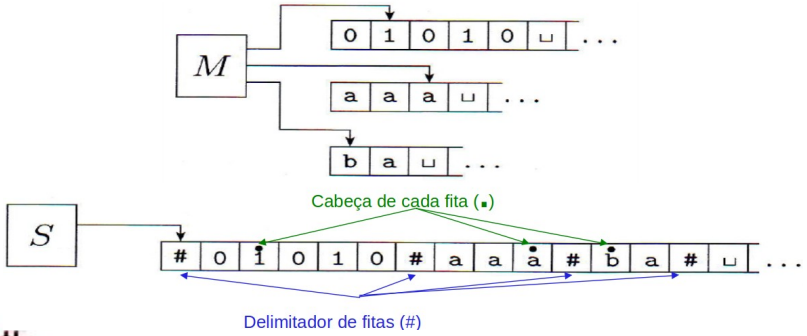


- Preparação da fita: (ex: $w = w_1 \cdots w_n$)

$$\#\dot{w}_1 w_2 \cdots w_n \#\dot{\sqcup}\#\dot{\sqcup}\# \cdots \#$$

- Leitura dos símbolos atuais: percorre a fita lendo os símbolos com ponto em cima (até o (k+1)-ésimo #)

Funcionamento de S:

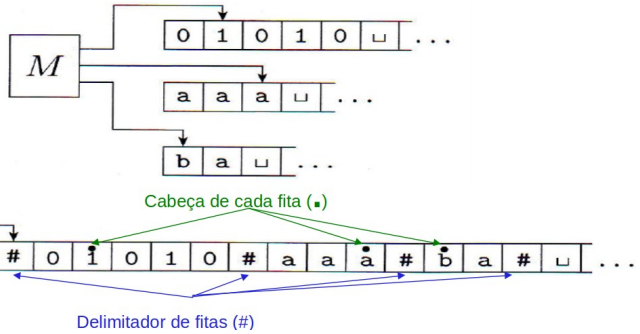


- Preparação da fita: (ex: $w = w_1 \cdots w_n$)

$$\#\dot{w}_1 w_2 \cdots w_n \#\dot{\sqcup}\#\dot{\sqcup}\# \cdots \#$$

- Leitura dos símbolos atuais: percorre a fita lendo os símbolos com ponto em cima (até o (k+1)-ésimo #)
- Atualização das cabeças:

Funcionamento de S:

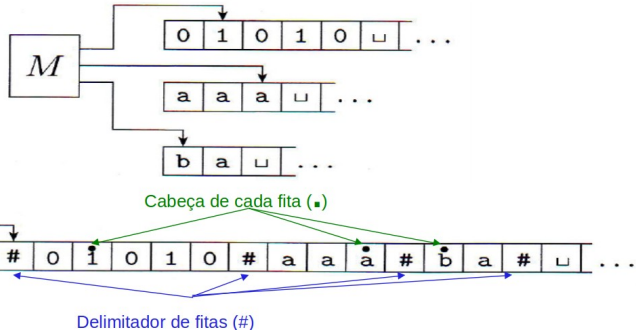


- Preparação da fita: (ex: $w = w_1 \cdots w_n$)

$$\#\dot{w}_1 w_2 \cdots w_n \#\dot{\sqcup}\#\dot{\sqcup}\# \cdots \#$$

- Leitura dos símbolos atuais: percorre a fita lendo os símbolos com ponto em cima (até o (k+1)-ésimo #)
- Atualização das cabeças: percorre a fita fazendo as atualizações conforme a função de transição (tirando e colocando pontos para atualizar as cabeças de fitas), até o (k+1)-ésimo #

Funcionamento de S:

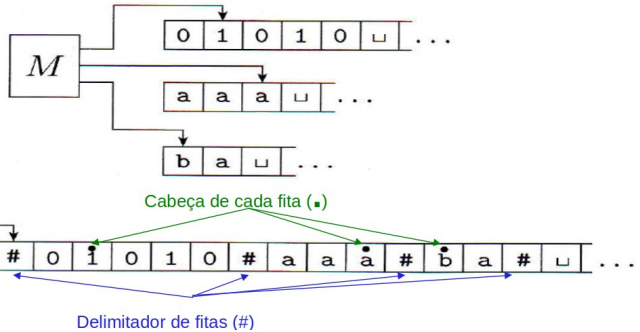


- Preparação da fita: (ex: $w = w_1 \cdots w_n$)

$$\#\dot{w}_1 w_2 \cdots w_n \#\dot{\sqcup}\#\dot{\sqcup}\# \cdots \#$$

- Leitura dos símbolos atuais: percorre a fita lendo os símbolos com ponto em cima (até o (k+1)-ésimo #)
- Atualização das cabeças: percorre a fita fazendo as atualizações conforme a função de transição (tirando e colocando pontos para atualizar as cabeças de fitas), até o (k+1)-ésimo #
- Se uma cabeça de fita vai (em um movimento para a direita) para um “#”:

Funcionamento de S:



- Preparação da fita: (ex: $w = w_1 \cdots w_n$)

$$\#\dot{w}_1 w_2 \cdots w_n \#\dot{\sqcup}\#\dot{\sqcup}\# \cdots \#$$

- Leitura dos símbolos atuais: percorre a fita lendo os símbolos com ponto em cima (até o (k+1)-ésimo #)
- Atualização das cabeças: percorre a fita fazendo as atualizações conforme a função de transição (tirando e colocando pontos para atualizar as cabeças de fitas), até o (k+1)-ésimo #
- Se uma cabeça de fita vai (em um movimento para a direita) para um “#”: desloca o conteúdo da fita para a direita e coloca o símbolo “branco” no lugar daquele “#”

COROLÁRIO 3.15

Uma linguagem é Turing-reconhecível se e somente se alguma máquina de Turing multifita a reconhece.

COROLÁRIO 3.15

Uma linguagem é Turing-reconhecível se e somente se alguma máquina de Turing multifita a reconhece.

Prova:

Linguagem L é TR => MTM reconhece L:

COROLÁRIO 3.15

Uma linguagem é Turing-reconhecível se e somente se alguma máquina de Turing multifita a reconhece.

Prova:

Linguagem L é TR => MTM reconhece L:

L é TR => existe uma MT de fita única que a reconhece => existe uma MT multifita que a reconhece (pois fita única é um caso especial de multifita)

COROLÁRIO 3.15

Uma linguagem é Turing-reconhecível se e somente se alguma máquina de Turing multifita a reconhece.

Prova:

Linguagem L é TR => MTM reconhece L:

L é TR => existe uma MT de fita única que a reconhece => existe uma MT multifita que a reconhece (pois fita única é um caso especial de multifita)

MTM reconhece L => Linguagem L é TR:

COROLÁRIO 3.15

Uma linguagem é Turing-reconhecível se e somente se alguma máquina de Turing multifita a reconhece.

Prova:

Linguagem L é TR => MTM reconhece L:

L é TR => existe uma MT de fita única que a reconhece => existe uma MT multifita que a reconhece (pois fita única é um caso especial de multifita)

MTM reconhece L => Linguagem L é TR:

MTM reconhece L => uma MT fita única a reconhece (pelo teorema) => L é TR

Máquinas de Turing Não-Determinísticas

Máquinas de Turing Não-Determinísticas

$$\delta : Q \times \Gamma \longrightarrow \mathcal{P}(Q \times \Gamma \times \{\mathrm{E}, \mathrm{D}\}).$$

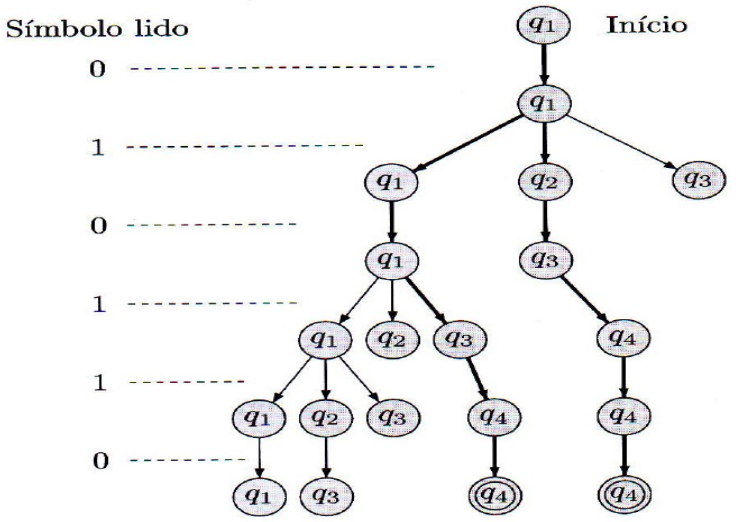
TEOREMA 3.16

Toda máquina de Turing não-determinística tem uma máquina de Turing determinística que lhe é equivalente.

TEOREMA 3.16

Toda máquina de Turing não-determinística tem uma máquina de Turing determinística que lhe é equivalente.

A ideia de transformação de AFN para AFD não serve, pois aqui a MT escreve sobre a fita: cada ramo de computação produz, potencialmente, um conteúdo de fita diferente...



