MAC 414
Autômatos, Computabilidade e Complexidade
aula 14 — 9/11/2020

O que é Algoritmo?

O que é Algoritmo?

Sequência de instruções que, para cada entrada válida para em um número finito de passos dando uma resposta correspondente à entrada.

O que é Algoritmo?

Sequência de instruções que, para cada entrada válida para em um número finito de passos dando uma resposta correspondente à entrada. Computa uma função.

O que é Algoritmo?

Sequência de instruções que, para cada entrada válida para em um número finito de passos dando uma resposta correspondente à entrada. Computa uma função.

Pode ser escrito em linguagem de programação, como MT, como função recursiva primitiva ou μ-recursiva, outros formalismos.

O que é Algoritmo?

Sequência de instruções que, para cada entrada válida para em um número finito de passos dando uma resposta correspondente à entrada. Computa uma função.

Pode ser escrito em linguagem de programação, como MT, como função recursiva primitiva ou μ-recursiva, outros formalismos.

Tudo simulável por MTs.

Tese de Church-Turing

Algoritmo é o mesmo que uma Máquina de Turing que para sempre.

Tese de Church-Turing

Algoritmo é o mesmo que uma Máquina de Turing que para sempre.

Tese de Church-Turing

Algoritmo é o mesmo que uma Máquina de Turing que para sempre.

Em particular uma função é computável sse for Turing-computável.

Tese de Church-Turing

Algoritmo é o mesmo que uma Máquina de Turing que para sempre.

Em particular uma função é computável sse for Turing-computável.

Tese de Church-Turing

Algoritmo é o mesmo que uma Máquina de Turing que para sempre.

Em particular uma função é computável sse for Turing-computável.

Noção informal $\longrightarrow$ definição formal.

Tese de Church-Turing

Algoritmo é o mesmo que uma Máquina de Turing que para sempre.

Em particular uma função é computável sse for Turing-computável.

Noção informal $\longrightarrow$ definição formal.
Aceita *porque* todas as tentativas de produzir novas definições recaíram nessa.

Tese de Church-Turing

Algoritmo é o mesmo que uma Máquina de Turing que para sempre.

Em particular uma função é computável sse for Turing-computável.

Noção informal $\longrightarrow$ definição formal.
Aceita *porque* todas as tentativas de produzir novas definições recaíram nessa.
Dá um *caminho* para mostrar que uma dada função *não é computável*.

MT universal

MT universal

Simulador genérico de MTs.

MT universal

Simulador genérico de MTs.

Pode ser pensado como um computador que executa código "escrito em MT".

MT universal

Simulador genérico de MTs.

Pode ser pensado como um computador que executa código "escrito em MT".

Entrada: descrição de uma MT $\mathscr{M}$ e uma entrada x para ela.

MT universal

Simulador genérico de MTs.

Pode ser pensado como um computador que executa código "escrito em MT".

Entrada: descrição de uma MT $\mathscr{M}$ e uma entrada x para ela.

Execução: computa $\mathscr{M}(x)$ (se estiver definido).

MT universal

Simulador genérico de MTs.

Pode ser pensado como um computador que executa código “escrito em MT”.

Entrada: descrição de uma MT $\mathscr{M}$ e uma entrada x para ela.

Execução: computa $\mathscr{M}(x)$ (se estiver definido).

Não pode haver restrição no número de estados de $\mathscr{M}$ ou no tamanho do seu alfabeto.

Convenções

Convenções

Os estados de $\mathscr{M}$ serão numerados como $q_0, q_1, \ldots, q_n$, com $s = q_0$.

Convenções

Os estados de $\mathscr{M}$ serão numerados como $q_0, q_1, \ldots, q_n$, com $s = q_0$.

Seja k um inteiro tal que exista uma representação binária de n em k bits. Vamos codificar o estado q_j por um q seguido da representação de j em k bits. Essa palavra será denotada como "q_j".

Convenções

Os estados de $\mathscr{M}$ serão numerados como $q_0, q_1, \ldots, q_n$, com $s = q_0$.

Seja k um inteiro tal que exista uma representação binária de n em k bits. Vamos codificar o estado q_j por um q seguido da representação de j em k bits. Essa palavra será denotada como "q_j".

Ex: Se $k = 4$, "q_5"= $q0101$.

