9 COMPUTABILIDADE

9 COMPUTABILIDADE

Objetivo do estudo da solucionabilidade de problemas

- investigar a existência ou não de algoritmos que solucionem determinada classe de problemas
- investigar os limites da computabilidade e, conseqüentemente, os limites do que pode, efetivamente, ser implementado em um computador
- evitar a pesquisa de soluções inexistentes.
  - Em 1901, Hilbert formulou uma lista de 23 problemas ainda não solucionados, que seriam o desafio para o novo Século;
  - Décimo problema: sabendo-se que as equações *diofantinas* são equações algébricas com coeficientes inteiros.
    Dada uma equação diofantina $P(a_0,\ldots,a_n) = 0$, onde P é um polinômio com coeficientes inteiros, quer-se saber se existe um procedimento efetivo que seja capaz de determinar, em um tempo finito, se suas raízes são inteiras.

Buscava-se estabelecer fundamentos rigorosos para a aritmética, visando representar todas as leis científicas em equações matemáticas. O princípio era estabelecer a **consistência formal** da aritmética e, após, estender as demais áreas do conhecimento. Buscou-se definir uma linguagem puramente sintática (sem significado) denominada: **sistema formal**. Recaiu-se no problema da existência ou não de um procedimento mecânico efetivo que decidisse a veracidade ou falsidade de qualquer enunciado aritmético.

Em 1928, David Hilbert propôs o *Problema de Decisão* (*Entscheidungsproblem*) - questionava a existência de um procedimento mecânico (baseado no trabalho de Gottfried Leibniz, que buscava mecanismo mecânico de manipulação de fórmulas) capaz de decidir se dado um enunciado (proposição) da lógica de primeira ordem, ele seria válido ou não, em um tempo finito.

Esta idéia de considerar a matemática um sistema formal empolgou os matemáticos no século XIX. Pretendia-se obter uma teoria aritmética como um sistema formal consistente e completo- o que, infelizmente, não foi possível, como foi afirmado por Kurt Gödel (em 1931) em seu *Teorema da Não-Completude* (*Incompleteness Theorem*):

Todas as formulações axiomáticas consistentes da Teoria dos

> *Números incluem proposições indecidíveis, ou seja, que não podem ser provadas como verdadeiras ou como falsas. Portanto, se um sistema formal é consistente, ele não pode ser completo e a consistência dos axiomas não pode ser provada, usando o próprio sistema formal.*

A demonstração usou uma codificação de primos (*Número de Gödel*) de forma que as formulações axiomáticas eram codificadas em números naturais.

1936, Alonzo Church provou que não existia nenhum algoritmo definido através de Funções Recursivas que decidisse se, para duas expressões dadas em Cálculo Lambda, se elas seriam ou não equivalentes.

1936, Alan Turing também contribuiu na resolução do Problema da Decisão, transformando-o em um *Problema da Parada* em sua máquina.

Na resolução do Problema de Decisão, uma boa parte do sucesso se deu pelo uso da Técnica da *Redução* (*Princípio da Redução*), na qual se transforma partes de um problema em outro, o que permite usar resultados já conhecidos para se obter a resolução de partes do problema atual e, assim, resolvê-lo.

1970, Yuri Matiyasevich, a solução definitiva foi apresentada. Ele provou que tal procedimento efetivo (algoritmo) não existe! A prova é complexa envolvendo teoria dos números e resultados anteriores sobre este problema. Basicamente se tem:

> *Se R é uma relação que pode ser codificada em um número natural, então R é uma equação diofantina se e somente se R é enumerável.*

Isto resulta que: o conjunto das equações *diofantinas* com raízes inteiras não é solucionável (decidível).

Problema de decisão é uma questão sobre um **sistema formal** com uma resposta do tipo sim ou não.

> ***Dados dois números naturais, eles são primos entre si?***
>
> A resposta para tal problema pode ser sim ou não, dependendo dos números tomados.

Um **problema de decisão** também pode ser formalizado como o problema de verificar se uma determinada palavra (cadeia finita de caracteres)

pertence ou não a uma **linguagem formal**.

O conjunto (linguagem formal) contém exatamente as palavras para as quais a resposta à pergunta é sim.

Problemas de decisão são problemas que tem pôr objetivo determinar se um dado elemento de algum universo pertence ou não a um determinado conjunto A (ou, equivalentemente, se satisfaz uma determinada propriedade).

