Problema da Parada

Teoria da Computação

INF05501

Relembrando

- O Problema da Auto-Aplicação tenta determinar a **existência de um programa universal**, o qual é capaz de processar qualquer outro programa, incluindo ele mesmo

- Tal problema é **indecidível** (não-solucionável)

- O Princípio da Redução permite que partamos de problemas cuja a classe de solucionabilidade seja conhecida e possamos **determinar a classe de outros problemas**

- Logo, podemos usar o Problema da Auto-Aplicação para determinar a **decidibilidade de outros problemas**

Problema da Parada

Dadas uma Máquina Universal M e uma entrada qualquer w sobre o alfabeto de entrada Σ de M, decidir se M irá parar para w

Problema da Parada

Dadas uma Máquina Universal M e uma entrada qualquer w
sobre o alfabeto de entrada Σ de M,
decidir se M irá parar para w

Portanto, podemos definir tal problema através da seguinte linguagem:

$L_P = \{(m, w) \mid m = codigo(M)$ e $w \in ACEITA(M) \cup REJEITA(M)\}$

Teorema 1

Problema da Parada é Não-Solucionável

Teorema 1

Problema da Parada é Não-Solucionável

Portanto, deve-se demonstrar que L_P não é recursiva

Prova do Teorema 1

Supondo-se que L_P **seja recursiva**, então existe uma **Máquina Universal** M_P a qual aceita L_P e **sempre para para qualquer entrada**

Logo:

$ACEITA(M_P) = L_P$

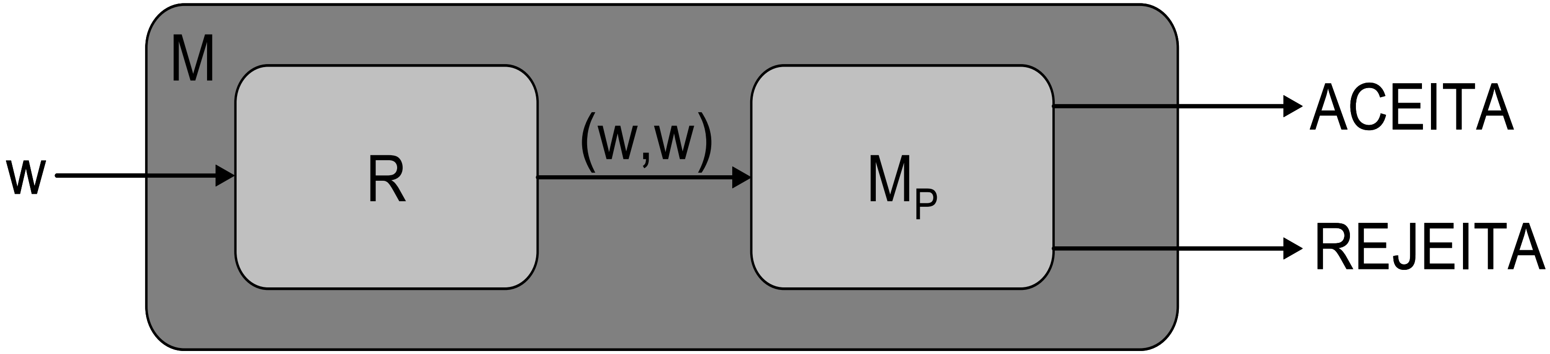
$REJEITA(M_P) = \Sigma^* - L_P$

$LOOP(M_P) = \emptyset$

Prova do Teorema 1 (cont.)

Suponha-se a existência de uma Máquina Universal R que, para qualquer entrada w, gera o par (w, w)