Convenções

Convenções

Análogo para letras:

Convenções

Análogo para letras:
$\hat{\Sigma} = \Sigma \cup \{\sqcup, \triangleright, \leftarrow, \rightarrow\} = \{\sigma_0, \ldots, \sigma_m\}$,onde $\sqcup, \triangleright, \leftarrow, \rightarrow$ recebem os números de 0 a 3.

Convenções

Análogo para letras:

$\hat{\Sigma} = \Sigma \cup \{\sqcup, \triangleright, \leftarrow, \rightarrow\} = \{\sigma_0, \ldots, \sigma_m\}$,onde $\sqcup, \triangleright, \leftarrow, \rightarrow$ recebem os números de 0 a 3.

Seja ℓ um inteiro tal que exista uma representação binária de m em ℓ bits. Vamos codificar σ_j por um a seguido da representação de j em ℓ bits. Essa palavra será denotada como "a_j".

Convenções

Análogo para letras:
$\hat{\Sigma} = \Sigma \cup \{\sqcup, \triangleright, \leftarrow, \rightarrow\} = \{\sigma_0, \ldots, \sigma_m\}$,onde $\sqcup, \triangleright, \leftarrow, \rightarrow$ recebem os números de 0 a 3.

Seja ℓ um inteiro tal que exista uma representação binária de m em ℓ bits. Vamos codificar σ_j por um a seguido da representação de j em ℓ bits. Essa palavra será denotada como "a_j".

Ex: Se $\ell = 4$, "$\leftarrow$"$= a0010$.

Convenções

Análogo para letras:

$\hat{\Sigma} = \Sigma \cup \{\sqcup, \triangleright, \leftarrow, \rightarrow\} = \{\sigma_0, \ldots, \sigma_m\}$, onde $\sqcup, \triangleright, \leftarrow, \rightarrow$ recebem os números de 0 a 3.

Seja ℓ um inteiro tal que exista uma representação binária de m em ℓ bits. Vamos codificar σ_j por um a seguido da representação de j em ℓ bits. Essa palavra será denotada como "a_j".

Ex: Se $\ell = 4$, "$\leftarrow$"$=$ $a0010$.

Se $x \in \hat{\Sigma}^*$, "x" é a concatenação dos códigos das suas letras.

Convenções

Análogo para letras:
$\hat{\Sigma} = \Sigma \cup \{\sqcup, \triangleright, \leftarrow, \rightarrow\} = \{\sigma_0, \ldots, \sigma_m\}$,onde $\sqcup, \triangleright, \leftarrow, \rightarrow$ recebem os números de 0 a 3.

Seja ℓ um inteiro tal que exista uma representação binária de m em ℓ bits. Vamos codificar σ_j por um a seguido da representação de j em ℓ bits. Essa palavra será denotada como "a_j".

Ex: Se $\ell = 4$, "$\leftarrow$"$= a0010$.

Se $x \in \hat{\Sigma}^*$, "x" é a concatenação dos códigos das suas letras.

Se a única letra de verdade é a, ela tem número 4 e "$\triangleright aa \sqcup a$" $= a001a100a100a000a100$.

Convenções

Convenções

A codificação de $\mathscr{M}$, denotada "$\mathscr{M}$" é concatenação de todas as quádruplas (q, σ, p, τ) tais que $\delta(q, \sigma) = (p, \tau)$, onde

Convenções

A codificação de $\mathscr{M}$, denotada "$\mathscr{M}$" é concatenação de todas as quádruplas (q, σ, p, τ) tais que $\delta(q, \sigma) = (p, \tau)$, onde

$$\text{“}(q, \sigma, p, \tau)\text{''} = \#\text{“}q\text{''“}\sigma\text{''“}p\text{''“}\tau\text{''}.$$

Convenções

A codificação de $\mathscr{M}$, denotada "$\mathscr{M}$" é concatenação de todas as quádruplas (q, σ, p, τ) tais que $\delta(q, \sigma) = (p, \tau)$, onde

$$``(q, \sigma, p, \tau)" = \#``q"``\sigma"``p"``\tau".$$

Quádruplas em ordem lexicográfica (conveniência).

Convenções

A codificação de $\mathscr{M}$, denotada "$\mathscr{M}$" é concatenação de todas as quádruplas (q,σ,p,τ) tais que $\delta(q,\sigma)=(p,\tau)$, onde

"(q,σ,p,τ)" = #"q""σ""p""τ".

Quádruplas em ordem lexicográfica (conveniência).