TEOREMA 3.16

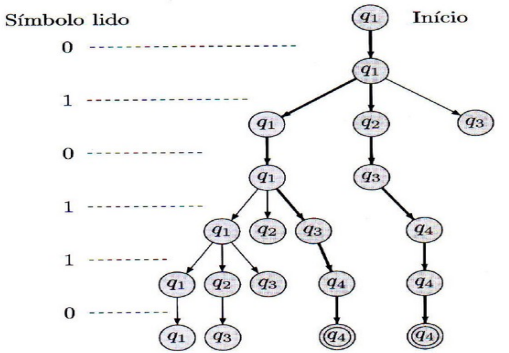
Toda máquina de Turing não-determinística tem uma máquina de Turing determinística que lhe é equivalente.

Ideia da prova:

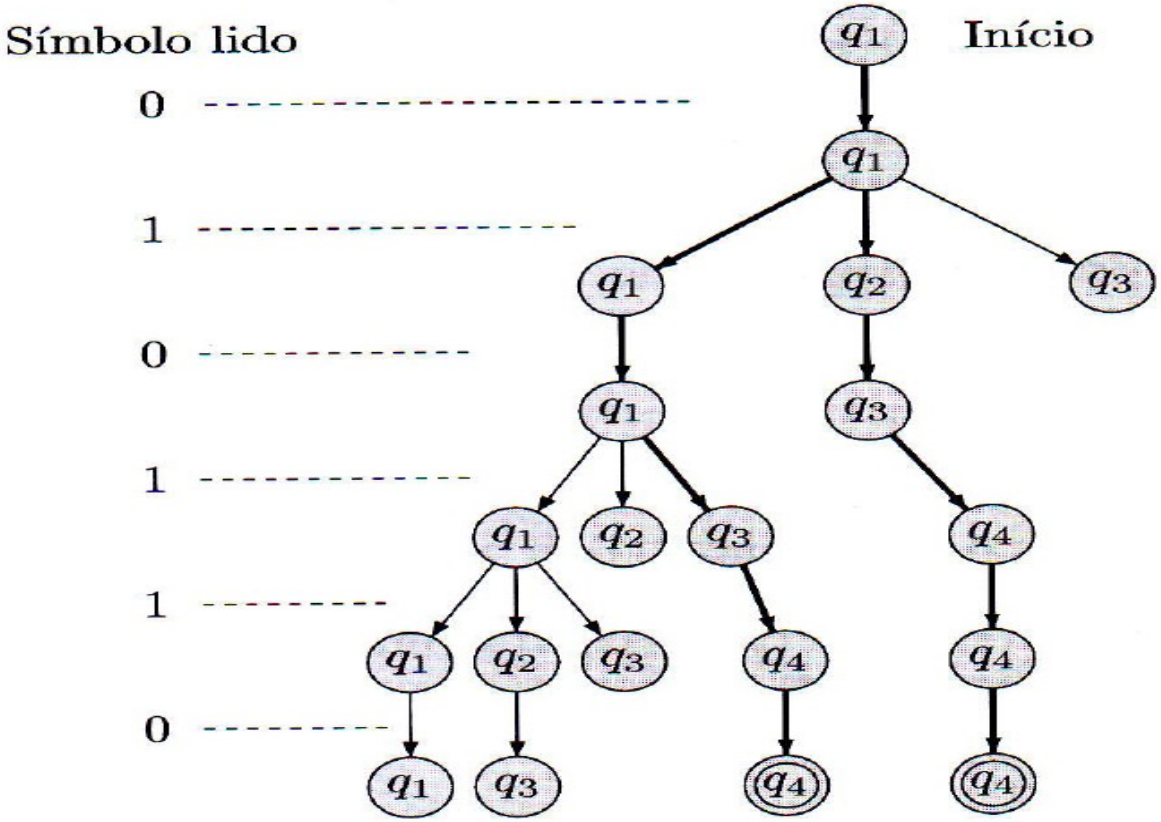
MT determinística D precisa simular todas as possíveis computações de uma MT não determinística N

Computação de N pode ser representada por uma árvore (filhos de um nó são as possibilidades de transição) (Lembram da árvore de computações de AFNs?)

Cada caminho (a partir da raiz) é uma computação possível

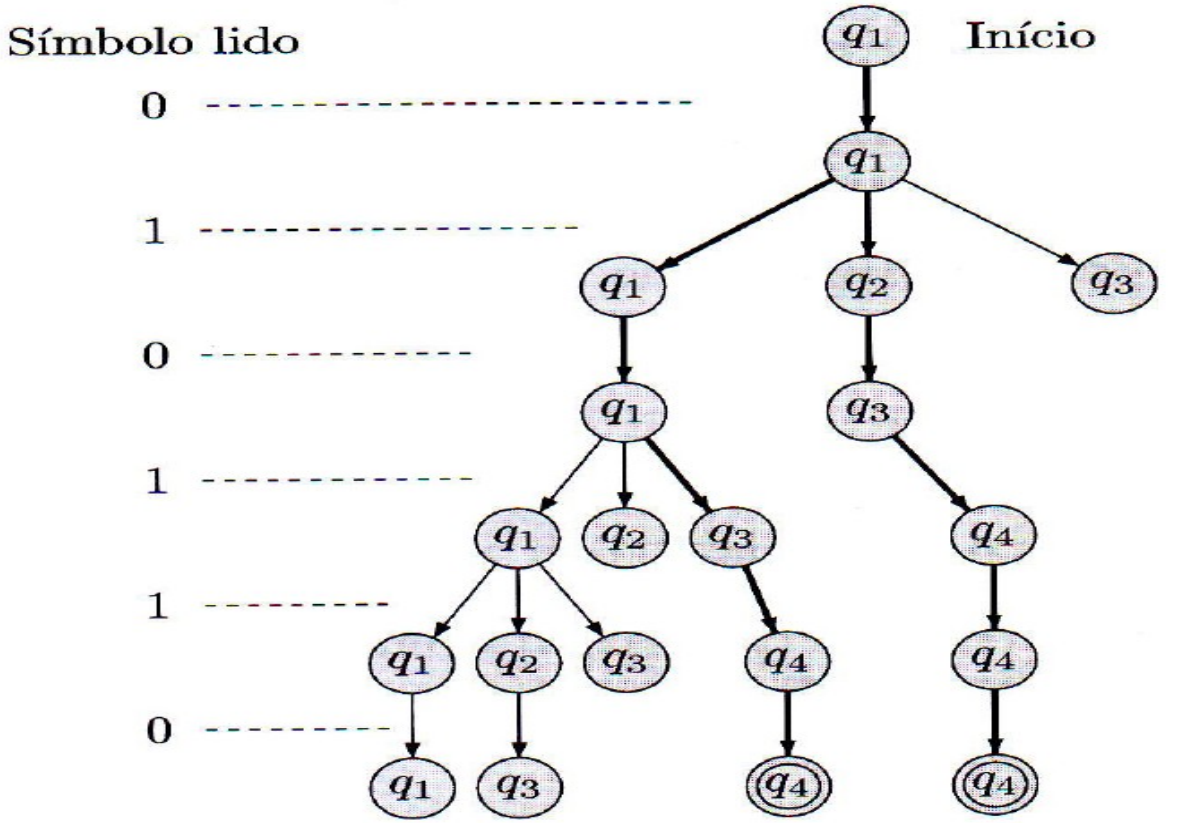


Como executar essas computações em uma máquina sequencial?



Como executar essas computações em uma máquina sequencial?

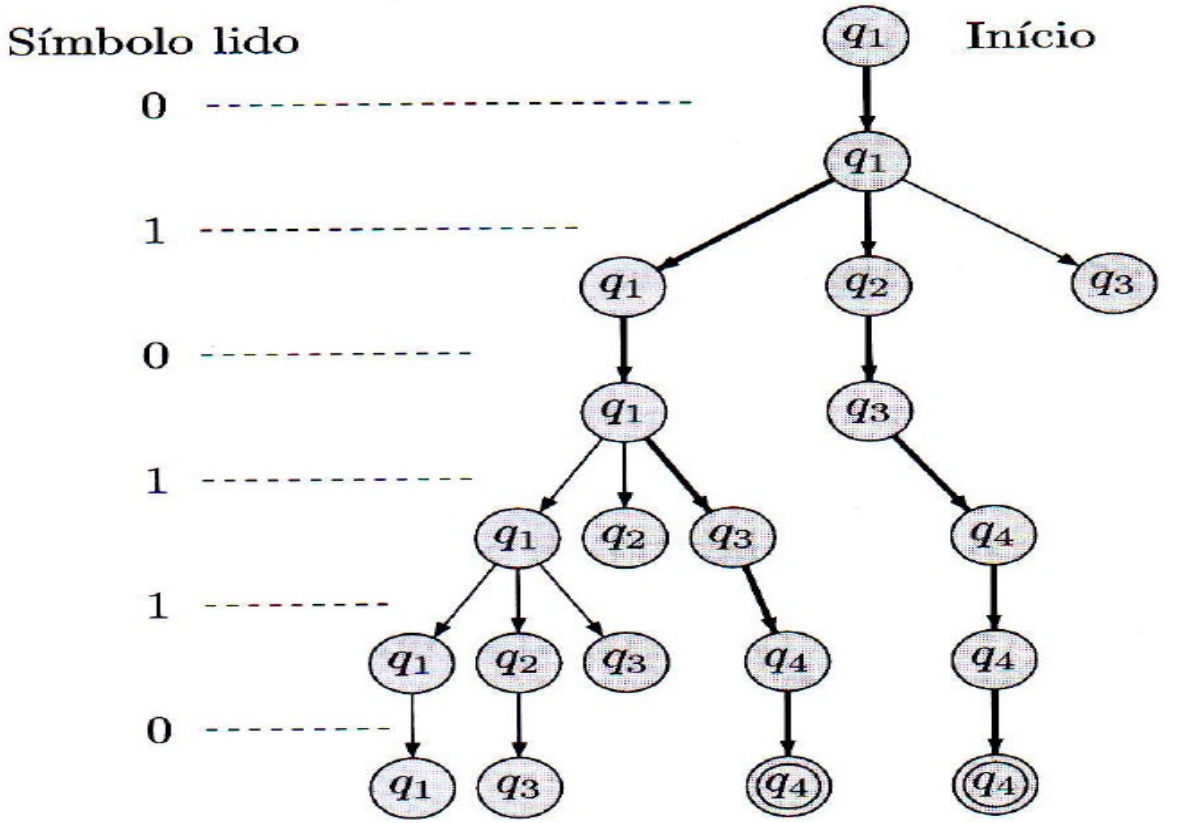
Lembre-se que, no caso de máquinas de Turing, um ramo pode não parar nunca…



Como executar essas computações em uma máquina sequencial?

