Princípio da Redução

Teoria da Computação

INF05501

Relembrando

- Vimos que a solucionabilidade de problemas pode ser verificada usando-se **problemas de decisão** descritos na forma de problemas de reconhecimento de linguagens

- Desta forma, um problema é solucionável se a correspondente **linguagem é recursiva** e é parcialmente solucionável se a **linguagem for enumerável recursivamente**

- Vimos também que usamos uma **função bijetora para codificar programas e/ou máquinas** para podermos transformá-los em problemas de decisão do tipo de reconhecimento de linguagens

Problema da Auto-Aplicação

Dado um programa monolítico arbitrário P para a Máquina Norma, decidir se a função computada $\langle P, Norma \rangle$ é definida para p, onde p é a codificação de P

Problema da Auto-Aplicação

Dado um programa monolítico arbitrário P para a Máquina Norma, decidir se a função computada $\langle P, Norma \rangle$ é definida para p, onde p é a codificação de P

Isto é:

"Existe um programa monolítico em Norma capaz de processar a si mesmo?"

Problema da Auto-Aplicação (cont.)

- Logo, o problema da auto-aplicação corresponde a, dado um programa monolítico arbitrário P para Norma, **decidir se a computação de P em Norma termina para a entrada p**

- Podemos descrever tal problema como uma linguagem:

  $L_{AA} = \{p \mid \langle P, Norma \rangle(p)$ é definida, P é programa de Norma e $p = cod\_bij(P)\}$

Teorema 1

Problema da Auto-Aplicação é Parcialmente Solucionável

Teorema 1

Problema da Auto-Aplicação é Parcialmente Solucionável

Isto é, a linguagem L_{AA} é enumerável recursivamente

Teorema 1

Problema da Auto-Aplicação é Parcialmente Solucionável

Isto é, a linguagem L_{AA} é enumerável recursivamente

Para a prova, precisamos mostrar que existe um programa monolítico Q para Norma, tal que:

$ACEITA(Q) = L_{AA}$

$REJEITA(Q) \cup LOOP(Q) = \Sigma^* - L_{AA}$

Prova do Teorema 1

Sejam:

P um **programa monolítico** qualquer para Norma

Q um **programa monolítico** para Norma capaz de **simular qualquer outro programa**

A entrada para Q é $p = cod\_bij(P)$, sendo que Q **simula** P **para a entrada** p

Prova do Teorema 1 (cont.)

$p \in ACEITA(Q)$ sss $\langle P, Norma \rangle(p)$ é definido, ou seja, a **computação** de P em Norma é **finita**

$p \in LOOP(Q)$ sss $\langle P, Norma \rangle(p)$ é indefinido, ou seja, a **computação** de P em Norma é **infinita**

$REJEITA(Q) = \emptyset$ (consequência dos dois casos acima)

Prova do Teorema 1 (cont.)

Portanto:

$ACEITA(Q) = L_{AA}$

$REJEITA(Q) \cup LOOP(Q) = \Sigma^* - L_{AA}$

Prova do Teorema 1 (cont.)

Portanto:

$ACEITA(Q) = L_{AA}$

$REJEITA(Q) \cup LOOP(Q) = \Sigma^* - L_{AA}$

Do que se conclui que L_{AA} é enumerável recursivamente

Prova do Teorema 1 (cont.)

Portanto:

$ACEITA(Q) = L_{AA}$

$REJEITA(Q) \cup LOOP(Q) = \Sigma^* - L_{AA}$

Do que se conclui que L_{AA} é enumerável recursivamente

Logo, o Problema da Auto-Aplicação é parcialmente solucionável

Teorema 2

Problema da Auto-Aplicação é Não-Solucionável

Teorema 2

Problema da Auto-Aplicação é Não-Solucionável

Isto é, a linguagem L_{AA} não é recursiva

Teorema 2

Problema da Auto-Aplicação é Não-Solucionável

Isto é, a linguagem L_{AA} não é recursiva

A prova é por absurdo, portanto, **supõe-se que L_{AA} é recursiva**

Então, existe uma programa Q para Norma tal que:

$ACEITA(Q) = L_{AA}$

$REJEITA(Q) = \Sigma^* - L_{AA}$

$LOOP(Q) = \emptyset$

Prova do Teorema 2

Suponha um programa R igual a Q, mas que possui um **trecho de programa** a mais, o qual é **executado ao final de cada computação finita** de Q

Tal trecho tem a seguinte **função**:

- Ao final da computação, **testa o valor de saída de** Q
- Se Q aceita ou rejeita, então R fica em loop infinito
- Se Q fica em loop infinito, então R aceita

Prova do Teorema 2 (cont.)

Assim, para a aplicação de R como entrada de R, tem-se, dado que $r = cod\_bij(R)$, que:

- R fica em loop infinito quando Q, ao simular R, aceita ou rejeita $\Rightarrow$ R fica em loop infinito quando Q para

- R para quando Q, ao simular R, fica em loop infinito $\Rightarrow$ R para quando Q fica em loop infinito

Prova do Teorema 2 (cont.)