Note que não é preciso dizer quem é s, já que por convenção ele é $q00\cdots0$.

Convenções

A codificação de $\mathscr{M}$, denotada "$\mathscr{M}$" é concatenação de todas as quádruplas (q, σ, p, τ) tais que $\delta(q, \sigma) = (p, \tau)$, onde

"(q, σ, p, τ)" = #"q""σ""p""τ".

Quádruplas em ordem lexicográfica (conveniência).

Note que não é preciso dizer quem é s, já que por convenção ele é $q00\cdots0$.

Nem H, já que esses são os estados que não aparecem como primeira componente de nenhuma quádrupla

Example 5.2.1: Consider the Turing machine $M = (K, \Sigma, \delta, s, \{h\})$, where $K = \{s, q, h\}$, $\Sigma = \{\sqcup, \triangleright, a\}$, and δ is given in this table.

state,	symbol	δ
s	a	$(q, \sqcup)$
s	$\sqcup$	$(h, \sqcup)$
s	$\triangleright$	$(s, \rightarrow)$
q	a	(s, a)
q	$\sqcup$	$(s, \rightarrow)$
q	$\triangleright$	$(q, \rightarrow)$

$(s, a, q, \sqcup)$

state/symbol	representation
s	$q00$
q	$q01$
h	$q11$
$\sqcup$	$a000$
$\triangleright$	$a001$
$\leftarrow$	$a010$
$\rightarrow$	$a011$
a	$a100$

Example 5.2.1: Consider the Turing machine $M = (K, \Sigma, \delta, s, \{h\})$, where $K = \{s, q, h\}$, $\Sigma = \{\sqcup, \triangleright, a\}$, and δ is given in this table.

state,	symbol	δ
s	a	$(q, \sqcup)$
s	$\sqcup$	$(h, \sqcup)$
s	$\triangleright$	$(s, \rightarrow)$
q	a	(s, a)
q	$\sqcup$	$(s, \rightarrow)$
q	$\triangleright$	$(q, \rightarrow)$

state/symbol	representation
s	$q00$
q	$q01$
h	$q11$
$\sqcup$	$a000$
$\triangleright$	$a001$
$\leftarrow$	$a010$
$\rightarrow$	$a011$
a	$a100$

$$
\text{“}\mathscr{M}\text{”} = \#q00a100q01a000\#q00a000q11a000\\
\#q00a001q00a011\#q01a100q00a011\\
\#q01a000q00a011\#q01a001q01a011.
$$

q $\sqcup$ s $\rightarrow$

A máquina universal $\mathscr{U}$

A máquina universal $\mathscr{U}$

Descrição funcional:

Se $\mathscr{M}$ é uma MT e w é uma entrada válida para $\mathscr{M}$, então:

$$\mathscr{U}(\text{“}\mathscr{M}\text{”}\text{“}w\text{”}) = \text{“}\underline{\mathscr{M}(w)}\text{”}.$$

"Erro de sintaxe"

A máquina universal $\mathscr{U}$

Descrição funcional:

Se $\mathscr{M}$ é uma MT e w é uma entrada válida para $\mathscr{M}$, então:

$$\mathscr{U}(\text{“}\mathscr{M}\text{”}\text{“}w\text{”}) = \text{“}\mathscr{M}(w)\text{”}.$$

Em particular, se $(\mathscr{M}, w)$ não para, $\mathscr{U}$ com a entrada acima também não.

A máquina universal $\mathscr{U}$

Descrição funcional:

Se $\mathscr{M}$ é uma MT e w é uma entrada válida para $\mathscr{M}$, então:

$$\mathscr{U}(\text{“}\mathscr{M}\text{”“}w\text{”}) = \text{“}\mathscr{M}(w)\text{”}.$$

Em particular, se $(\mathscr{M}, w)$ não para, $\mathscr{U}$ com a entrada acima também não.

Vamos descrever $\mathscr{U}$ como MT de três fitas. A $\mathscr{U}$ de verdade é a MT que simula essa daí.

Inicialização

1. Entrada: $\triangleright \sqcup$"$\mathscr{M}$""w"

Inicialização

1. Entrada: $\triangleright\sqcup$"$\mathscr{M}$""w" fita 1
2. Copie "$\mathscr{M}$" para a fita 2, apagando da 1, junto com $\triangleright\sqcup$. Cursor 2 para a esquerda.

