MAC 414
Autômatos, Computabilidade e Complexidade
aula 15 — 11/11/2020

Recursiva $\times$ R.E.

Recursiva $\times$ R.E.

Teorema: Existem linguagens R.E. que não são recursivas.

Recursiva × R.E.

Teorema: Existem linguagens R.E. que não são recursivas.

Teorema

Uma linguagem é recursiva sse ela e seu complemento são recursivamente enumeráveis.

[illegible]: L rec ⇔ L e $\bar{L}$ são R.E.

⇐ dado x
Roda M e M' em paralelo

M s.d. L
M' s.d $\bar{L}$

O problema da parada

Teorema

O problema da parada é indecidível.

O problema da parada

Teorema

O problema da parada é indecidível.

O problema da parada

Teorema

O problema da parada é indecidível.

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

Indecidível

Indecidível

Um problema é indecidível se
não existe um algoritmo para ele.

Indecidível

Um problema é indecidível se
não existe um algoritmo para ele.

Problema de decisão envolve um conjunto de instâncias, onde para cada uma a resposta é *sim* ou *não*.

Indecidível

Um problema é indecidível se
não existe um algoritmo para ele.

Problema de decisão envolve um conjunto de instâncias, onde para cada uma a resposta é *sim* ou *não*.

Uma solução para o problema é um algoritmo que computa a resposta dada a instância.

Indecidível

Um problema é indecidível se
não existe um algoritmo para ele.

Problema de decisão envolve um conjunto de instâncias, onde para cada uma a resposta é *sim* ou *não*.

Uma solução para o problema é um algoritmo que computa a resposta dada a instância.

Formalizando: é o problema de pertinência para a linguagem das instâncias com resposta *sim*.

Indecidível

Um problema é indecidível se
não existe um algoritmo para ele.

Problema de decisão envolve um conjunto de instâncias, onde para cada uma a resposta é *sim* ou *não*.

Uma solução para o problema é um algoritmo que computa a resposta dada a instância.

Formalizando: é o problema de pertinência para a linguagem das instâncias com resposta *sim*.

Mostrar que é decidível: apresentar algoritmo.

Indecidível

Um problema é indecidível se
não existe um algoritmo para ele.

Problema de decisão envolve um conjunto de instâncias, onde para cada uma a resposta é *sim* ou *não*.

Uma solução para o problema é um algoritmo que computa a resposta dada a instância.

Formalizando: é o problema de pertinência para a linguagem das instâncias com resposta *sim*.

Mostrar que é decidível: apresentar algoritmo.

Mostrar que é indecidível:

Indecidível

Um problema é indecidível se
não existe um algoritmo para ele.

Problema de decisão envolve um conjunto de instâncias, onde para cada uma a resposta é *sim* ou *não*.

Uma solução para o problema é um algoritmo que computa a resposta dada a instância.

Formalizando: é o problema de pertinência para a linguagem das instâncias com resposta *sim*.

Mostrar que é decidível: apresentar algoritmo.

Mostrar que é indecidível: será que tem que diagonalizar sempre?

Redução

Redução

Def: Sejam $L_1, L_2 \subseteq \Sigma^*$. Uma redução de L_1 a L_2 é uma função recursiva $\tau : \Sigma^* \to \Sigma^*$ tal que para todo $x \in \Sigma^*$, $\tau(x) \in L_2$ sse $x \in L_1$.

Redução

Def: Sejam $L_1, L_2 \subseteq \Sigma^*$. Uma redução de L_1 a L_2 é uma função recursiva $\tau : \Sigma^* \to \Sigma^*$ tal que para todo $x \in \Sigma^*$, $\tau(x) \in L_2$ sse $x \in L_1$.

Proposição

Suponha que exista uma redução de L_1 a L_2. Então,

Redução

Def: Sejam $L_1, L_2 \subseteq \Sigma^*$. Uma redução de L_1 a L_2 é uma função recursiva $\tau : \Sigma^* \to \Sigma^*$ tal que para todo $x \in \Sigma^*$, $\tau(x) \in L_2$ sse $x \in L_1$.

Proposição

Suponha que exista uma redução de L_1 a L_2. Então,

Redução

Def: Sejam $L_1, L_2 \subseteq \Sigma^*$. Uma redução de L_1 a L_2 é uma função recursiva $\tau : \Sigma^* \to \Sigma^*$ tal que para todo $x \in \Sigma^*$, $\tau(x) \in L_2$ sse $x \in L_1$.