- **Se existir um algoritmo** que receba como entrada um elemento qualquer x e retorne como saída sim, caso x pertença ao conjunto A, ou não, caso contrário, então se diz que o problema de decisão para o conjunto A é **solucionável**.
- **Se não existir um algoritmo** capaz de fazer essa avaliação, então se diz que o problema de decisão para o conjunto A é **não solucionável**.

Princípio da Redução

O estudo da solucionabilidade de um problema é feito na investigação da solucionabilidade de um problema a partir de outro, cuja classe de solucionabilidade é conhecida.

a) Sejam A e B dois problemas de decisão. Suponha que é possível modificar o problema A de tal forma que ele se porta como um caso do problema B;
b) Se A é não-solucionável então, como A é um caso de B, conclui-se que B também é não-solucionável;
c) Se B é solucionável então, como A é um caso de B, conclui-se que A também é solucionável.

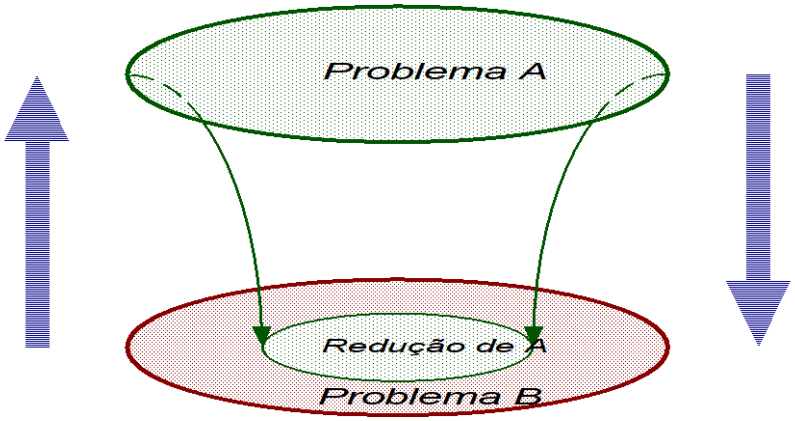


Figura 9.1 Princípio da Redução

Considere o caso de um aluno de um Curso de Ciência da Computação. Para se formar, necessita ser aprovado em todas as disciplinas que compõem a grade curricular do Curso, entre elas a disciplina de Teoria da Computação.

Considere então os problemas:

Problema A: "O aluno foi aprovado em Teoria da Computação?"
Problema B: "O aluno vai se formar?"

- Se determinado aluno x, não é aprovado em Teoria da Computação, Logo, o aluno x não se formará (pelos menos até conseguir a aprovação nessa disciplina).

- Já um outro aluno y, tem sua formatura confirmada, logo, o aluno y também foi aprovado em Teoria da Computação (e nas demais disciplinas da grade curricular).

9.1 Classes de Solucionabilidade de Problemas

Definição 9.1 Problema Solucionável.

Um problema é dito *Solucionável* ou *Totalmente Solucionável* se existe um algoritmo (Máquina Universal) que solucione o problema tal que sempre pára para qualquer entrada, com uma resposta afirmativa (ACEITA) ou negativa (REJEITA).

Definição 9.2 Problema Não-Solucionável.

Um problema é dito *Não-Solucionável* se *não* existe um algoritmo (Máquina Universal) que solucione o problema tal que sempre pára para qualquer entrada.

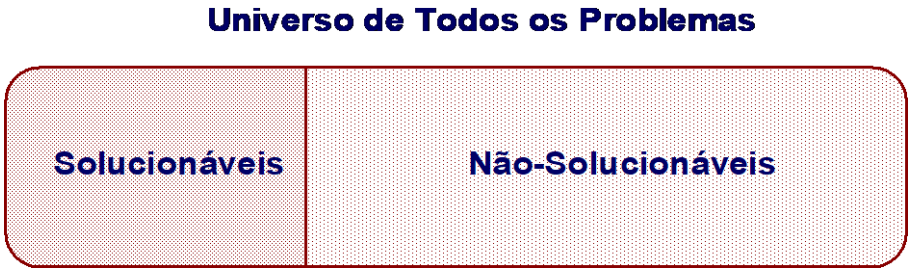


Figura 9.2 Particionamento do conjunto de todos problemas em classes

Definição 9.1
Problema Parcialmente Solucionável ou Computável.