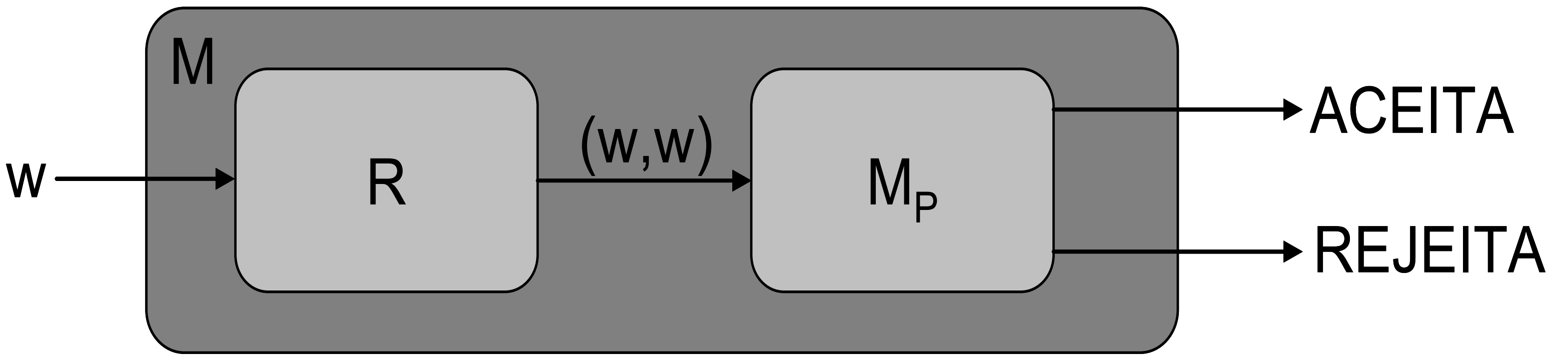
Seja M uma máquina da seguinte forma:



Prova do Teorema 1 (cont.)

Suponha-se a existência de uma Máquina Universal R que, para qualquer entrada w, gera o par (w, w)

Seja M uma máquina da seguinte forma:



Problema da Auto-Aplicação foi reduzido ao Problema da Parada!!

Prova do Teorema 1 (cont.)

Portanto:

- Se $w \in L_{AA}$, então $R(w) = (w, w) \in L_P$
- Se $w \notin L_{AA}$, então $R(w) = (w, w) \notin L_P$

Como supõe-se que L_P seja recursiva e L_{AA} foi reduzida a L_P, então L_{AA} **deve ser recursiva**

Prova do Teorema 1 (cont.)

Isto significa que o **Problema da Auto-Aplicação deve ser solucionável**

Prova do Teorema 1 (cont.)

Isto significa que o **Problema da Auto-Aplicação deve ser solucionável**

Visto que o **Problema da Auto-Aplicação foi provado ser não-solucionável**, então é absurdo supor-se que o Problema da Parada seja solucionável

Prova do Teorema 1 (cont.)

Isto significa que o **Problema da Auto-Aplicação deve ser solucionável**

Visto que o **Problema da Auto-Aplicação foi provado ser não-solucionável**, então é absurdo supor-se que o Problema da Parada seja solucionável

Logo, o Problema da Parada é não-solucionável

Teorema 2

Problema da Parada é Parcialmente Solucionável

Teorema 2

Problema da Parada é Parcialmente Solucionável

Portanto, deve-se demonstrar que L_P é enumerável recursivamente

Teorema 2

Problema da Parada é Parcialmente Solucionável

Portanto, deve-se demonstrar que L_P é enumerável recursivamente

Prova é análoga à do Teorema 1

Outro Problemas de Decisão

- Muitos problemas conhecidos são **não-solucionáveis**

- Alguns deles se apresentam com **variações de outros problemas conhecidos** e, desta forma, tornam a aplicação da **redução mais simples e direta**

- Por exemplo, são problemas relacionados ao Problema da Parada:
  - Problema da Palavra Vazia
  - Problema da Totalidade
  - Problema da Equivalência
  - Problema da Vacuidade

Problema da Parada da Palavra Vazia

Dada uma Máquina Universal M, existe um algoritmo que verifique se M para, aceitando ou rejeitando, ao processar a entrada vazia?

Problema da Parada da Palavra Vazia

Dada uma Máquina Universal M, existe um algoritmo que verifique se M para, aceitando ou rejeitando, ao processar a entrada vazia?

O Problema da Palavra Vazia é uma **variação do Problema da Parada**, restringindo-se a entrada à palavra vazia (ou à ausência de entrada)

Problema da Parada da Palavra Vazia

Dada uma Máquina Universal M, existe um algoritmo que verifique se M para, aceitando ou rejeitando, ao processar a entrada vazia?