Proposição

Suponha que exista uma redução de L_1 a L_2. Então,

a) *Se L_2 é recursiva ou recursivamente enumerável, então L_1 também é.*

b) *Se L_1 é indecidível, então L_2 também é.*

Redução

Def: Sejam $L_1, L_2 \subseteq \Sigma^*$. Uma redução de L_1 a L_2 é uma função recursiva $\tau : \Sigma^* \to \Sigma^*$ tal que para todo $x \in \Sigma^*$, $\tau(x) \in L_2$ sse $x \in L_1$.

Proposição

Suponha que exista uma redução de L_1 a L_2. Então,

a) *Se L_2 é recursiva ou recursivamente enumerável, então L_1 também é.*

b) *Se L_1 é indecidível, então L_2 também é.*

Na prática: se descreve L recursiva contendo L_1 e se especifica τ só em L.

Alguns problemas indecidíveis

Dada uma MT $\mathscr{M}$:

1. E além disso, uma palavra w, $(\mathscr{M}, w)$ para? fl

Alguns problemas indecidíveis

Dada uma MT $\mathscr{M}$:

1. E além disso, uma palavra w, $(\mathscr{M}, w)$ para?
2. $(\mathscr{M}, \lambda)$ para?

Alguns problemas indecidíveis

Dada uma MT $\mathscr{M}$:

1. E além disso, uma palavra w, $(\mathscr{M}, w)$ para?
2. $(\mathscr{M}, \lambda)$ para?
3. Existe algum w tal que $(\mathscr{M}, w)$ para?

Alguns problemas indecidíveis

Dada uma MT $\mathscr{M}$:

1. E além disso, uma palavra w, $(\mathscr{M}, w)$ para?
2. $(\mathscr{M}, \lambda)$ para?
3. Existe algum w tal que $(\mathscr{M}, w)$ para?
4. $\mathscr{M}$ para para toda entrada?

Alguns problemas indecidíveis

Dada uma MT $\mathscr{M}$:

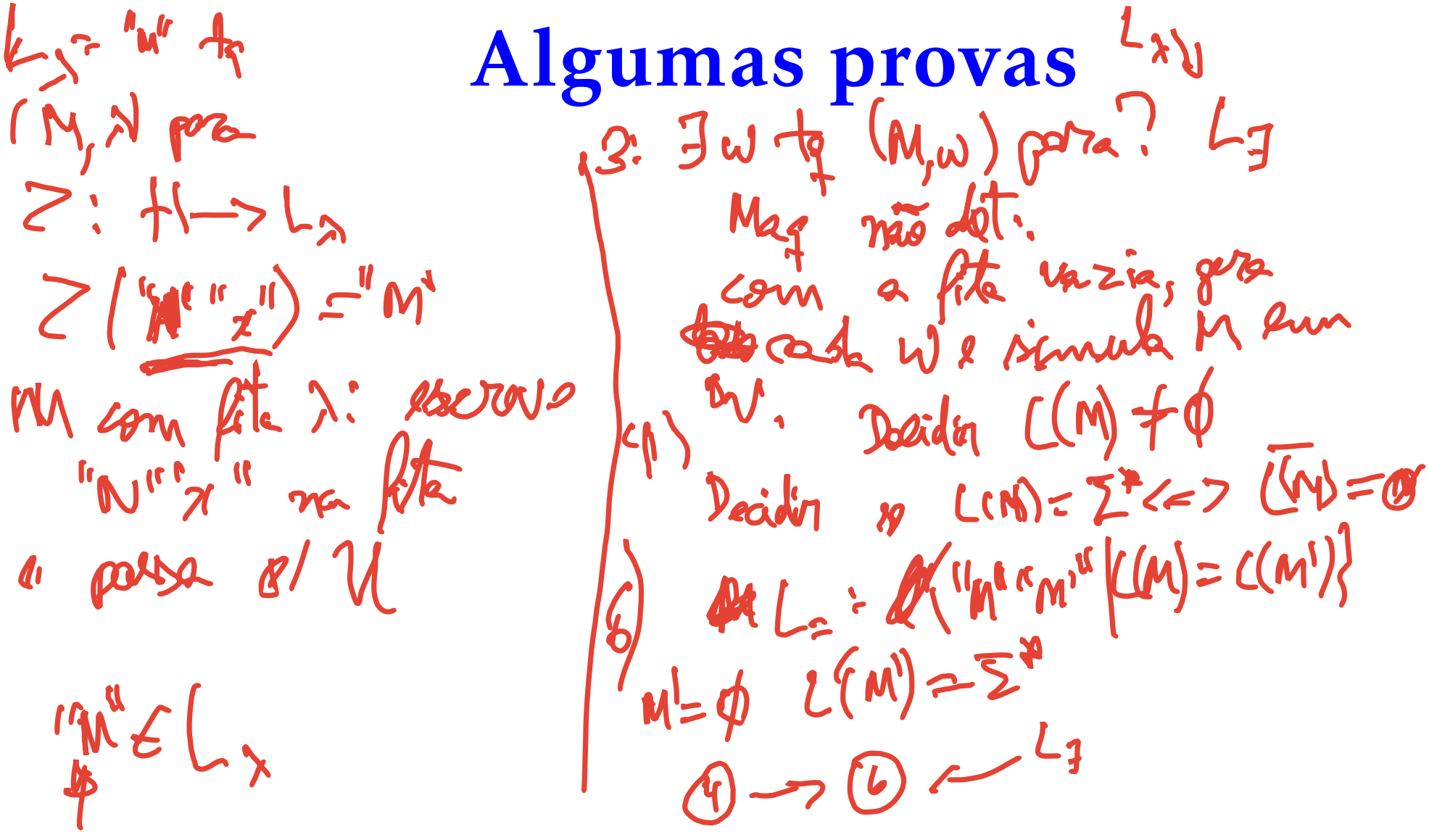
1. E além disso, uma palavra w, $(\mathscr{M}, w)$ para?
2. $(\mathscr{M}, \lambda)$ para?
3. Existe algum w tal que $(\mathscr{M}, w)$ para?
4. $\mathscr{M}$ para para toda entrada?
5. A lingugem semi-decidida por $\mathscr{M}$ é regular? Livre de contexto? Recursiva?