Inicialização

1. Entrada: $\triangleright\sqcup$"$\mathscr{M}$""w"
2. Copie "$\mathscr{M}$" para a fita 2, apagando da 1, junto com $\triangleright\sqcup$. Cursor 2 para a esquerda.
3. Escreva "s" na fita 3. Cursor 3 para a esquerda.

Inicialização

q000..0

1. Entrada: $\triangleright\sqcup$"$\mathscr{M}$""w"
2. Copie "$\mathscr{M}$" para a fita 2, apagando da 1, junto com $\triangleright\sqcup$. Cursor 2 para a esquerda.
3. Escreva "s" na fita 3. Cursor 3 para a esquerda.
4. Escreva "$\triangleright\sqcup$" à esquerda da fita 1, e desloque "w" para encostar. Fita 1 contem "$\triangleright\sqcup w$". Cursor 1 no início do "$\sqcup$" mais esquerda.

Operação

Operação

Simulação de um passo de $\mathscr{M}$.

Invariantes:

1. Fita 1 contém a codificação da fita de $\mathscr{M}$, com cursor na posição correspondente.

Operação

Simulação de um passo de $\mathscr{M}$.

Invariantes:

1. Fita 1 contém a codificação da fita de $\mathscr{M}$, com cursor na posição correspondente.
2. Fita 2 contem "$\mathscr{M}$", cursor à esquerda.

Operação

Simulação de um passo de $\mathscr{M}$.

Invariantes:

1. Fita 1 contém a codificação da fita de $\mathscr{M}$, com cursor na posição correspondente.
2. Fita 2 contem "$\mathscr{M}$", cursor à esquerda.
3. Fita 3 contém a codificação do estado atual, cursor à esquerda.

O passo:

O passo:

Seja x o conteúdo da fita 3 e y a codificação de caractere começando pelo cursor da fita 1.

O passo:

a0011

a1000

a1000

Seja x o conteúdo da fita 3 e y a codificação de caractere começando pelo cursor da fita 1.

Procura na fita 2 o texto $\#xy$.

- Se achou, mude o estado registrado na fita 3, conforme a 3ª componente da quádrupla, e opere a fita 1 conforme a 4ª. Volte cursores 2 e 3 para a esquerda.

a1000

O passo:

Seja x o conteúdo da fita 3 e y a codificação de caractere começando pelo cursor da fita 1.

Procura na fita 2 o texto $\#xy$.

- Se achou, mude o estado registrado na fita 3, conforme a 3ª componente da quádrupla, e opere a fita 1 conforme a 4ª. Volte cursores 2 e 3 para a esquerda.
- Se não achou, pare. A fita 3 indica um estado de parada de $\mathscr{M}$.

O passo:

Seja x o conteúdo da fita 3 e y a codificação de caractere começando pelo cursor da fita 1.

Procura na fita 2 o texto $\#xy$.

- Se achou, mude o estado registrado na fita 3, conforme a 3ª componente da quádrupla, e opere a fita 1 conforme a 4ª. Volte cursores 2 e 3 para a esquerda.
- Se não achou, pare. A fita 3 indica um estado de parada de $\mathscr{M}$.

O passo:

Seja x o conteúdo da fita 3 e y a codificação de caractere começando pelo cursor da fita 1.

Procura na fita 2 o texto $\#xy$.

- Se achou, mude o estado registrado na fita 3, conforme a 3ª componente da quádrupla, e opere a fita 1 conforme a 4ª. Volte cursores 2 e 3 para a esquerda.
- Se não achou, pare. A fita 3 indica um estado de parada de $\mathscr{M}$.

O invariante claramente se mantem. Assim, $\mathscr{U}$ para com "x" na fita 1 sse $(\mathscr{M}, w)$ para com x na fita.

Relembrando

$L \subseteq \Sigma^*$ é:

Relembrando

$L \subseteq \Sigma^*$ é:

Recursiva: se existe uma MT $\mathscr{M}$ que para sempre, computando 0 ou 1, e tal que para todo x, $\mathscr{M}(x) = 1$ sse $x \in L$.

Relembrando

$L \subseteq \Sigma^*$ é:

- Recursiva: se existe uma MT $\mathscr{M}$ que para sempre, computando 0 ou 1, e tal que para todo x, $\mathscr{M}(x) = 1$ sse $x \in L$.
- Recursivamente enumerável: se existe uma MT $\mathscr{M}$ tal que $L = \{x \mid (\mathscr{M}, x) \text{ para}\}$.