Assim, para a aplicação de R como entrada de R, tem-se, dado que $r = cod\_bij(R)$, que:

- R fica em loop infinito quando Q, ao simular R, aceita ou rejeita $\Rightarrow$ R fica em loop infinito quando Q para

- R para quando Q, ao simular R, fica em loop infinito $\Rightarrow$ R para quando Q fica em loop infinito

Isto caracteriza uma **contradição** e, portanto, L_{AA} não é recursiva

Prova do Teorema 2 (cont.)

Assim, para a aplicação de R como entrada de R, tem-se, dado que $r = cod\_bij(R)$, que:

- R fica em loop infinito quando Q, ao simular R, aceita ou rejeita $\Rightarrow$ R fica em loop infinito quando Q para

- R para quando Q, ao simular R, fica em loop infinito $\Rightarrow$ R para quando Q fica em loop infinito

Isto caracteriza uma **contradição** e, portanto, L_{AA} não é recursiva

Logo, o Problema da Auto-Aplicação é não-solucionável

Princípio da Redução

- Consiste na construção de um algoritmo de mapeamento entre linguagens que codificam problemas

- Sabendo-se a **classe de uma das linguagens envolvidas**, pode-se estabelecer certas **conclusões acerca da outra linguagem**

- Deste modo, sendo conhecida a classe de solucionabilidade de um dos problemas codificados como uma linguagem, é possível derivarem-se **informações sobre a classe a que o outro problema codificado pertence**

Princípio da Redução (cont.)

- Para o estudo do Princípio da Redução, **supõe-se** o seguinte:
  - Função de codificação bijetora $codigo$ para qualquer Máquina Universal
  - Problema da Auto-Aplicação para qualquer Máquina Universal
- Tais suposições garantem a **generalidade** das definições e resultados a serem estudados
- Para a comparação de linguagens, usa-se uma máquina de redução

Máquina de Redução

Sejam dois problemas A e B e as correspondentes linguagens L_A e L_B

Uma Máquina de Redução R de L_A para L_B é tal que, para $w \in \Sigma$:

- *Se $w \in L_A$, então $R(w) \in L_B$*
- *Se $w \notin L_A$, então $R(w) \notin L_B$*

Portanto, o mapeamento de linguagens é uma função computável total

Teorema da Redução

Sejam dois problemas A e B e as correspondentes linguagens L_A e L_B

Se existe uma máquina de redução R de L_A para L_B (sobre um alfabeto Σ), então os seguintes resultados podem ser estabelecidos:

- *Se L_B é recursiva, então L_A é recursiva* (Caso 1)
- *Se L_B é enumerável recursivamente, então L_A é enumerável recursivamente* (Caso 2)
- *Se L_A não é recursiva, então L_B não é recursiva* (Caso 3)
- *Se L_A não é enumerável recursivamente, então L_B não é enumerável recursivamente* (Caso 4)

Prova do Teorema da Redução

Seja R *uma Máquina de Turing de Redução que* ***sempre para e que reduz*** L_A ***a*** L_B

Iremos considerar **cada caso** para a prova.

Prova do Teorema da Redução (cont.)

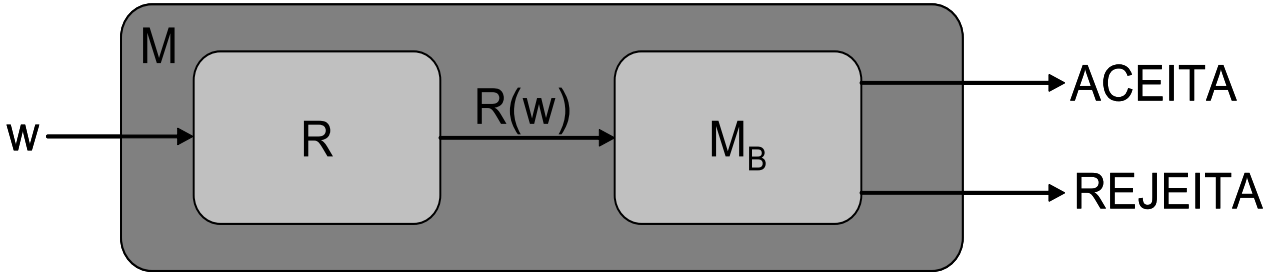
Caso 1: L_B é recursiva

Prova do Teorema da Redução (cont.)

Caso 1: L_B é recursiva

Se L_B é recursiva, então deve existir uma **Máquina Universal** M_B que aceita L_B e **sempre para para qualquer entrada**

Seja a seguinte Máquina Universal M:



Prova do Teorema da Redução (cont.)