Alguns problemas indecidíveis

Dada uma MT $\mathscr{M}$:

1. E além disso, uma palavra w, $(\mathscr{M}, w)$ para?
2. $(\mathscr{M}, \lambda)$ para?
3. Existe algum w tal que $(\mathscr{M}, w)$ para?
4. $\mathscr{M}$ para para toda entrada?
5. A linguagem semi-decidida por $\mathscr{M}$ é regular? Livre de contexto? Recursiva?
6. Dada uma outra MT, $\mathscr{M}'$, elas param nas mesmas palavras?

Algumas provas



Outros problemas indecidíveis

Outros problemas indecidíveis

Dada uma gramática livre de contexto G, ela gera Σ^*?

Outros problemas indecidíveis

Dada uma gramática livre de contexto G, ela gera Σ^*?

O problema de correspondência de Post:
Dadas duas sequências $u_1, \ldots, u_n, v_1, \ldots, v_n$ de palavras em Σ^*, existe uma sequência $i_1, \ldots, i_m$ de índices (permitido repetições) tal que os produtos $u_{i_1} u_{i_2} \cdots u_{i_m}$ e $v_{i_1} v_{i_2} \cdots v_{i_m}$ são iguais?

Outros problemas indecidíveis

Dada uma gramática livre de contexto G, ela gera Σ^*?

O problema de correspondência de Post:
Dadas duas sequências $u_1, \ldots, u_n, v_1, \ldots, v_n$ de palavras em Σ^*, existe uma sequência $i_1, \ldots, i_m$ de índices (permitido repetições) tal que os produtos $u_{i_1} u_{i_2} \cdots u_{i_m}$ e $v_{i_1} v_{i_2} \cdots v_{i_m}$ são iguais?

O Jogo da Vida (Game of Life). JH Conway

Autômatos celulares

Outros problemas indecidíveis

Dada uma gramática livre de contexto G, ela gera Σ^*?

O problema de correspondência de Post:
Dadas duas sequências $u_1, \ldots, u_n, v_1, \ldots, v_n$ de palavras em Σ^*, existe uma sequência $i_1, \ldots, i_m$ de índices (permitido repetições) tal que os produtos $u_{i_1} u_{i_2} \cdots u_{i_m}$ e $v_{i_1} v_{i_2} \cdots v_{i_m}$ são iguais? Winning Ways II.2

O Jogo da Vida (Game of Life). Conway, Guy, Berlekamp
Dada uma configuração, é indecidível se ela vai desaparecer, se vai entrar em loop, etc.