Relembrando

$L \subseteq \Sigma^*$ é:

- Recursiva: se existe uma MT $\mathscr{M}$ que para sempre, computando 0 ou 1, e tal que para todo x, $\mathscr{M}(x) = 1$ sse $x \in L$.
- Recursivamente enumerável: se existe uma MT $\mathscr{M}$ tal que $L = \{x \mid (\mathscr{M}, x) \text{ para}\}$.

Relembrando

$L \subseteq \Sigma^*$ é:

Recursiva: se existe uma MT $\mathscr{M}$ que para sempre, computando 0 ou 1, e tal que para todo x, $\mathscr{M}(x) = 1$ sse $x \in L$.

Recursivamente enumerável: se existe uma MT $\mathscr{M}$ tal que $L = \{x \mid (\mathscr{M}, x) \text{ para}\}$.

Fato: toda linguagem recursiva é recursivamente enumerável.

Timeout

Timeout

A linguagem $H = \{``\mathscr{M}"``w" \mid \mathscr{M}$ é MT, $(\mathscr{M}, w)$ para$\}$ é semi-decidida por $\mathscr{U}$, logo é recursivamente enumerável. É recursiva?

Timeout

A linguagem $H = \{``\mathscr{M}"``w" \mid \mathscr{M}$ é MT, $(\mathscr{M}, w)$ para$\}$ é semi-decidida por $\mathscr{U}$, logo é recursivamente enumerável. É recursiva?

Problema: $\mathscr{M}$ pode ficar rodando indefinidamente.

Timeout

A linguagem $H = \{``\mathscr{M}"``w" \mid \mathscr{M}$ é MT, $(\mathscr{M}, w)$ para$\}$ é semi-decidida por $\mathscr{U}$, logo é recursivamente enumerável. É recursiva?

Problema: $\mathscr{M}$ pode ficar rodando indefinidamente.

Truque padrão: parar o programa se passar do tempo.

Timeout

A linguagem $H = \{``\mathscr{M}"``w" \mid \mathscr{M}$ é MT, $(\mathscr{M}, w)$ para$\}$ é semi-decidida por $\mathscr{U}$, logo é recursivamente enumerável. É recursiva?

Problema: $\mathscr{M}$ pode ficar rodando indefinidamente.

Truque padrão: parar o programa se passar do tempo.

Prop: *A linguagem* $\{``\mathscr{M}"``w", (n)_2 \mid \mathscr{M}$ *é MT,* $n \in \mathbb{N}, (\mathscr{M}, w)$ *para em até n passos*$\}$ *é recursiva.*

Timeout

A linguagem $H = \{``\mathscr{M}"``w" \mid \mathscr{M} \text{ é MT}, (\mathscr{M}, w) \text{ para}\}$ é semi-decidida por $\mathscr{U}$, logo é recursivamente enumerável. É recursiva?

Problema: $\mathscr{M}$ pode ficar rodando indefinidamente.

Truque padrão: parar o programa se passar do tempo.

Prop: *A linguagem* $\{``\mathscr{M}"``w", (n)_2 \mid \mathscr{M} \textit{ é MT}, n \in \mathbb{N}, (\mathscr{M}, w) \textit{ para em até } n \textit{ passos}\}$ *é recursiva.*

Dem: Modifique $\mathscr{U}$ para contar passos e parar se passou o tempo.

O problema da parada

Desejo: um programa **Para** (Q, x) que, dado o código de um programa Q e uma entrada x para ele, decide se Q pára dada a entrada x, devolvendo 0 ou 1.

O problema da parada

Desejo: um programa **Para** (Q, x) que, dado o código de um programa Q e uma entrada x para ele, decide se Q para dada a entrada x, devolvendo 0 ou 1.

Se existisse, existiria este programa:

```
diagonal (Q)
    while Para (Q,Q);
```

O problema da parada

Desejo: um programa **Para** (Q, x) que, dado o código de um programa Q e uma entrada x para ele, decide se Q para dada a entrada x, devolvendo 0 ou 1.

Se existisse, existiria este programa:

diagonal (Q)

while **Para** (Q,Q);

diagonal (**diagonal**) para?

1 Para(d, d) = 1

0 Para(d, d) = 0

O problema da parada

Formalizando: vamos provar que

$H = \{``\mathscr{M}"``w" \mid \mathscr{M}$ é MT, $(\mathscr{M}, w)$ para$\}$

não é recursiva.