O Problema da Palavra Vazia é uma **variação do Problema da Parada**, restringindo-se a entrada à palavra vazia (ou à ausência de entrada)

Portanto, podemos definir tal problema através da seguinte linguagem:

$L_\varepsilon = \{m \mid m = codigo(M)$ e $\varepsilon \in ACEITA(M) \cup REJEITA(M)\}$

Teorema 3

Problema da Parada da Palavra Vazia é Não-Solucionável

Teorema 3

Problema da Parada da Palavra Vazia é Não-Solucionável

Portanto, L_ε não deve ser recursiva

Teorema 3

Problema da Parada da Palavra Vazia é Não-Solucionável

Portanto, L_ε não deve ser recursiva

Para a prova, usa-se o Princípio da Redução, reduzindo-se L_P (linguagem que descreve o Problema da Parada) à linguagem L_ε

Prova do Teorema 3

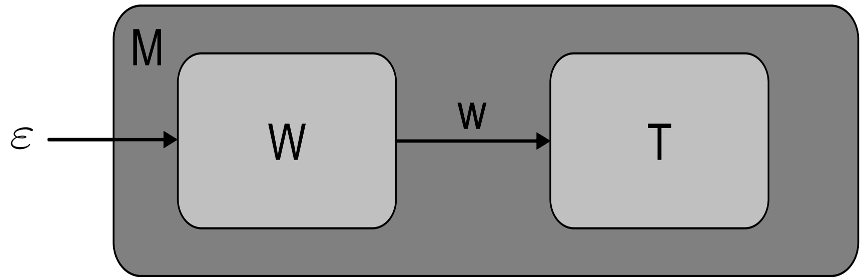
Sejam:

T um **Máquina Universal** qualquer definida sobre o alfabeto Σ

w uma **palavra** qualquer sobre Σ

W uma **Máquina Universal** que recebe como entrada a palavra vazia e gera a palavra w

M uma **Máquina Universal** definida em termos de T e W, da seguinte maneira:



Prova do Teorema 3 (cont.)

Podemos chegar às seguintes conclusões:

- Se T aceita a palavra w, então M aceita a palavra vazia
- Se T não aceita a palavra w (rejeita ou fica em loop), então M não aceita a palavra vazia (rejeita ou fica em loop)

Prova do Teorema 3 (cont.)

Ou seja, supondo-se t e m como os códigos de T e M, respectivamente:

- Se $(t, w) \in L_P$, então $m \in L_\varepsilon$
- Se $(t, w) \notin L_P$, então $m \notin L_\varepsilon$

Portanto, o **Problema da Parada é reduzido ao Problema da Parada da Palavra Vazia**

Prova do Teorema 3 (cont.)

Ou seja, supondo-se t e m como os códigos de T e M, respectivamente:

- Se $(t, w) \in L_P$, então $m \in L_\varepsilon$
- Se $(t, w) \notin L_P$, então $m \notin L_\varepsilon$

Portanto, o **Problema da Parada é reduzido ao Problema da Parada da Palavra Vazia**

Como o Problema da Parada é não-solucionável (L_P não é recursiva), o Problema da Parada da Palavra Vazia também é não-solucionável (L_ε não é recursiva)

Problema da Totalidade

Dada uma Máquina Universal M qualquer, existe um algoritmo que verifique se M para, aceitando ou rejeitando, ao processar qualquer entrada?

Problema da Totalidade

Dada uma Máquina Universal M qualquer, existe um algoritmo que verifique se M para, aceitando ou rejeitando, ao processar qualquer entrada?

O Problema da Totalidade é uma **variação do Problema da Parada** onde não há restrições quanto às possíveis entradas

Problema da Totalidade

Dada uma Máquina Universal M qualquer, existe um algoritmo que verifique se M para, aceitando ou rejeitando, ao processar qualquer entrada?