Mais exemplos

Mais exemplos

Dadas matrizes 3×3 de inteiros, existe um produto delas (com repetições) que dá a matriz nula?

Mais exemplos

Dadas matrizes 3×3 de inteiros, existe um produto delas (com repetições) que dá a matriz nula?

Um k-hipergrafo consiste de um conjunto finito V de vértices e uma coleção de k-subconjuntos de V, as arestas.

Mais exemplos

Dadas matrizes 3×3 de inteiros, existe um produto delas (com repetições) que dá a matriz nula?

Um k-hipergrafo consiste de um conjunto finito V de vértices e uma coleção de k-subconjuntos de V, as arestas. A subdivisão em $S \subset V$ com $1 < |S| \leq k$ é:

1. Seja $\mathscr{V} = \{A \setminus \{p\} \mid \text{aresta } A \text{ contem } S\}$, $p \in S$.

Mais exemplos

Dadas matrizes 3×3 de inteiros, existe um produto delas (com repetições) que dá a matriz nula?

Um k-hipergrafo consiste de um conjunto finito V de vértices e uma coleção de k-subconjuntos de V, as arestas. A subdivisão em $S \subset V$ com $1 < |S| \leq k$ é:

1. Seja $\mathscr{V} = \{A \backslash \{p\} \mid \text{aresta } A \text{ contem } S\}$, $p \in S$.
2. Remova as arestas que contém S.

Mais exemplos

Dadas matrizes 3×3 de inteiros, existe um produto delas (com repetições) que dá a matriz nula?

Um k-hipergrafo consiste de um conjunto finito V de vértices e uma coleção de k-subconjuntos de V, as arestas. A subdivisão em $S \subset V$ com $1 < |S| \leq k$ é:

1. Seja $\mathscr{V} = \{A \backslash \{p\} \mid \text{aresta } A \text{ contem } S\}$, $p \in S$.
2. Remova as arestas que contém S.
3. Adicione um vértice novo, v e novas arestas: $T \cup \{v\}$, para todo $T \in \mathscr{V}$

Mais exemplos

Dadas matrizes 3×3 de inteiros, existe um produto delas (com repetições) que dá a matriz nula?

Um k-hipergrafo consiste de um conjunto finito V de vértices e uma coleção de k-subconjuntos de V, as arestas. A subdivisão em $S \subset V$ com $1 < |S| \leq k$ é:

1. Seja $\mathscr{V} = \{A \backslash \{p\} \mid \text{aresta } A \text{ contem } S\}$, $p \in S$.
2. Remova as arestas que contém S.
3. Adicione um vértice novo, v e novas arestas: $T \cup \{v\}$, para todo $T \in \mathscr{V}$

Mais exemplos

Dadas matrizes 3×3 de inteiros, existe um produto delas (com repetições) que dá a matriz nula?

Um k-hipergrafo consiste de um conjunto finito V de vértices e uma coleção de k-subconjuntos de V, as arestas. A subdivisão em $S \subset V$ com $1 < |S| \leq k$ é:

1. Seja $\mathscr{V} = \{A \backslash \{p\} \mid \text{aresta } A \text{ contem } S\}$, $p \in S$.
2. Remova as arestas que contém S.
3. Adicione um vértice novo, v e novas arestas: $T \cup \{v\}$, para todo $T \in \mathscr{V}$

Problema: dado um 5-hipergrafo, ele tem uma subdivisão comum com o de 6 vértices e todas arestas possíveis?

Mais exemplos

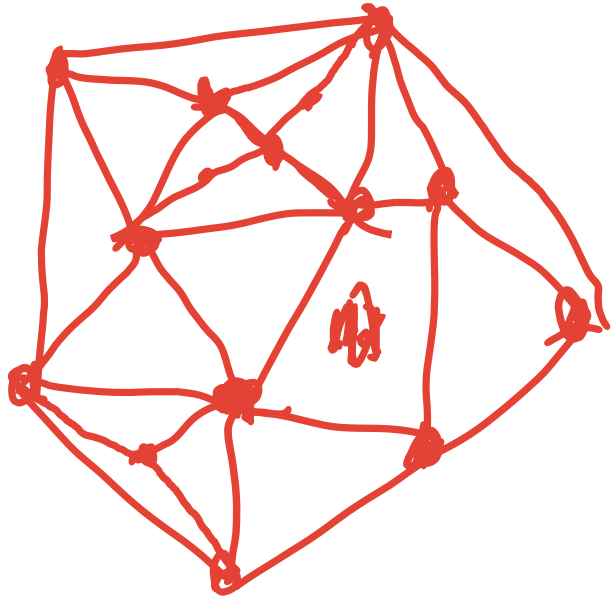
Dadas matrizes 3×3 de inteiros, existe um produto delas (com repetições) que dá a matriz nula?