Um problema é dito *Parcialmente Solucionável* ou *Computável* se existe um algoritmo (Máquina Universal) que solucione o problema tal que pare quando a resposta é afirmativa (ACEITA). Entretanto, quando a resposta esperada for negativa, o algoritmo pode parar (REJEITA) ou permanecer processando indefinidamente (LOOP).

Definição 9.2
Problema Completamente Insolúvel ou Não-Computável.

Um problema é dito *Completamente Insolúvel* ou *Não-Computável* se *não* existe um algoritmo (Máquina Universal) que solucione o problema tal que pare quando a resposta é afirmativa (ACEITA).

Figura 9.3 Particionamento do conjunto de todos problemas em classes

➢ Alguns problemas não-solucionáveis são parcialmente solucionáveis.

Relacionamento entre as classes de problemas:

- a união da Classe dos Problemas Solucionáveis com a Classe dos Problemas Não-Solucionáveis é o Universo de Todos os Problemas;
- a união da Classe dos Problemas Parcialmente Solucionáveis com a classe dos Problemas Completamente Insolúveis é o Universo de Todos os Problemas;
- a Classe dos Problemas Parcialmente Solucionáveis contém propriamente a Classe dos Problemas Solucionáveis e parte da Classe dos Não-Solucionáveis;
- todo problema solucionável é parcialmente solucionável;
- existem problemas não-solucionáveis que possuem solução parcial;
- os problemas completamente insolúveis não possuem solução total nem parcial.
- Para qualquer algoritmo que solucione um problema parcialmente solucionável que é não-solucionável, sempre existe pelo menos uma palavra de entrada que faz com que o algoritmo fique em *loop*. **Seu complemento é completamente insolúvel!!**

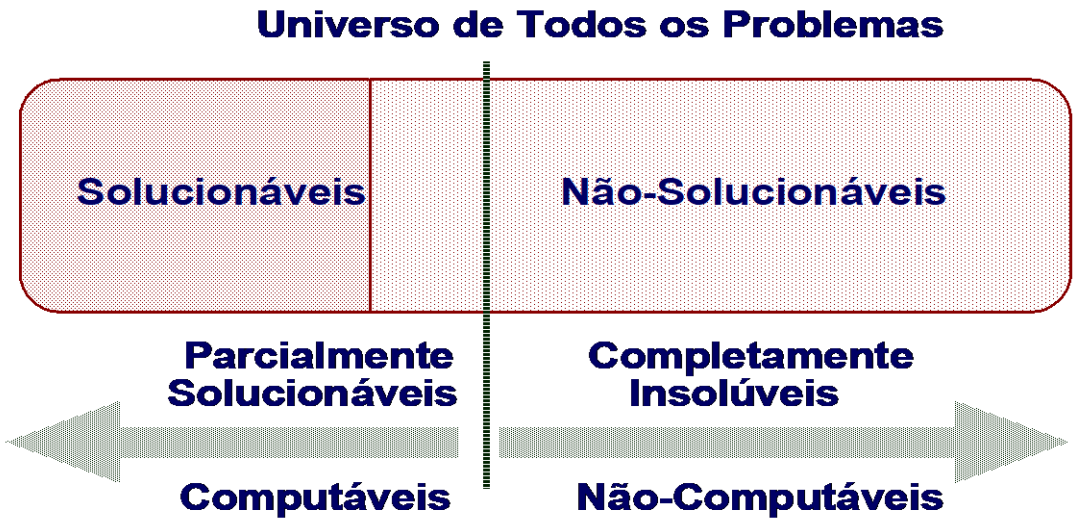


Figura 9.4 Relação entre as classes de problemas

Abordagem

- Concentra-se nos problemas com respostas binárias do tipo sim ou não (*problemas sim/não* ou *problemas de decisão)*.
- A vantagem de tal abordagem é que a verificação da solucionabilidade de um problema pode ser tratada como a verificação se determinada linguagem é recursiva, associando as condições de ACEITA/REJEITA de uma Máquina Universal às respostas sim/não, respectivamente.
- ***a Classe dos Problemas Solucionáveis é equivalente à Classe das Linguagens Recursivas***

Muitos problemas são não-solucionáveis.