Lembre-se que, no caso de máquinas de Turing, um ramo pode não parar nunca…

VAMOS FAZER UM PROCESSAMENTO "EM LARGURA" !!!



TEOREMA 3.16

Toda máquina de Turing não-determinística tem uma máquina de Turing determinística que lhe é equivalente.

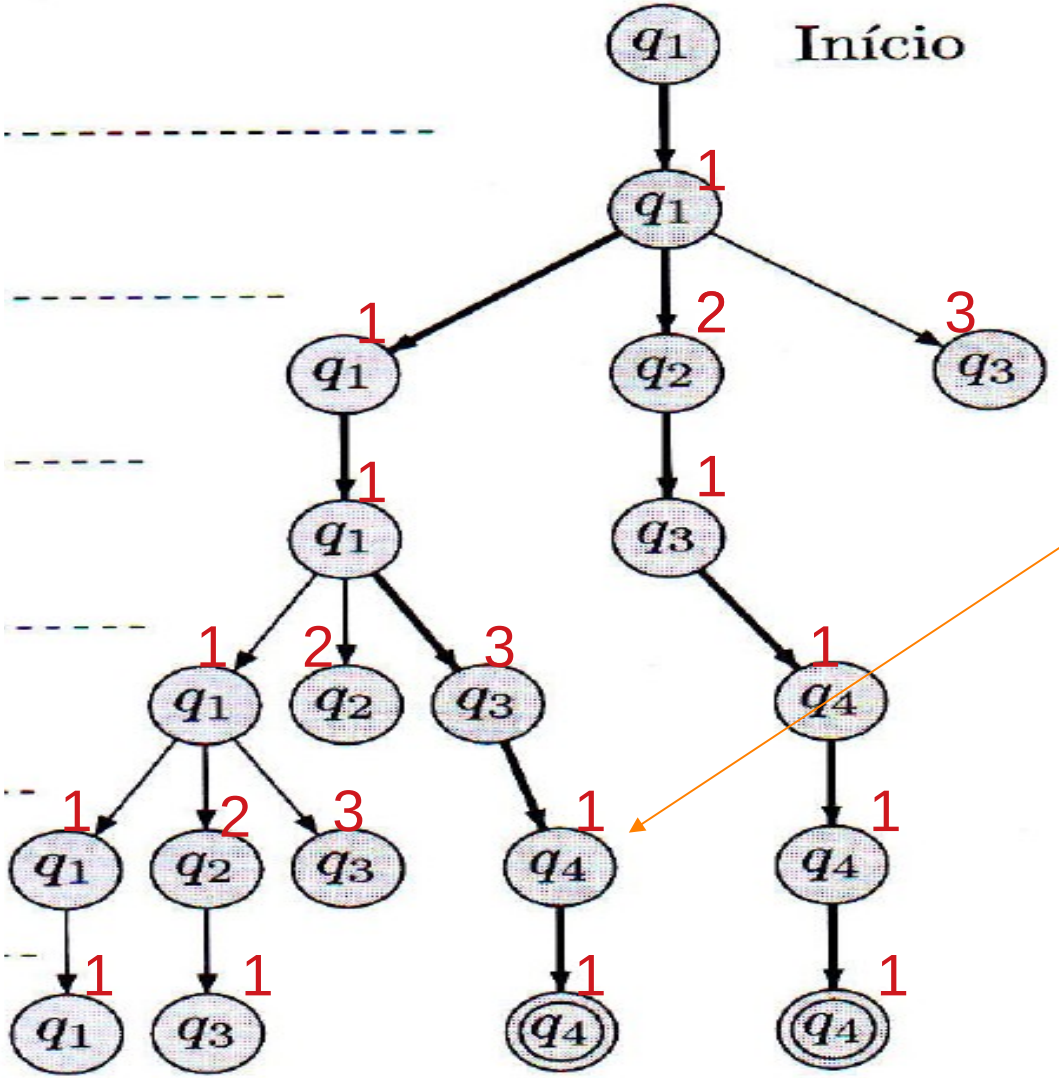
Ideia da prova:

Usar para MT determinística D para simular todas as possíveis computações de uma MT não determinística N

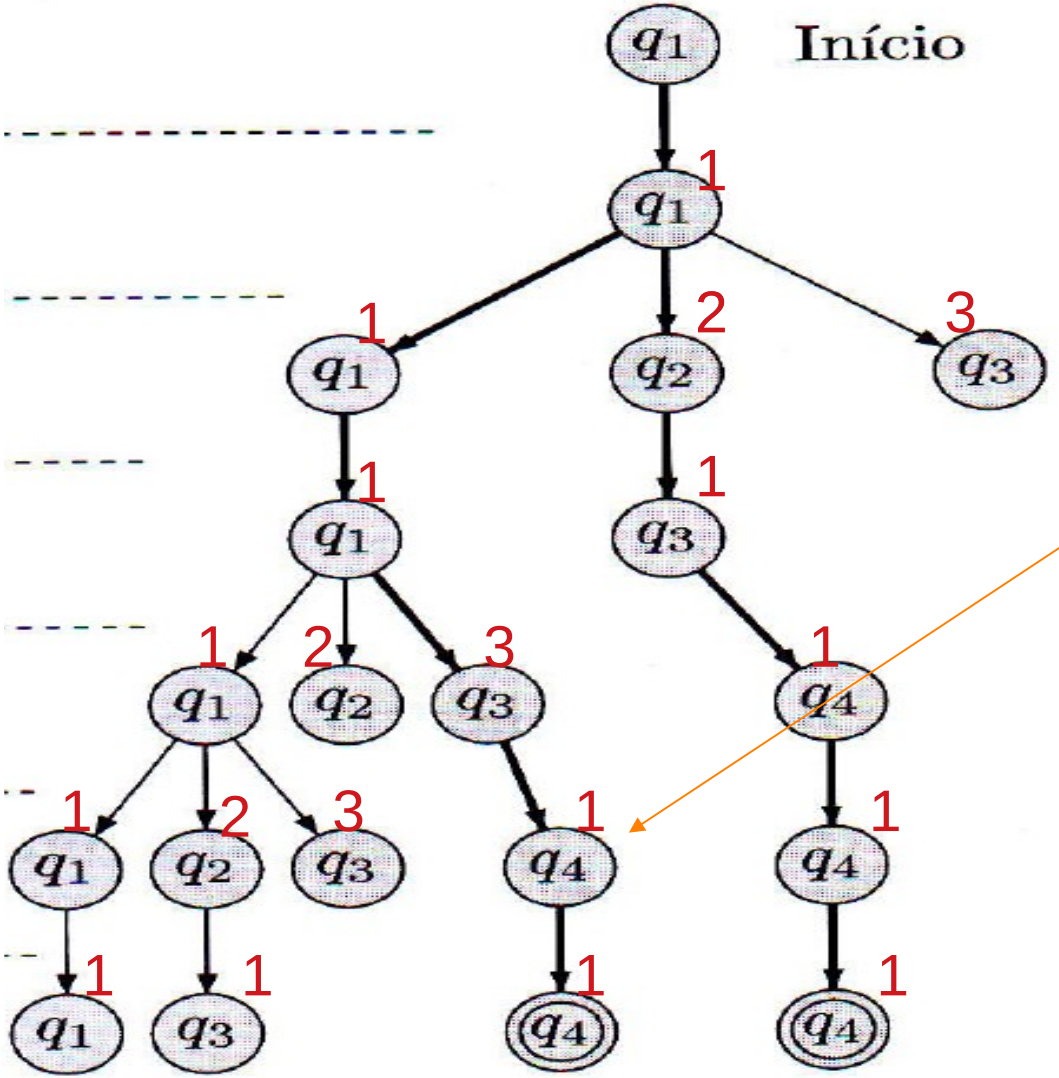
Computação de N pode ser representada por uma árvore (filhos de um nó são as possibilidades de transição)

Cada caminho (a partir da raiz) é uma computação possível

Cada nó dessa árvore terá um “endereço” que indica o caminho da raiz até ele, ou seja, qual filho seguir a partir da raiz. Ex: 314 → terceiro filho da raiz, depois primeiro filho do nó atual, depois quarto filho do nó atual



Ex: qual o “endereço” deste nó?