Um k-hipergrafo consiste de um conjunto finito V de vértices e uma coleção de k-subconjuntos de V, as arestas. A subdivisão em $S \subset V$ com $1 < |S| \leq k$ é:

1. Seja $\mathscr{V} = \{A \backslash \{p\} \mid \text{aresta } A \text{ contem } S\}$, $p \in S$.
2. Remova as arestas que contém S.
3. Adicione um vértice novo, v e novas arestas: $T \cup \{v\}$, para todo $T \in \mathscr{V}$

Problema: dado um 5-hipergrafo, ele tem uma subdivisão comum com o de 6 vértices e todas arestas possíveis? *Topologia*!

3-flip



Tudo não dá!

Teorema (de Rice)

Seja $\mathscr{C}$ um subconjunto próprio e não vazio do conjunto das linguagens recursivamente enumeráveis. Então, é indecidível:
Dada uma MT $\mathscr{M}$, a linguagem que ela semi-decide está em $\mathscr{C}$?

Tudo não dá!

Teorema (de Rice)

Seja $\mathscr{C}$ um subconjunto próprio e não vazio do conjunto das linguagens recursivamente enumeráveis. Então, é indecidível:
Dada uma MT $\mathscr{M}$, a linguagem que ela semi-decide está em $\mathscr{C}$?

Tudo não dá!

Teorema (de Rice)

Seja $\mathscr{C}$ um subconjunto próprio e não vazio do conjunto das linguagens recursivamente enumeráveis. Então, é indecidível:
Dada uma MT $\mathscr{M}$, a linguagem que ela semi-decide está em $\mathscr{C}$?

Dem: S.p.g., a linguagem vazia não está em $\mathscr{C}$: podemos trocar $\mathscr{C}$ pelo seu complemento. Seja M uma MT qualquer que semi-decide uma linguagem $L \in \mathscr{C}$.

Continuando

Redução do problema da parada a ele.
Dado $w = "\mathscr{M}""x"$, vamos descrever $\mathscr{T} = \tau(w)$, com duas fitas.

Continuando

Redução do problema da parada a ele.

Dado $w = \text{“}\mathscr{M}\text{”}\text{“}x\text{”}$, vamos descrever $\mathscr{T} = \tau(w)$, com duas fitas.

A fita 1 recebe a entrada y.

Continuando

Redução do problema da parada a ele.
Dado $w = "\mathscr{M}""x"$, vamos descrever $\mathscr{T} = \tau(w)$, com duas fitas.
A fita 1 recebe a entrada y.
$\mathscr{T}$:

1. Escreve w na fita 2, e executa $\mathscr{U}$ nessa fita.

Continuando

Redução do problema da parada a ele.
Dado $w = \text{“}\mathscr{M}\text{”}\text{“}x\text{”}$, vamos descrever $\mathscr{T} = \tau(w)$, com duas fitas.
A fita 1 recebe a entrada y.
$\mathscr{T}$:

1. Escreve w na fita 2, e executa $\mathscr{U}$ nessa fita.
2. Se $\mathscr{U}$ parou, executa M na fita 1.

Continuando

Redução do problema da parada a ele.
Dado $w = \text{“}\mathscr{M}\text{”}\text{“}x\text{”}$, vamos descrever $\mathscr{T} = \tau(w)$, com duas fitas.
A fita 1 recebe a entrada y.
$\mathscr{T}$:

1. Escreve w na fita 2, e executa $\mathscr{U}$ nessa fita.
2. Se $\mathscr{U}$ parou, executa M na fita 1.

Continuando

Redução do problema da parada a ele.

Dado $w = \text{“}\mathscr{M}\text{”}\text{“}x\text{”}$, vamos descrever $\mathscr{T} = \tau(w)$, com duas fitas.

A fita 1 recebe a entrada y.

$\mathscr{T}$:

1. Escreve w na fita 2, e executa $\mathscr{U}$ nessa fita.
2. Se $\mathscr{U}$ parou, executa M na fita 1.

Se $(\mathscr{M}, x)$ não para, $\mathscr{U}(w)$ não para, e a linguagem de $\mathscr{T}$ é $\varnothing \notin \mathscr{C}$,

Continuando

Redução do problema da parada a ele.