a) *Detector Universal de Loops*. Dados um programa e uma entrada quaisquer, não existe algoritmo genérico capaz de verificar se o programa vai parar ou não para a entrada. **Este problema é universalmente conhecido como o *Problema da Parada*.**

b)*Equivalência de Compiladores*. Não existe algoritmo genérico que sempre pare capaz de comparar quaisquer dois compiladores de linguagens livres do contexto (reconhecidas pelo formalismo Autômato Não-Determinístico com Uma Pilha) como PASCAL, verificando se são equivalentes (se reconhecem a mesma linguagem);

Alguns problemas não-solucionáveis são parcialmente solucionáveis,

- Existe um algoritmo capaz de responder sim, embora, eventualmente, possa ficar em *loop* infinito para uma resposta que deveria ser não.
- problemas parcialmente solucionáveis são computáveis
- ***a Classe dos Problemas Parcialmente Solucionáveis é equivalente à Classe das Linguagens Enumeráveis Recursivamente***

Classe dos Problemas: Computáveis × Não-Computáveis

- o cardinal da Classe dos Problemas Computáveis é contável;
- o cardinal da Classe dos Problemas Não-Computáveis é não-contável.

  ⇒ pode-se afirmar que o cardinal da Classe dos Problemas Não-Computáveis é "muito maior" que o da Classe dos Problemas Computáveis, ou seja, existem muito mais problemas não-computáveis do que computáveis.

9.2 Problemas de Decisão

A essência de um problema de decisão

Dado um programa P para uma máquina universal M, decidir se a função computada ⟨P, M⟩ é total (ou seja, se a correspondente computação é finita).

➢ Não-solucionabilidade

- ♦ refere-se à inexistência de um método geral e efetivo para decidir se um programa para uma máquina universal pára para qualquer entrada.
- ♦ é possível existir métodos específicos para programas particulares.
- ♦ Importância da existência de problemas não-solucionáveis:
  - alguns desses problemas não-solucionáveis permitem estabelecer, por si sós, importantes resultados (como a inexistência de um detetor universal de *loops*);
  - demonstrar limitações da capacidade de expressarem-se soluções através de programas.
  - pode ser usada para verificar que outros problemas também o são, usando o princípio da redução.

Observação 9.5

Solucionabilidade de Problemas × Problema de Reconhecimento de Linguagens.

➢ A investigação da solucionabilidade de problemas em máquinas universais pode ser vista como um problema de reconhecimento de linguagens:

a) O problema é reescrito como um problema de decisão, capaz de gerar respostas do tipo afirmativo/negativo (sim/não);

b) Os argumentos do problema sim/não são codificados como palavras de um alfabeto, gerando uma linguagem.

➢ Assim, a questão da solucionabilidade de um problema pode ser traduzida como uma investigação se a linguagem gerada é recursiva (problema solucionável) ou enumerável recursivamente (problema parcialmente solucionável).

9.3 Codificação de Programas

➢ *A abordagem é específica para programas monolíticos da Máquina Norma*

Codificação de um programa como um número natural do Capítulo 3

➢ A função de codificação introduzida, código, não é bijetora (mas é injetora).
➢ Nem todo natural é codificação de algum programa.

Exemplo 9.1
Codificação Bijetora de Programas.

➢ Seja P o conjunto de todos os programas do tipo que está sendo considerado (monolítico, iterativo ou recursivo).
➢ Considere a seqüência de inteiros p_0, p_1, p_2,… formada pelos códigos dos programas de P (ordenados pela relação "menor" sobre os naturais).

⇒ Então: código_bij = λp.código_bij(p): P → N

código_bij(P) = n (para qualquer p∈P, se o código de P é p_n)

⇒ E, código_bij^{-1} = λn.código_bij^{-1}(n): N → P

código_ bij^{-1}(n)= P (para qualquer natural n retorna o programa P)