O Problema da Totalidade é uma **variação do Problema da Parada** onde não há restrições quanto às possíveis entradas

Tal problema pode ser traduzido na seguinte linguagem:

$L_T = \{m \mid m = codigo(M)$ e $LOOP(M) = \emptyset\}$

Teorema 4

Problema da Totalidade é Não-Solucionável

Teorema 4

Problema da Totalidade é Não-Solucionável

Portanto, L_T não deve ser recursiva

Teorema 4

Problema da Totalidade é Não-Solucionável

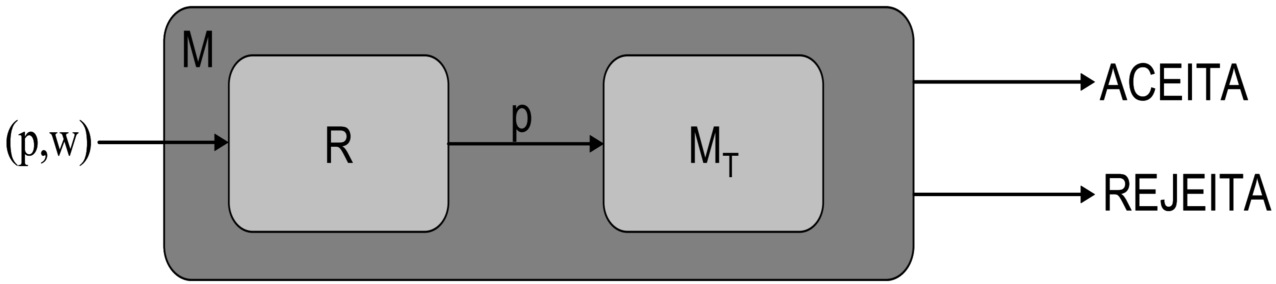
Portanto, L_T não deve ser recursiva

Para a prova, usa-se o Princípio da Redução, reduzindo-se L_P (linguagem que descreve o Problema da Parada) à linguagem L_T

Prova do Teorema 4

Supondo-se que L_T **seja recursiva**, então deve existir uma **Máquina Universal** M_T **que sempre para, onde** $ACEITA(M_T) = L_T$

Supondo-se uma Máquina Universal R que, para qualquer entrada (p, w), gera p, tem-se uma Máquina Universal M definida usando R e M_T como segue:



Prova do Teorema 4 (cont.)

Desta forma, o Problema da Parada foi reduzido ao Problema da Totalidade, pois

- Se $(p, w) \in L_P$, então $R((p, w)) \in L_T$
- Se $(p, w) \notin L_P$, então $R((p, w)) \notin L_T$

Prova do Teorema 4 (cont.)

Desta forma, o Problema da Parada foi reduzido ao Problema da Totalidade, pois

- Se $(p, w) \in L_P$, então $R((p, w)) \in L_T$
- Se $(p, w) \notin L_P$, então $R((p, w)) \notin L_T$

Como é suposto que o Problema da Totalidade é solucionável (L_T é recursiva), então o **Problema da Parada deveria ser solucionável**

Prova do Teorema 4 (cont.)

Desta forma, o Problema da Parada foi reduzido ao Problema da Totalidade, pois

- Se $(p, w) \in L_P$, então $R((p, w)) \in L_T$
- Se $(p, w) \notin L_P$, então $R((p, w)) \notin L_T$

Como é suposto que o Problema da Totalidade é solucionável (L_T é recursiva), então o **Problema da Parada deveria ser solucionável**

Como o Problema da Parada é não-solucionável (L_P não é recursiva), é absurdo dizer que ele é solucionável e, portanto, o Problema da Totalidade é não-solucionável

Problema da Equivalência

Dadas duas Máquinas Universais M e P quaisquer, existe um algoritmo que verifique se M e P reconhecem a mesma linguagem?

Problema da Equivalência

Dadas duas Máquinas Universais M e P quaisquer, existe um algoritmo que verifique se M e P reconhecem a mesma linguagem?

O Problema da Equivalência é também conhecido com **Problema da Equivalência de Compiladores**, referindo-se a decidir se dois compiladores reconhecem uma mesma linguagem; i.e., são **equivalentes em termos de reconhecimento desta linguagem**

Problema da Equivalência

Dadas duas Máquinas Universais M e P quaisquer, existe um algoritmo que verifique se M e P reconhecem a mesma linguagem?