Dado $w = \text{“}\mathscr{M}\text{”}\text{“}x\text{”}$, vamos descrever $\mathscr{T} = \tau(w)$, com duas fitas.

A fita 1 recebe a entrada y.

$\mathscr{T}$:

1. Escreve w na fita 2, e executa $\mathscr{U}$ nessa fita.
2. Se $\mathscr{U}$ parou, executa M na fita 1.

Se $(\mathscr{M}, x)$ não para, $\mathscr{U}(w)$ não para, e a linguagem de $\mathscr{T}$ é $\varnothing \notin \mathscr{C}$,

Se $(\mathscr{M}, x)$ para, $\mathscr{U}(w)$ para, e o resto do processamento é (M, y), que para sse $y \in L$; ou seja, $\mathscr{T}$ semi-decide $L \in \mathscr{C}$.

Significância maior

O Teorema de Rice vale para qualquer formalismo equivalente a máquinas de Turing:

1. Programas nas mais variadas linguagens de programação

Significância maior

O Teorema de Rice vale para qualquer formalismo equivalente a máquinas de Turing:

1. Programas nas mais variadas linguagens de programação
2. Funções μ-recursivas.

Significância maior

O Teorema de Rice vale para qualquer formalismo equivalente a máquinas de Turing:

1. Programas nas mais variadas linguagens de programação
2. Funções μ-recursivas.
3. Sistemas de reescritura.

Significância maior

O Teorema de Rice vale para qualquer formalismo equivalente a máquinas de Turing:

1. Programas nas mais variadas linguagens de programação
2. Funções μ-recursivas.
3. Sistemas de reescritura.

⋮

Sistemas de reescritura

Formato geral:

Sistemas de reescritura

Formato geral:

- Um ou mais alfabetos.

Sistemas de reescritura

Formato geral:

- Um ou mais alfabetos.
- Regras de reescritura: pares $x \to y$.

Sistemas de reescritura

Formato geral:

- Um ou mais alfabetos.
- Regras de reescritura: pares $x \to y$.
- Regras de derivação: a partir de uma palavra, substituir a ocorrência de um lado da regra pelo outro.

Sistemas de reescritura

Formato geral:

- Um ou mais alfabetos.
- Regras de reescritura: pares $x \to y$.
- Regras de derivação: a partir de uma palavra, substituir a ocorrência de um lado da regra pelo outro.
- Regras de geração: como começar e que palavras aceitar.

Sistemas de reescritura

Formato geral:

- Um ou mais alfabetos.
- Regras de reescritura: pares $x \to y$.
- Regras de derivação: a partir de uma palavra, substituir a ocorrência de um lado da regra pelo outro.
- Regras de geração: como começar e que palavras aceitar.

Sistemas de reescritura

Formato geral:

- Um ou mais alfabetos.
- Regras de reescritura: pares $x \to y$.
- Regras de derivação: a partir de uma palavra, substituir a ocorrência de um lado da regra pelo outro.
- Regras de geração: como começar e que palavras aceitar.

Exemplos:

Sistemas de reescritura

Formato geral:

- Um ou mais alfabetos.
- Regras de reescritura: pares $x \to y$.
- Regras de derivação: a partir de uma palavra, substituir a ocorrência de um lado da regra pelo outro.
- Regras de geração: como começar e que palavras aceitar.

Exemplos:

1. Gramáticas generalizadas.

Sistemas de reescritura

Formato geral:

- Um ou mais alfabetos.
- Regras de reescritura: pares $x \to y$.
- Regras de derivação: a partir de uma palavra, substituir a ocorrência de um lado da regra pelo outro.
- Regras de geração: como começar e que palavras aceitar.

Exemplos:

1. Gramáticas generalizadas.
2. Sistemas de Markov.

Sistemas de reescritura

Formato geral:

- Um ou mais alfabetos.
- Regras de reescritura: pares $x \to y$.
- Regras de derivação: a partir de uma palavra, substituir a ocorrência de um lado da regra pelo outro.
- Regras de geração: como começar e que palavras aceitar.

Exemplos:

1. Gramáticas generalizadas.
2. Sistemas de Markov.
3. Sistemas de Lyndenmayer.

Turing completo — computa todas funções recursivas

L-systems