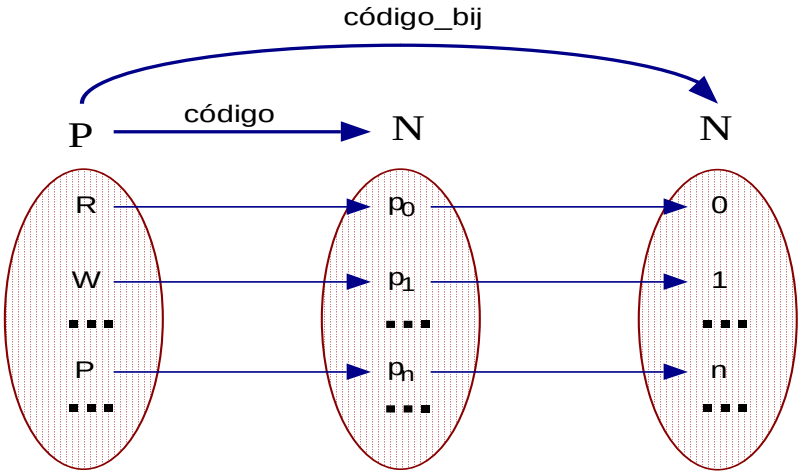


Figura 9.5 Função bijetora de codificação de programas

Um algoritmo para determinar código_bij(P)

- calcula p = código(P)
- verifica (usando *decodificação* – quantos números naturais menores do que p são codificações de programas, Se forem n-1, então código_bij(P) = n

Um algoritmo para determinar código_bij^{-1}(n)

- verifica (usando decodificação) os n primeiros números naturais que denotam codificações de programas.
- Se p é o n-ésimo natural, então o n-ésimo programa será a decodificação de p.

9.4 Problema da Autoaplicação

Definição 9.6
Problema da Autoaplicação.

Dado um programa monolítico arbitrário P para a Máquina Norma, decidir se a função computada ⟨P, Norma⟩ é definida para p, onde p é a codificação (bijetora) de P

- Dado um programa monolítico arbitrário P para Norma, decidir se a computação de P em Norma termina ou não, para a entrada p.

  LAA = { p | ⟨P, Norma⟩(p) é definida, P programa de Norma, p= código_bi (P) }

- Assim, a questão da solucionabilidade do Problema da Auto-Aplicação é a investigação se a linguagem é recursiva (problema solucionável) ou enumerável recursivamente (problema parcialmente solucionável).

Teorema 9.7
Problema da Autoaplicação é Parcialmente Solucionável.

LAA = { p | ⟨P, Norma⟩(p) é definida, P programa de Norma, p= código_bi(P) }

A linguagem LAA é enumerável recursivamente.

PROVA

- É necessário mostrar que existe um programa monolítico Q para Norma, tal que: **ACEITA(Q)** = LAA e **REJEITA(Q)∪LOOP(Q)**= N -LAA (N=Σ*)
- Sejam
  P um programa monolítico qualquer para Norma
  Q um programa monolítico para Norma **capaz de simular qualquer outro programa**. Ele recebe a entrada p = código_bi(P) e simula P para a entrada p.

  - **p ∈ ACEITA(Q)** se, e somente se, ⟨P, Norma⟩(p) é definido, ou seja, a computação de P em Norma é finita;
  - **p ∈ LOOP(Q)** se, e somente se, a computação de P em Norma é infinita;
  - **REJEITA(Q)** = ∅ (conseqüência dos dois casos acima).

Portanto:

ACEITA(Q) = LAA

REJEITA(Q) ∪ LOOP(Q) = N - LAA

- Logo, LAA **é enumerável recursivamente**
- Portanto, o Problema da Auto-Aplicação é parcialmente solucionável.

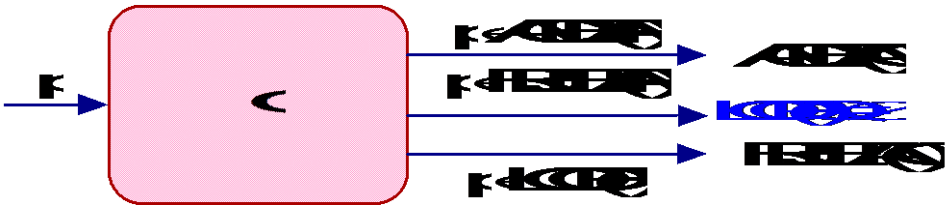
Teorema 9.8
Problema da Autoaplicação é Não-Solucionável.

A linguagem LAA que traduz o Problema da Auto-Aplicação* não *é recursiva.