O Problema da Equivalência é também conhecido com **Problema da Equivalência de Compiladores**, referindo-se a decidir se dois compiladores reconhecem uma mesma linguagem; i.e., são **equivalentes em termos de reconhecimento desta linguagem**

A linguagem que traduz tal problema é dada por:

$L_E = \{(m,p) \mid m = codigo(M), p = codigo(P),$
$ACEITA(M) = ACEITA(P)$ e $REJEITA(M) = REJEITA(P)\}$

Teorema 5

Problema da Equivalência é Não-Solucionável

Teorema 5

Problema da Equivalência é Não-Solucionável

Portanto, L_E não deve ser recursiva

Teorema 5

Problema da Equivalência é Não-Solucionável

Portanto, L_E não deve ser recursiva

Para a prova, usa-se o Princípio da Redução, reduzindo-se L_ε (linguagem que descreve o Problema da Parada da Palavra Vazia) à linguagem L_E

Prova do Teorema 5

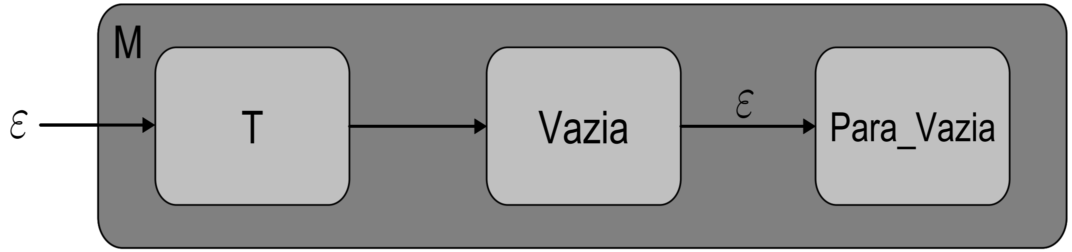
Sejam:

T uma **Máquina Universal** qualquer

$Vazia$ uma **Máquina Universal** que recebe qualquer palavra como entrada e sempre gera ε

$Para\_Vazia$ uma **Máquina Universal** que sempre para para a entrada vazia

M uma **Máquina Universal** definida em termos das máquinas anteriores como segue:



Prova do Teorema 5 (cont.)

Assim, supondo-se t e p como os codigos de T e $Para\_Vazia$, respectivamente, tem-se que:

- Se $t \in L_\varepsilon$, então $(t, p) \in L_E$
- Se $t \notin L_\varepsilon$, então $(t, p) \notin L_E$

Prova do Teorema 5 (cont.)

Assim, supondo-se t e p como os codigos de T e $Para\_Vazia$, respectivamente, tem-se que:

- Se $t \in L_{\varepsilon}$, então $(t, p) \in L_E$
- Se $t \notin L_{\varepsilon}$, então $(t, p) \notin L_E$

Desta forma, reduzimos o Problema da Parada da Palavra Vazia ao Problema da Equivalência

Prova do Teorema 5 (cont.)

Assim, supondo-se t e p como os codigos de T e $Para\_Vazia$, respectivamente, tem-se que:

- Se $t \in L_\varepsilon$, então $(t, p) \in L_E$
- Se $t \notin L_\varepsilon$, então $(t, p) \notin L_E$

Desta forma, reduzimos o Problema da Parada da Palavra Vazia ao Problema da Equivalência

Visto que O Problema da Parada da Palavra Vazia é não-solucionável, então o Problema da Equivalência também é não-solucionável

Problema da Vacuidade

Dada uma Máquina Universal M qualquer, existe um algoritmo que verifique se M ***não*** *para ao processar qualquer entrada?*

Problema da Vacuidade

*Dada uma Máquina Universal M qualquer, existe um algoritmo que verifique se M **não** para ao processar qualquer entrada?*

Isto é, o Problema da Vacuidade se refere a descobrir se T **não reconhece linguagem alguma**

Exercícios

1. Apresente a linguagem que descreve o Problema da Vacuidade (considere o alfabeto de entrada Σ)

2. O Problema da Vacuidade é solucionável ou não-solucionável?

3. Como podemos provar a resposta da questão anterior?