Pode-se concluir que:

- M sempre para para qualquer entrada, pois R e M_B sempre param
- Se $w \in L_A$, então M aceita w, pois $R(w) \in L_B$
- Se $w \notin L_A$, então M rejeita w, pois $R(w) \notin L_B$

Portanto, M aceita L_A e sempre para para qualquer entrada

Logo, L_A é uma linguagem recursiva

Prova do Teorema da Redução (cont.)

Caso 2: L_B é enumerável recursivamente

Prova do Teorema da Redução (cont.)

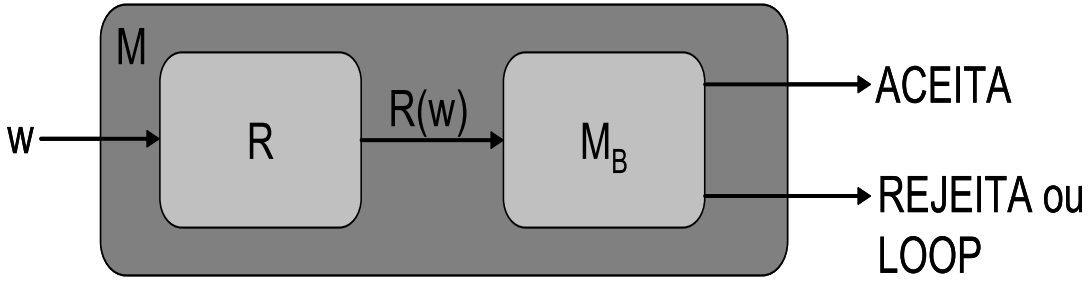
Caso 2: L_B é enumerável recursivamente

Se L_B é enumerável recursivamente, então deve existir uma **Máquina Universal** M_B tal que

$$ACEITA(M_B) = L_B$$

$$REJEITA(M_B) \cup LOOP(M_B) = \Sigma^* - L_B$$

Seja a seguinte Máquina Universal M:



Prova do Teorema da Redução (cont.)

Pode-se concluir que:

- Se $w \in L_A$, então M aceita w, pois $R(w) \in L_B$
- Se $w \notin L_A$, então M rejeita w ou fica em loop infinito, pois M_B rejeita ou fica em loop infinito para a entrada $R(w)$

Portanto, M aceita L_A, mas pode ficar em loop infinito para entradas que não pertencem à L_A

Logo, L_A é uma linguagem enumerável recursivamente

Prova do Teorema da Redução (cont.)

Por contraposição, os casos 3 (L_A não é recursiva) e 4 (L_A não é enumerável recursivamente) são equivalentes aos casos 1 e 2, respectivamente

Isto é,

r_A : A é recursiva

r_B : B é recursiva

e_A : A é enumerável recursivamente

e_B : B é enumerável recursivamente

$$(r_B \rightarrow r_A) \Leftrightarrow (\neg r_A \rightarrow \neg r_B)$$
$$(e_B \rightarrow e_A) \Leftrightarrow (\neg e_A \rightarrow \neg e_B)$$

Prova do Teorema da Redução (cont.)

Podemos estabelecer as seguintes relações entre problemas e linguagens que os codificam:

Linguagem recursiva $\Leftrightarrow$ Problema solucionável

Linguagem enumerável recursivamente $\Leftrightarrow$ Problema parcialmente solucionável

Prova do Teorema da Redução (cont.)

Então, dados dois problemas A e B, se for possível reduzir A a B, temos que

- Se B é **solucionável**, então A é **solucionável**

Prova do Teorema da Redução (cont.)

Então, dados dois problemas A e B, se for possível reduzir A a B, temos que

- Se B é **solucionável**, então A é **solucionável**
- Se B é **parcialmente solucionável**, então A é **parcialmente solucionável**

Prova do Teorema da Redução (cont.)

Então, dados dois problemas A e B, se for possível reduzir A a B, temos que

- Se B é **solucionável**, então A é **solucionável**
- Se B é **parcialmente solucionável**, então A é **parcialmente solucionável**
- Se A é **não-solucionável**, então B é **não-solucionável**

Prova do Teorema da Redução (cont.)

Então, dados dois problemas A e B, se for possível reduzir A a B, temos que

- Se B é **solucionável**, então A é **solucionável**
- Se B é **parcialmente solucionável**, então A é **parcialmente solucionável**
- Se A é **não-solucionável**, então B é **não-solucionável**
- Se A **não é parcialmente solucionável**, então B **não é parcialmente solucionável**

Tarefa

Em **grupos de 2 ou 3 componentes**, realizar o seguinte:

- Escolher um **problema indecidível**
- Enviar, por e-mail, nomes dos componentes do grupo e problema escolhido até **14/06**
- Escrever uma **pequena monografia** onde constem:
  - **Descrição informal** do problema
  - **Definição formal** do problema em termos de um problema de reconhecimento de linguagens
  - **Demonstração da indecidibilidade** do problema descrito (indicando o uso do Princípio da Redução)
  - **Bibliografia** utilizada (livro(s))
- **Entregar monografia e apresentar o trabalho ao professor no dia 23/06**