Ex: qual o “endereço” deste nó?

11131

TEOREMA 3.16

Toda máquina de Turing não-determinística tem uma máquina de Turing determinística que lhe é equivalente.

Ideia da prova:

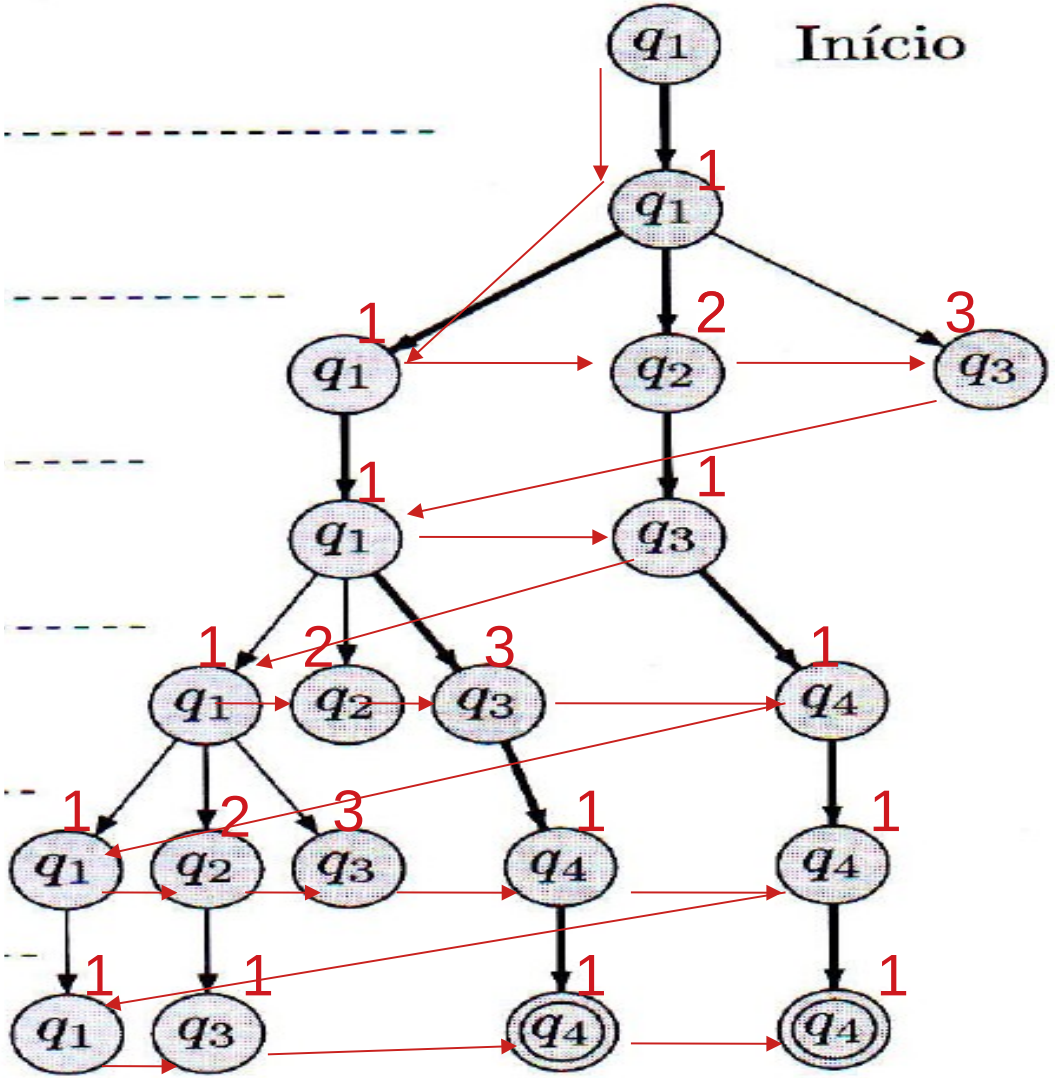
Usar para MT determinística D para simular todas as possíveis computações de uma MT não determinística N

Computação de N representada por uma árvore (filhos de um nó são as possibilidades de transição)

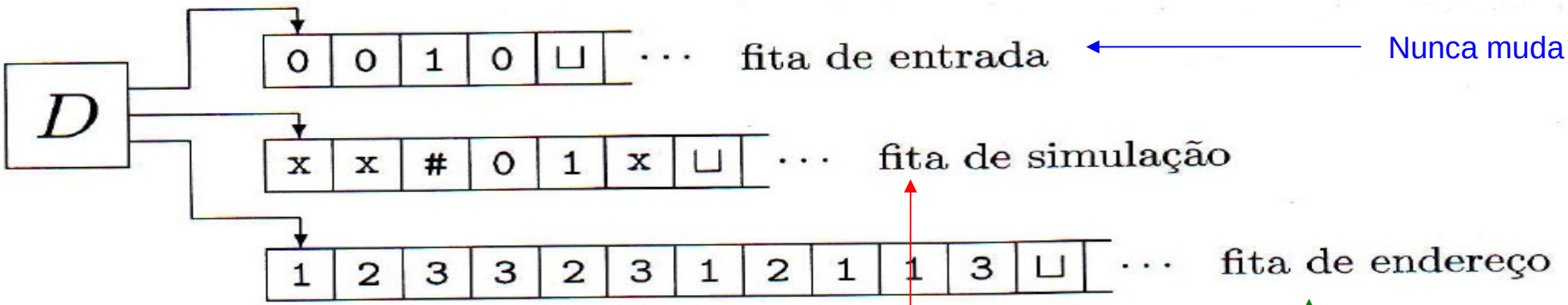
Cada caminho (a partir da raiz) é uma computação possível

Cada nó dessa árvore terá um endereço que indica o caminho da raiz até ele, ou seja, qual filho seguir a partir da raiz. Ex: 314 → terceiro filho da raiz, depois primeiro filho do nó atual, depois quarto filho do nó atual

D irá percorrer essa árvore por meio de uma busca em largura (por nível, da raiz às folhas) para impedir que caia em um ramo infinito


Início
q_1
q_1 1
q_1 1
q_2 2
q_3 3
q_1 1
q_3 1
q_1 1
q_2 2
q_3 3
q_4 1
q_1 1
q_2 2
q_3 3
q_4 1
q_4 1
q_1 1
q_3 1
q_4 1
q_4 1

Prova – usaremos uma MT multifita (3 fitas)

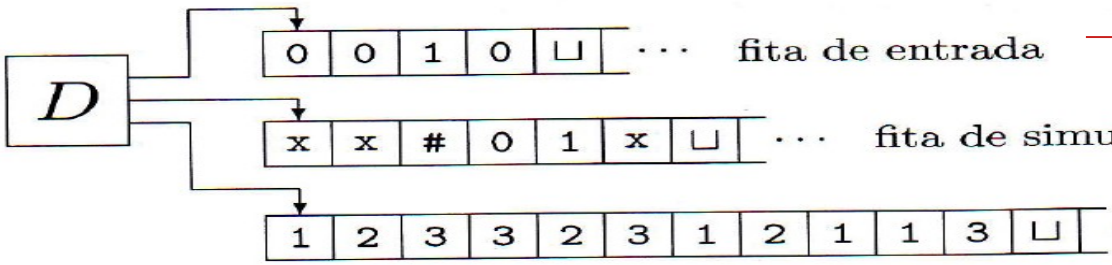


Usada para indicar em qual computação (caminho) estou, e até onde ir (o "endereço" de um nó)

Possui uma cadeia em Σ_b^* , $\Sigma_b = \{1, 2, \ldots, b\}$, sendo b o maior número de filhos de um nó (ie, o maior número de transições alternativas de um estado)

Prova

alfabetos das fitas



Γ_N

Γ_N

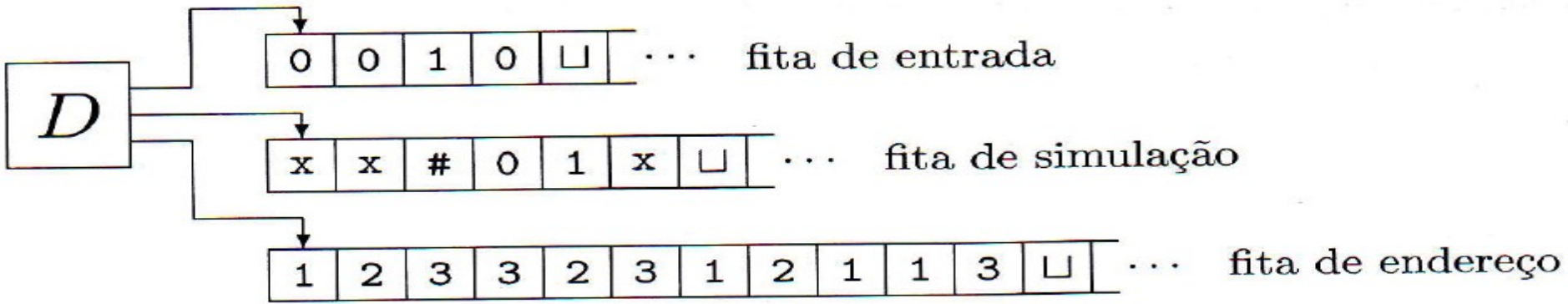
Σ_b U {branco}

1. Inicialmente, a fita 1 contém a entrada w e as fitas 2 e 3 estão vazias.
2. Copie a fita 1 para a fita 2.
3. Use a fita 2 para simular N com a entrada w sobre um ramo de sua computação não-determinística. Antes de cada passo de N, consulte o próximo símbolo na fita 3 para determinar qual escolha fazer entre aquelas permitidas pela função de transição de N. Se não restam mais símbolos na fita 3 ou se essa escolha não-determinística for inválida, aborte esse ramo indo para o estágio 4. Também vá para o estágio 4 se uma configuração de rejeição for encontrada. Se uma configuração de aceitação for encontrada, *aceite* a entrada.
4. Substitua a cadeia na fita 3 pela próxima cadeia na ordem lexicográfica. Simule o próximo ramo da computação de N indo para o estágio 2.