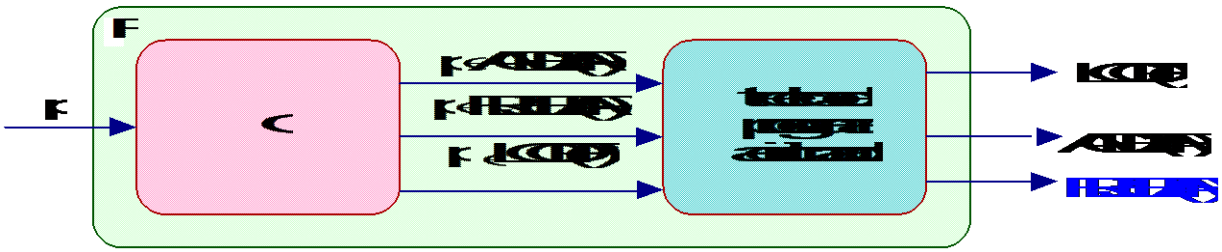
PROVA (por redução ao absurdo)

Suponha que LAA é recursiva.

- Então existe um programa monolítico Q para Norma, tal que:
  ACEITA(Q) = LAA
  REJEITA(Q) = N - LAA
  LOOP(Q) = ∅



- Suponha o programa R como Q, mas adicionando um trecho de programa que é executado ao final de cada computação finita de Q, com a seguinte função: antes de terminar a computação, testa o valor da saída de Q atribuindo:
  - se Q aceita ou rejeita, então R fica em *loop infinito*;
  - se Q fica em *loop infinito*, R aceita.



Aplicação de R como entrada de R, (suponha que r = código_bi(R)) tem-se que

- R fica em *loop infinito* quando Q, ao simular R, aceita ou rejeita. Ou seja, R fica em *loop infinito* quando R pára;
- R pára quando Q, ao simular R, fica em *loop infinito*. Ou seja, R pára quando R fica em *loop infinito*.

Assim, fica caracterizada a contradição.

- Logo, L*AA* *não* é recursiva
- Portanto, o Problema da Auto-Aplicação é *não-solucionável*.

9.5 Máquina de Redução

Definição 9.9
Máquina de Redução.

- Sejam dois problemas A e B e as correspondentes linguagens LA e LB.
- Uma *Máquina de Redução* R de LA para LB é tal que ($w \in \Sigma$):
  - Se $w \in$ LA, então $R(w) \in$ LB
  - Se $w \notin$ LA, então $R(w) \notin$ LB

$\Rightarrow$ Portanto, o mapeamento de linguagens é uma função computável total.

Teorema 9.10
Redução: Investigação da Solucionabilidade.

- Sejam dois problemas A e B e as correspondentes linguagens LA e LB.
- Se existe uma máquina de redução R de LA para LB (sobre um alfabeto Σ), então os seguintes resultados podem ser estabelecidos:
  - **a) Se LB é recursiva, então LA é recursiva;**
  - **b) Se LB é enumerável recursivamente, então LA é enumerável recursivamente;**
  - **c) Se LA não é recursiva, então LB não é recursiva;**
  - **d) Se LA não é enumerável recursivamente, então LB não é enumerável recursivamente.**

Seja R Máquina de Turing de Redução que sempre pára e que reduz LA a LB.

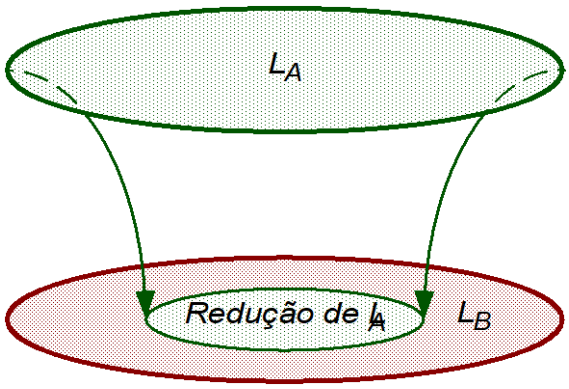
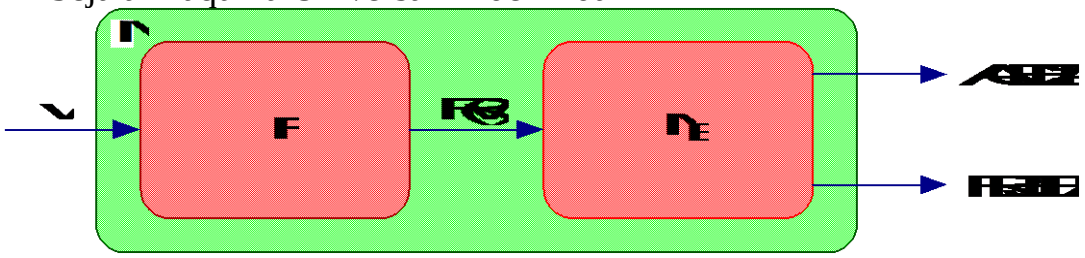