O problema da parada

Formalizando: vamos provar que

$H = \{``\mathscr{M}"``w" \mid \mathscr{M} \text{ é MT}, (\mathscr{M}, w) \text{ para}\}$

não é recursiva.

Se fosse, então

$D = \{``\mathscr{M}" \mid (\mathscr{M}, ``\mathscr{M}") \text{ para}\}$

também seria.

$\mathcal{L}(M) = D$

$N: D\sqcup x \to D\sqcup x, x$ $\in H!$

O problema da parada

Formalizando: vamos provar que

$H = \{``\mathscr{M}"``w" \mid \mathscr{M} \text{ é MT}, (\mathscr{M}, w) \text{ para}\}$

não é recursiva.

Se fosse, então

$D = \{``\mathscr{M}" \mid (\mathscr{M}, ``\mathscr{M}") \text{ para}\}$

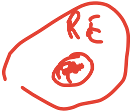
também seria.

M

Então, o complemento $\overline{D}$ também seria:

$\hat{M}$

$$\overline{D} = \{x \mid x \text{ não codifica MT}, \\ \text{ou } x = ``\mathscr{M}" \text{ e } (\mathscr{M}, x) \text{ não para}\}$$



O problema da parada

Formalizando: vamos provar que

$H = \{\text{“}\mathscr{M}\text{”“}w\text{”} \mid \mathscr{M} \text{ é MT}, (\mathscr{M}, w) \text{ para}\}$

não é recursiva.

$\{x \mid (N, x) \text{ para}\}$

Se fosse, então

$D = \{\text{“}\mathscr{M}\text{”} \mid (\mathscr{M}, \text{“}\mathscr{M}\text{”}) \text{ para}\}$

também seria.

$L(N) = \overline{D}$

$(N, \text{“}N\text{”})$?

Então, o complemento $\overline{D}$ também seria:

$\overline{D} = \{x \mid x \text{ não codifica MT},$

$\text{ou } x = \text{“}\mathscr{M}\text{”} \text{ e } (\mathscr{M}, x) \text{ não para}\}$

$\text{“}N\text{”} \in \overline{D}$.

$\notin \overline{D}$

Mas $\overline{D}$ nem recursivamente enumerável é!

O problema da parada

Teorema

O problema da parada é indecidível.

O problema da parada

Teorema

O problema da parada é indecidível.

Teorema

A família das linguagens recursivas é subconjunto próprio das recursivamente enumeráveis.

O problema da parada

Teorema

O problema da parada é indecidível.

Teorema

A família das linguagens recursivas é subconjunto próprio das recursivamente enumeráveis.

Teorema

O conjunto das linguagens recursivamente enumeráveis não é fechado por complemento.

Enumeração

Enumeração

Uma linguagem L é Turing-enumerável se existe uma MT M com um estado especial print tal que $L = \{ w \mid (s, \rhd\underline{\sqcup}) \vdash_M^* (\text{print}, \rhd\underline{\sqcup}w) \}$.

Teorema

Uma linguagem é Turing-enumerável sse ela for recursivamente enumerável.

Dem: Blá!

Enumeração

Uma linguagem L é Turing-enumerável se existe uma MT M com um estado especial print tal que $L = \{w \mid (s, \triangleright\underline{\sqcup}) \vdash_M^* (\text{print}, \triangleright\underline{\sqcup}w)\}$.

Teorema

Uma linguagem é Turing-enumerável sse ela for recursivamente enumerável.

Enumeração

Uma linguagem L é Turing-enumerável se existe uma MT M com um estado especial print tal que $L = \{w \mid (s, \triangleright\underline{\sqcup}) \vdash_M^* (\text{print}, \triangleright\underline{\sqcup}w)\}$.

Teorema

Uma linguagem é Turing-enumerável sse ela for recursivamente enumerável.

Uma linguagem L é lexicograficamente Turing-enumerável se existe uma MT que a enumera em ordem lexicográfica.

Teorema

Uma linguagem L é lexicograficamente Turing-enumerável sse for recursiva.

Se L é R.E., então é T.E.

M tq $L = \{x \mid (M, x) \text{ para}\}$

M' $L = \{x \mid \exists n \mid (M, x, n) \text{ para em} \leq n \text{ passos}\}$

gerar sistematicamente pares (x, n)