Máquinas de Turing Não-Determinísticas

$$\delta : Q \times \Gamma \longrightarrow \mathcal{P}(Q \times \Gamma \times \{\mathrm{E}, \mathrm{D}\}).$$

Simulação de uma MTND por uma MT Determinística:

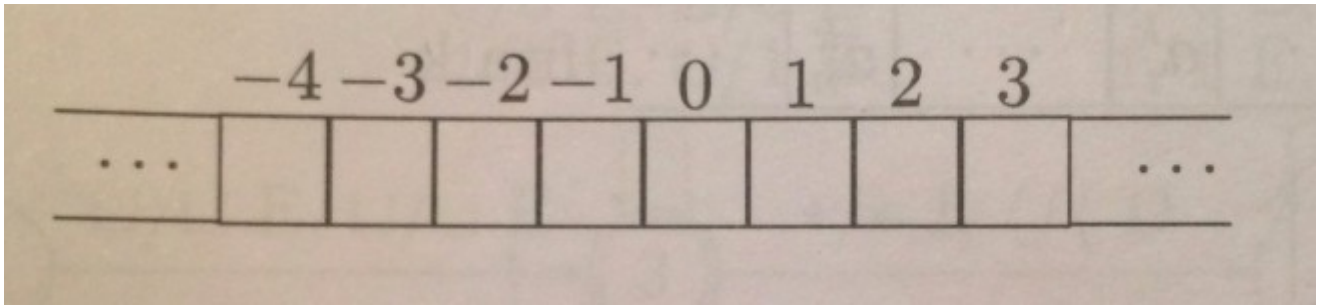


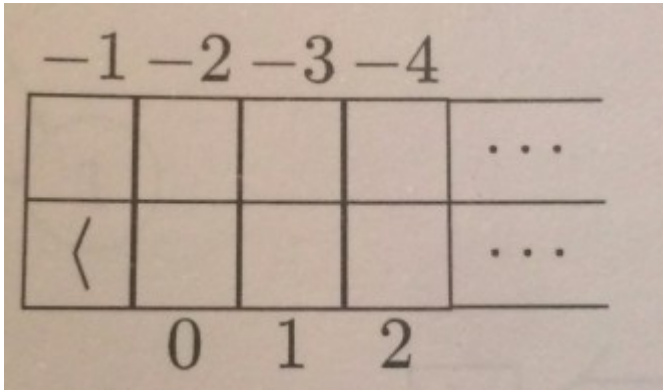
Uma cadeia é aceita se ALGUM ramo da computação a aceita
Uma cadeia é rejeitada se NENHUM ramo da computação a aceita (NÃO basta um ramo rejeitar)
D entrará em loop se N também entra

COROLÁRIO 3.18

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

Máquinas de Turing com fita ilimitada em ambos os lados

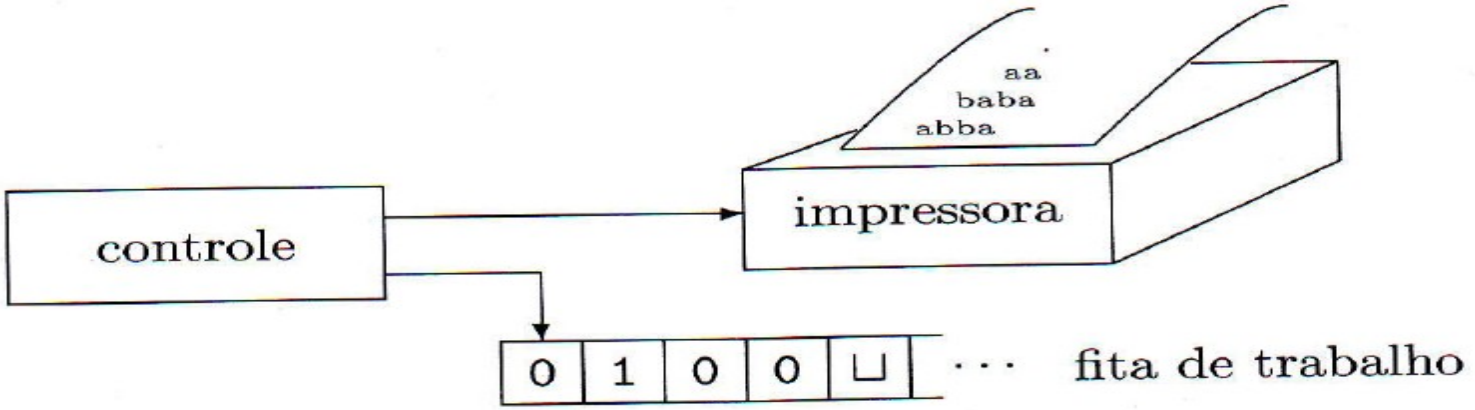




Exercício: provem a equivalência

Prova no livro do Vieira (2006), pag 226

Enumeradores



Duas fitas, sendo uma só de escrita (impressora)

Começa com a fita impressora em branco

Imprime cadeias da linguagem, em qualquer ordem, possivelmente com repetição

Controle de estados: similar à da máquina de Turing

TEOREMA 3.21

Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera.

TEOREMA 3.21

Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera.

Prova: (<=)

Temos um enumerador E de uma linguagem L

Uma MT M que reconhece L (utilizando E) funciona assim:

TEOREMA 3.21

Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera.

Prova: (<=)

Temos um enumerador E de uma linguagem L

Uma MT M que reconhece L (utilizando E) funciona assim:

M = “Sobre a cadeia de entrada w,

TEOREMA 3.21

Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera.

Prova: (<=)

Temos um enumerador E de uma linguagem L

Uma MT M que reconhece L (utilizando E) funciona assim:

M = “Sobre a cadeia de entrada w,

1. rode E e compare a cadeia enumerada com w;
2. se forem iguais, *aceite* w; se forem diferentes volte ao passo 1;
3. se acabar a enumeração, *rejeite* w.”

TEOREMA 3.21

Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera.

Prova: (<=)

Temos um enumerador E de uma linguagem L

Uma MT M que reconhece L (utilizando E) funciona assim:

M = “Sobre a cadeia de entrada w,

1. rode E e compare a cadeia enumerada com w;
2. se forem iguais, *aceite* w; se forem diferentes volte ao passo 1;
3. se acabar a enumeração, *rejeite* w.”

Se w pertence a L,

Se w não pertence a L,

TEOREMA 3.21

Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera.

Prova: (<=)

Temos um enumerador E de uma linguagem L

Uma MT M que reconhece L (utilizando E) funciona assim:

M = “Sobre a cadeia de entrada w,

1. rode E e compare a cadeia enumerada com w;
2. se forem iguais, *aceite* w; se forem diferentes volte ao passo 1;
3. se acabar a enumeração, *rejeite* w.”

Se w pertence a L, w será enumerada em algum momento e portanto será aceita

Se w não pertence a L,

TEOREMA 3.21

Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera.

Prova: (<=)

Temos um enumerador E de uma linguagem L

Uma MT M que reconhece L (utilizando E) funciona assim:

M = “Sobre a cadeia de entrada w,

1. rode E e compare a cadeia enumerada com w;
2. se forem iguais, *aceite* w; se forem diferentes volte ao passo 1;
3. se acabar a enumeração, *rejeite* w.”

Se w pertence a L, w será enumerada em algum momento e portanto será aceita

Se w não pertence a L,

M rejeita se L for finita e o enumerador pára em algum momento

M não pára se L for infinita ou se o enumerador não pára (mesmo L finita)

TEOREMA 3.21

Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera.

Prova: (<=)

Temos um enumerador E de uma linguagem L

Uma MT M que reconhece L (utilizando E) funciona assim:

M = “Sobre a cadeia de entrada w,

1. rode E e compare a cadeia enumerada com w;
2. se forem iguais, *aceite* w; se forem diferentes volte ao passo 1;
3. se acabar a enumeração, *rejeite* w.”

Se w pertence a L, w será enumerada em algum momento e portanto será aceita

Se w não pertence a L,

M rejeita se L for finita e o enumerador pára em algum momento

M não pára se L for infinita ou se o enumerador não pára (mesmo L finita)

Problema?

TEOREMA 3.21

Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera.

Prova: (<=)

Temos um enumerador E de uma linguagem L

Uma MT M que reconhece L (utilizando E) funciona assim:

M = “Sobre a cadeia de entrada w,

1. rode E e compare a cadeia enumerada com w;
2. se forem iguais, *aceite* w; se forem diferentes volte ao passo 1;
3. se acabar a enumeração, *rejeite* w.”