Figura 9. 8 Redução

PROVAS

a) Suponha que LB é uma linguagem recursiva. Então existe MB, Máquina Universal, que aceita LB e sempre pára para qualquer entrada.

- Seja a Máquina Universal M definida



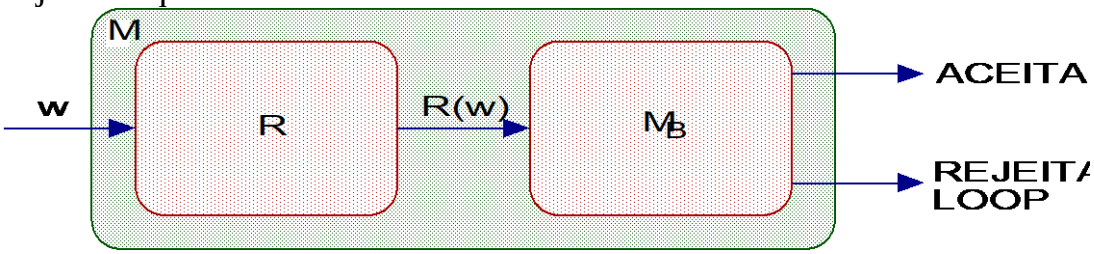
- As seguintes conclusões podem ser estabelecidas:
  - M sempre pára para qualquer entrada, pois R e MB sempre param;
  - se w ∈ LA, então M aceita w, pois R(w) ∈ LB
  - se w ∉ LA, então M rejeita w, pois R(w) ∉ LB
- Portanto, M aceita LA e sempre pára para qualquer entrada.
- Logo, LA é uma linguagem recursiva;

b) Suponha que LB é uma linguagem enumerável recursivamente. Então, existe MB, uma Máquina Universal, tal que:

ACEITA(MB) = LB

REJEITA(MB) ∪ LOOP(MB) = Σ* - LB

- Seja a Máquina Universal M definida



- As seguintes conclusões podem ser estabelecidas:
  - se w ∈ LA, então M aceita w, pois R(w) ∈ LB
  - se w ∉ LA, então M rejeita ou fica em *loop infinito* para a entrada w, pois MB rejeita ou fica em *loop infinito* para a entrada R(w).
- Portanto, M aceita LA, mas pode ficar em *loop infinito* para entradas não-pertencentes a LA.
- Logo, LA é uma linguagem enumerável recursivamente;

c) e d) Por contraposição, as afirmações c) e d) são equivalentes às afirmações a) e

b), respectivamente. Lembre-se de que (suponha que p e q são proposições):

$$(p \rightarrow q) \Leftrightarrow (\neg q \rightarrow \neg p)$$

9.6 Problema da Parada

Definição 9.11
Problema da Parada.

Dada uma Máquina Universal M qualquer e uma palavra w qualquer sobre o alfabeto de entrada, existe um algoritmo que verifique se M pára, aceitando ou rejeitando, ao processar a entrada w?

- Problema da Parada é um problema de decisão (do tipo sim/não) e pode ser redefinido pela seguinte linguagem:

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

Teorema 9.12
Problema da Parada é Não-Solucionável.

A linguagem L_P que traduz o Problema da Parada não é recursiva:

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

PROVA (por redução ao absurdo)

- Suponha que L_P é recursiva.