Se w pertence a L, w será enumerada em algum momento e portanto será aceita

Se w não pertence a L,

M rejeita se L for finita e o enumerador pára em algum momento

M não pára se L for infinita ou se o enumerador não pára (mesmo L finita)

Problema?
NÃO!
Não parar sobre cadeias que **não** pertençam à linguagem tudo bem (pois foi dito aqui que a linguagem é Turing-**reconhecível**, e não Turing-*decidível*)

TEOREMA 3.21

Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera.

Prova: (=>) Temos uma Máquina de Turing M

O enumerador E funciona assim (usando M!):

.

.

.

.

TEOREMA 3.21

Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera.

Prova: (=>) Temos uma Máquina de Turing M

O enumerador E funciona assim (usando M!):

Seja s_1, s_2, s_3, … sequências de Σ* (ordem crescente de comprimento e ordem lexicográfica)

.

.

TEOREMA 3.21

Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera.

Prova: (=>) Temos uma Máquina de Turing M

O enumerador E funciona assim (usando M!):

Seja s_1, s_2, s_3, … sequências de Σ* (ordem crescente de comprimento e ordem lexicográfica)

- Ignore a entrada
- para i = 1, 2, 3, …

  Rode M sobre s_i

  Imprima s_i se for aceita por M

TEOREMA 3.21

Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera.

Prova: (=>) Temos uma Máquina de Turing M

O enumerador E funciona assim (usando M!):

Seja s_1, s_2, s_3, … sequências de Σ* (ordem crescente de comprimento e ordem lexicográfica)

- Ignore a entrada
- para i = 1, 2, 3, …

    Rode M sobre s_i

    Imprima s_i se for aceita por M

Problema?

TEOREMA 3.21

Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera.

Prova: (=>) Temos uma Máquina de Turing M

O enumerador E funciona assim (usando M!):

Seja s_1, s_2, s_3, … sequências de Σ* (ordem crescente de comprimento e ordem lexicográfica)

- Ignore a entrada
- para i = 1, 2, 3, …

    Rode M sobre s_i

    Imprima s_i se for aceita por M

Problema?

E se M não parar para uma dada s_i?

TEOREMA 3.21

Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera.

Prova: (=>) Temos uma Máquina de Turing M

O enumerador E funciona assim (usando M!):

Seja s_1, s_2, s_3, … sequências de Σ* (ordem crescente de comprimento e ordem lexicográfica)

- Ignore a entrada
- para i = 1, 2, 3, …

    Rode M sobre s_i

    Imprima s_i se for aceita por M

Problema?

E se M não parar para uma dada s_i? **Travará o enumerador!!!**

TEOREMA 3.21

Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera.

Prova: (=>) Temos uma Máquina de Turing M

O enumerador E funciona assim:

Seja s_1, s_2, s_3, … sequências de Σ* (ordem crescente de comprimento e ordem lexicográfica)

- Ignore a entrada
- para i = 1, 2, 3, …

  Rode M sobre s_i

  Imprima s_i se for aceita por M

Problema?

E se M não parar para uma dada s_i? **Travará o enumerador...**

TEOREMA 3.21

Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera.

Prova: (=>) Temos uma Máquina de Turing M

O enumerador E funciona assim:

Seja s_1, s_2, s_3, … sequências de Σ* (ordem crescente de comprimento e ordem lexicográfica)

- Ignore a entrada
- para i = 1, 2, 3, …

  Rode i passos de M sobre s_1, ..., s_i

  Imprima as s_j que foram aceitas

TEOREMA 3.21

Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera.

Prova: (=>) Temos uma Máquina de Turing M

O enumerador E funciona assim:

Seja s_1, s_2, s_3, … sequências de Σ* (ordem crescente de comprimento e ordem lexicográfica)

- Ignore a entrada
- para i = 1, 2, 3, …

  Rode i passos de M sobre s_1, ..., s_i

  Imprima as s_j que foram aceitas

Se w pertence a L, w é impressa um número infinito de vezes

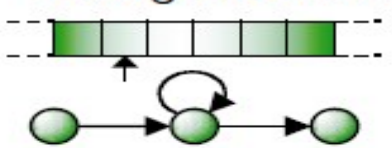
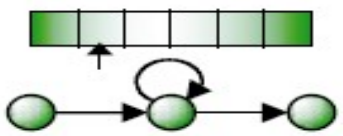
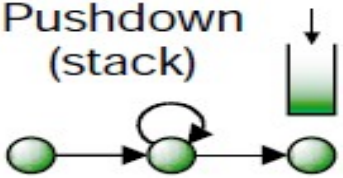
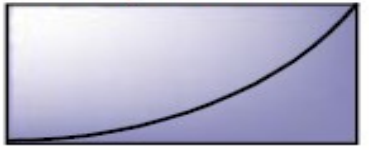
TEOREMA 3.21

Uma linguagem é Turing-reconhecível se e somente se algum enumerador a enumera.

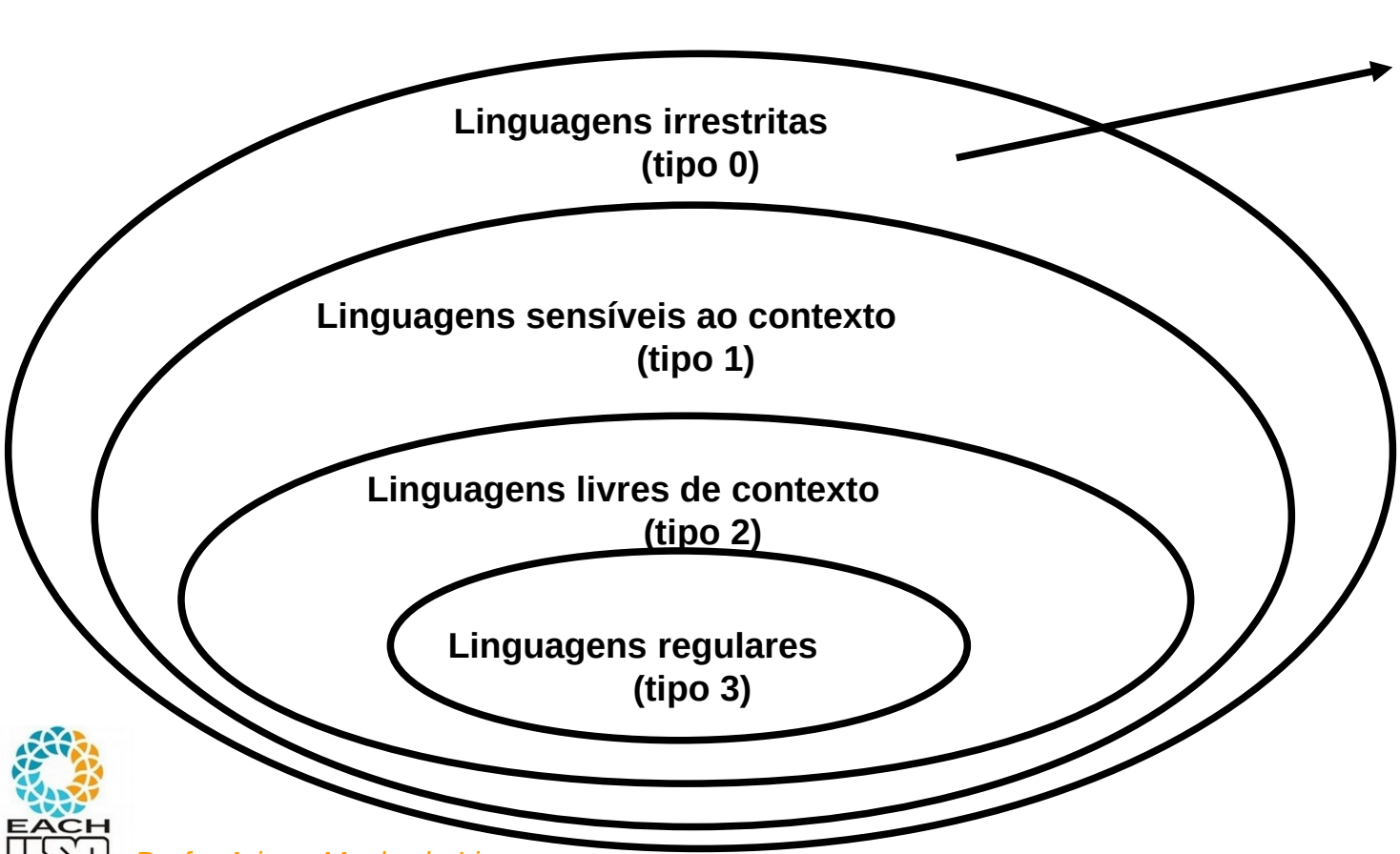
Por isso linguagens Turing-**reconhecíveis** são também chamadas linguagens recursivamente enumeráveis

Enumerabilidade tem a ver com identificar quais problemas são Turing-**reconhecíveis**

Linguagens, modelos computacionais (dispositivos, gramáticas) e suas complexidades

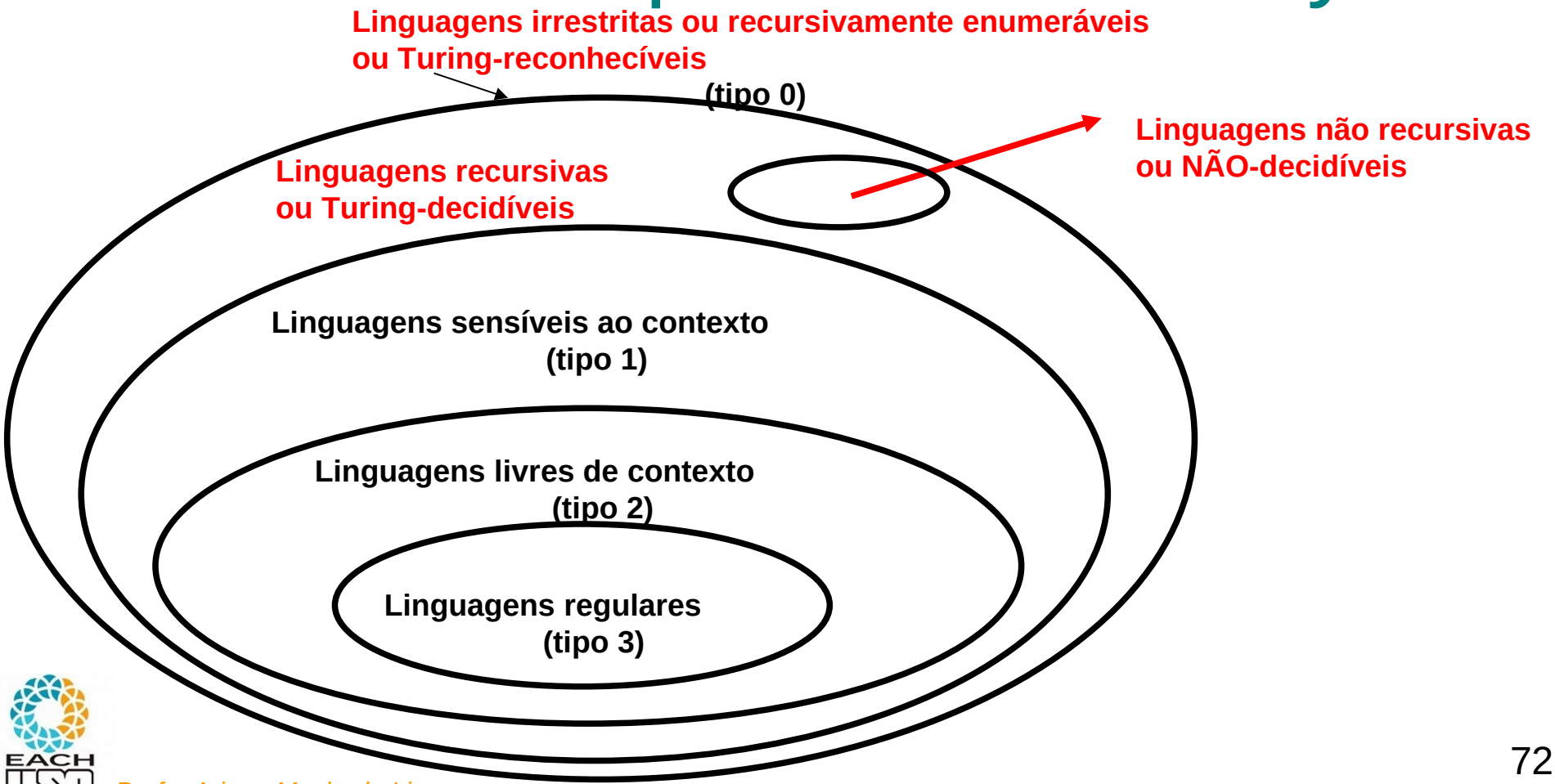
Recursively enumerable languages	Turing machine 	Unrestricted $Baa \rightarrow A$	Undecidable 
Context-sensitive languages	Linear bounded 	Context sensitive $At \rightarrow aA$	Exponential? 
Context-free languages	Pushdown (stack) 	Context free $S \rightarrow gSc$	Polynomial 
Regular languages	Finite-state automaton 	Regular $A \rightarrow cA$	Linear 

Hierarquia de Chomsky

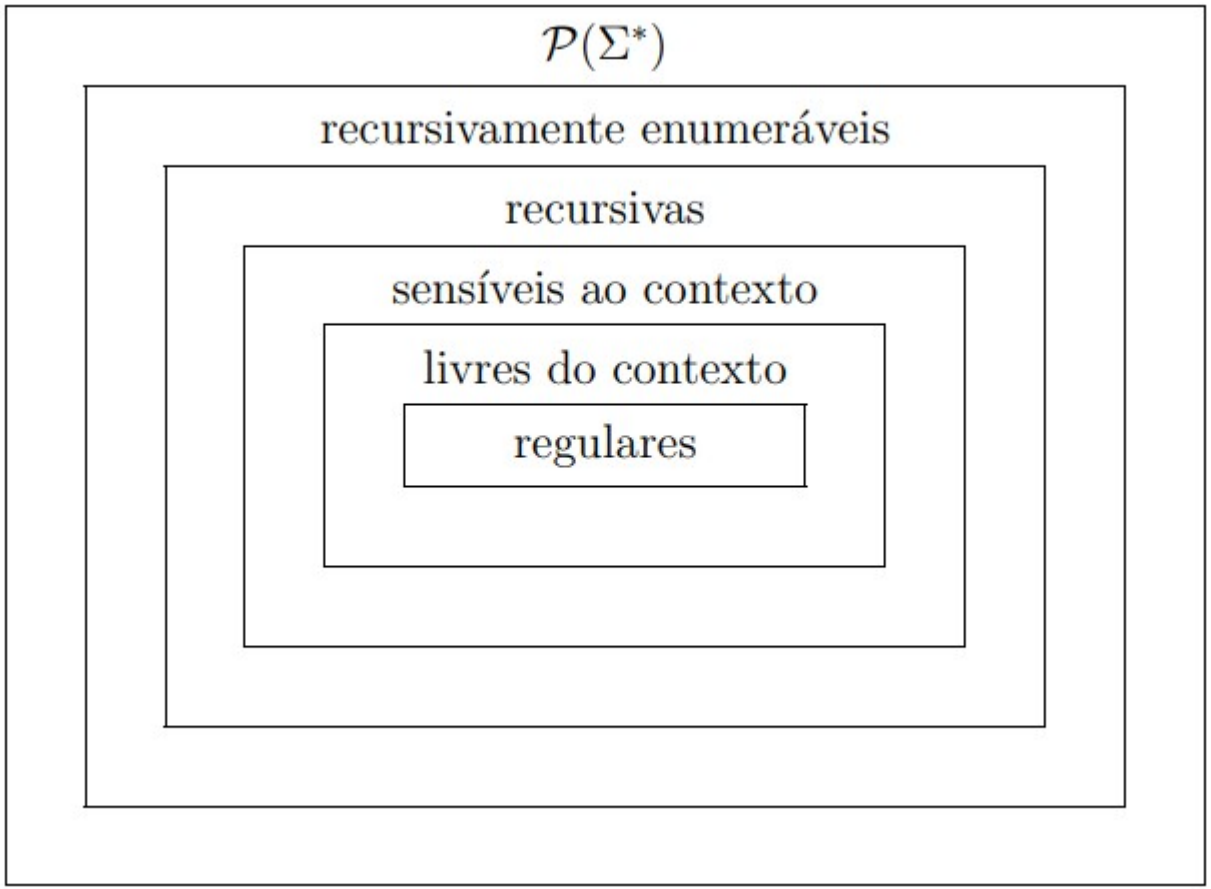


Linguagens recursivas
(Turing-decidíveis)
e não recursivas
(as demais Turing-reconhecíveis)

Hierarquia de Chomsky

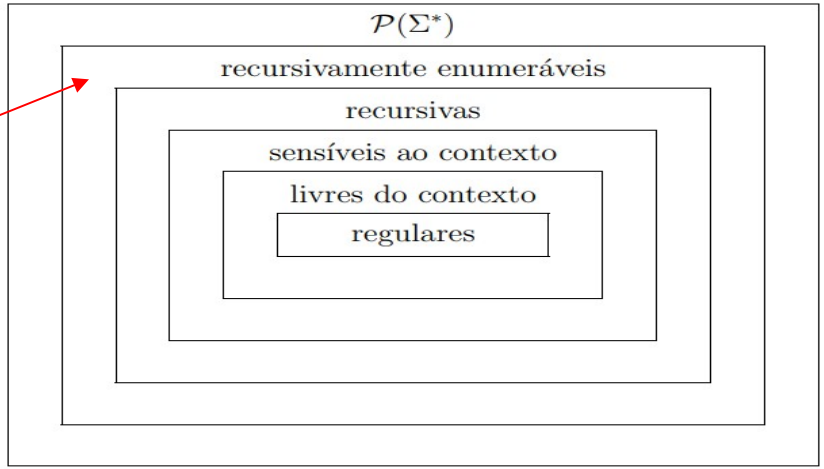


Mais precisamente...



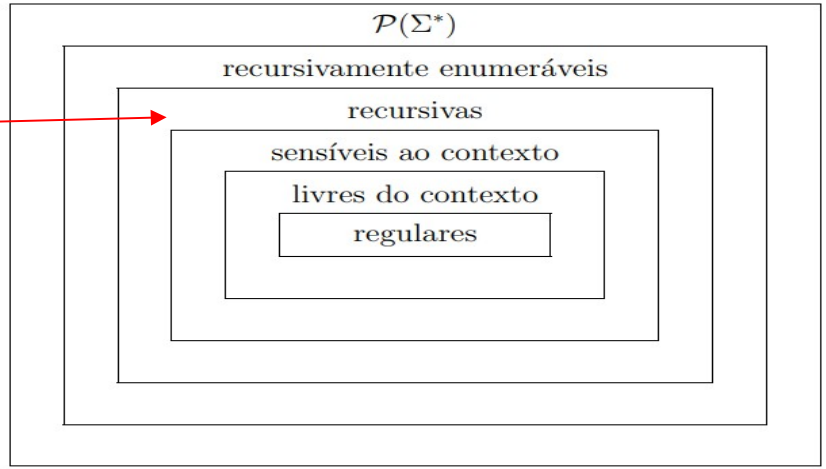
(VIERA, 2006)

Linguagens irrestritas ou recursivamente enumeráveis ou Turing-reconhecíveis



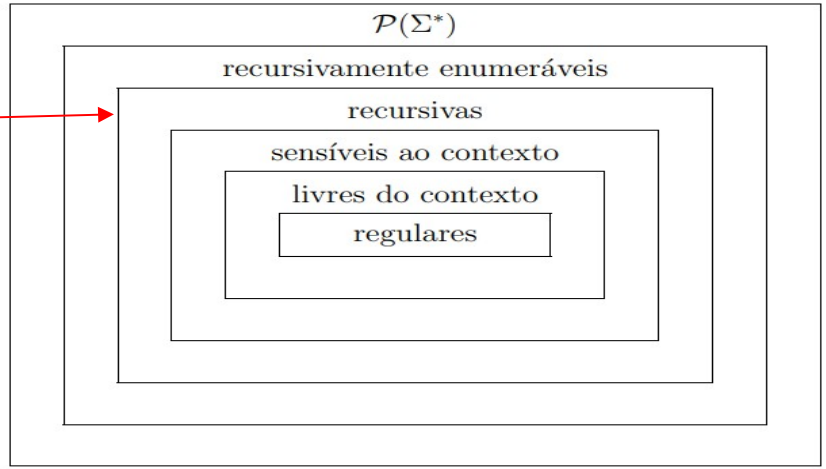
- Uma linguagem L é chamada **irrestrita** ou **recursivamente enumerável** ou **Turing-reconhecível** se ela for **reconhecida** por pelo menos uma Máquina de Turing M, ou seja:
  - Para toda cadeia w Є L, M pára e aceita w
  - Para toda cadeia z Є Σ*-L, M pára e rejeita z ou executa uma sequência infinita de movimentações

Linguagens recursivas ou Turing-decidíveis



- Uma linguagem L é chamada **recursiva** ou **Turing-decidível** se ela for **decidida** por pelo menos uma Máquina de Turing M, ou seja:
  - Para toda cadeia w Є L, M pára e aceita w
  - Para toda cadeia z Є Σ*-L, M pára e rejeita z

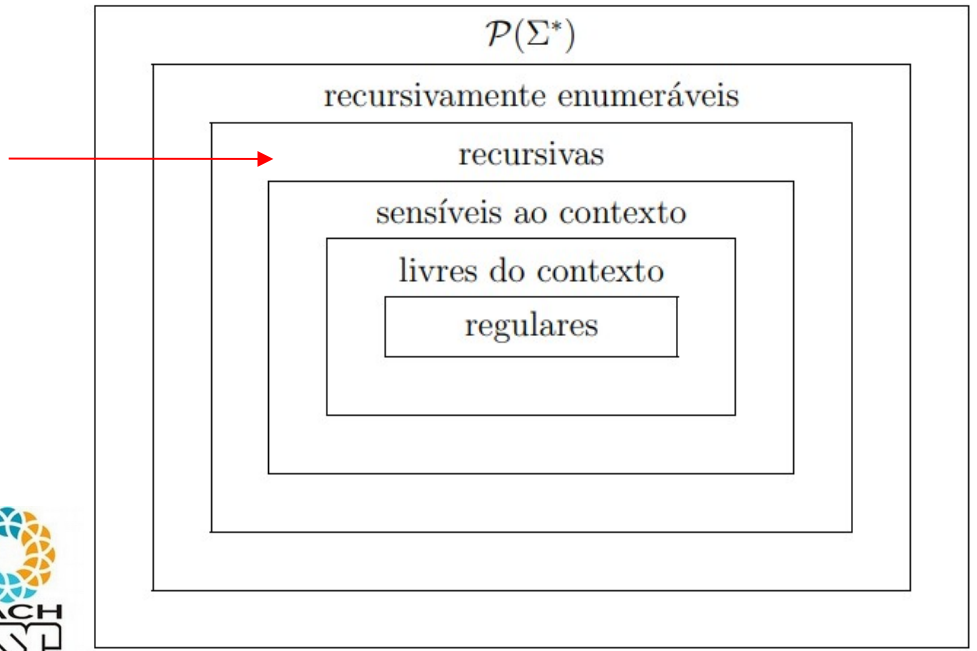
Linguagens não-recursivas



- Uma linguagem L é chamada **não-recursiva** se ela for **reconhecida** por pelo menos uma Máquina de Turing M **mas não decidida** por nenhuma Máquina de Turing, ou seja:
  - Para toda cadeia w Є L, M pára e aceita w
  - Para **pelo menos uma** cadeia z Є Σ*-L, M executa uma sequência infinita de movimentações (podendo parar e rejeitar as demais cadeias y Є Σ*-L)

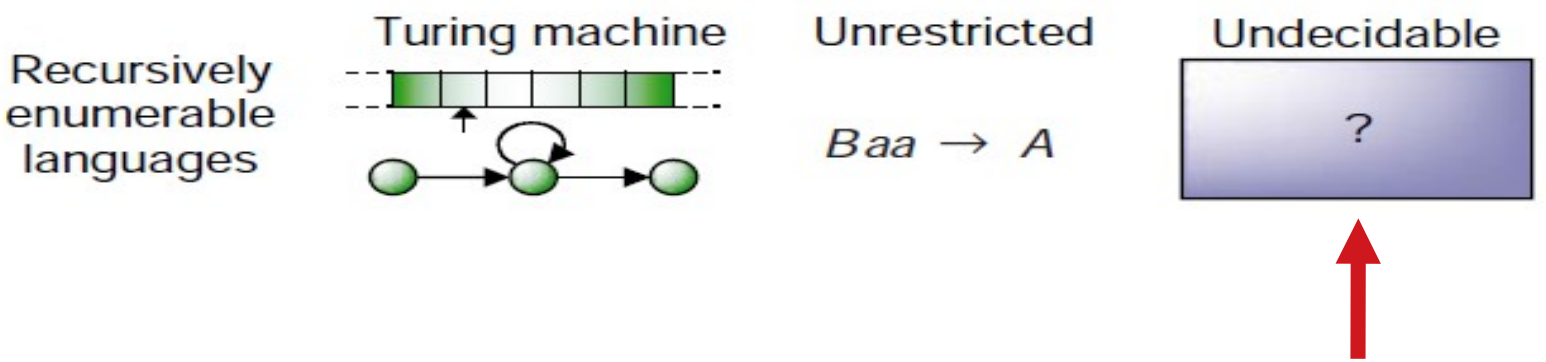
Linguagens recursivas

- Toda linguagem sensível ao contexto é recursiva
- Toda linguagem recursiva que NÃO é sensível ao contexto é também chamada estritamente recursiva



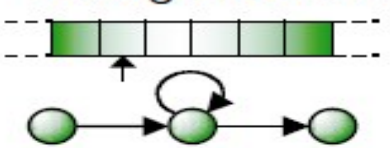
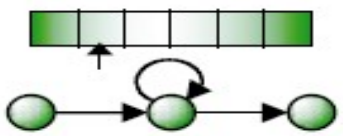
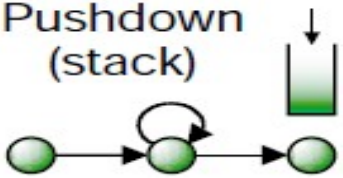
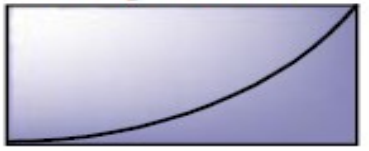
- Não há uma classe de gramáticas equivalente

Linguagens, dispositivos, gramáticas e complexidades



Logo, NO CASO GERAL, linguagens recursivamente enumeráveis possuem reconhecimento **INDECIDÍVEL**!

Linguagens, dispositivos, gramáticas e complexidades

Recursively enumerable languages	Turing machine 	Unrestricted $Baa \rightarrow A$	Undecidable 
Context-sensitive languages	Linear bounded 	Context sensitive $At \rightarrow aA$	Exponential? 
Context-free languages	Pushdown (stack) 	Context free $S \rightarrow gSc$	Polynomial 
Regular languages	Finite-state automaton 	Regular $A \rightarrow cA$	Linear 

Exercícios

- Já podem fazer os exercícios 3.1 a 3.8, e problemas 3.15 e 3.16 (dica: use a descrição de alto-nível de Mts para esses dois problemas).

Lista MÍNIMA cap 3

- 3.1a, 3.1d, 3.2d, 3.2e, 3.6, 3.8, 3.15, 3.16

Referências

SIPSER, M. Introdução à Teoria da Computação. Ed. Thomson. Cap 3.1 e 3.2

VIEIRA, N. J. Introdução aos fundamentos da computação: linguagens e máquinas. 2006