  Então existe uma Máquina Universal M_P sobre o alfabeto Σ, tal que:

  $ACEITA(M_P) = L_P$

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

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

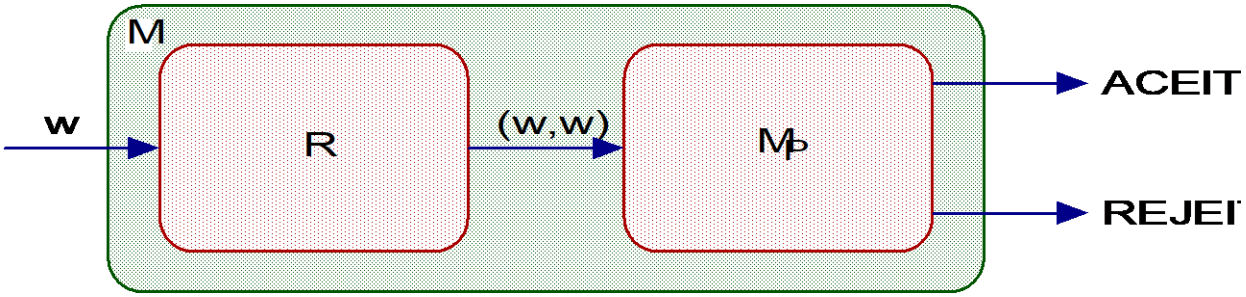
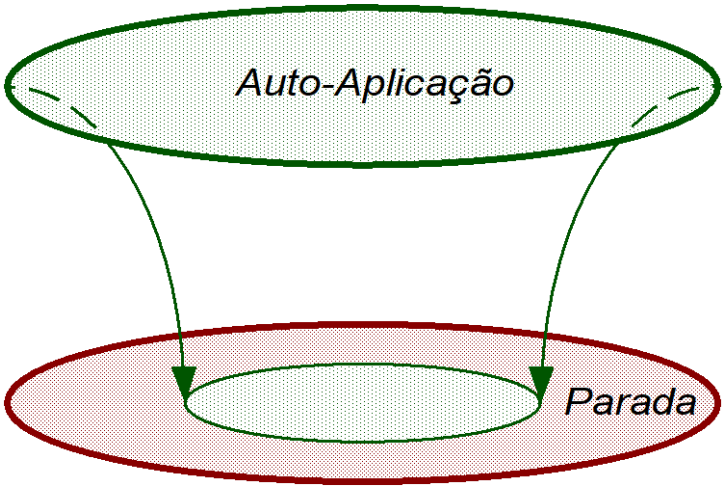


Figura 9.11 Máquina construída usando o princípio da redução



- Problema da Auto-Aplicação foi reduzido ao Problema da Parada:
  - se w ∈ LAA, então R(w) = (w, w) ∈ LP
  - se w ∉ LAA, então R(w) = (w, w) ∉ LP
- Como é suposto que o Problema da Parada é solucionável, então, pelo ***Teorema 9.10***, o Problema da Auto-Aplicação é solucionável, o que é um absurdo.
- Logo, é absurdo supor que o Problema da Parada é solucionável
- Portanto, é não-solucionável.

⇒ Analogamente ao problema da Auto-Aplicação, o Problema da Parada é parcialmente solucionável.

Teorema 9.13
Problema da Parada é Parcialmente Solucionável.

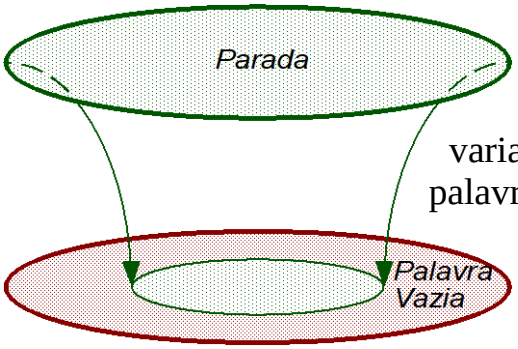
A linguagem que traduz o Problema da Parada é enumerável recursivamente:

LP = { (m, w) | m = código(M) e w ∈ ACEITA(M) ∪ REJEITA(M) }

9.7 Outros Problemas de Decisão

Definição 9.14
Problema da Parada da Palavra Vazia.

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



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

Teorema 9.15
Problema da Parada da Palavra Vazia é Não-Solucionável.

A linguagem que traduz o Problema da Parada da Palavra Vazia não é recursiva:

$$L\varepsilon = \{ m \mid m = \text{código}(M) \text{ e } \varepsilon \in \text{ACEITA}(M) \cup \text{REJEITA}(M) \}$$

- Logo, o Problema da Parada da Palavra Vazia é não-solucionável

Definição 9.16
Problema da Totalidade.

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

Teorema 9.17
Problema da Totalidade é Não-Solucionável.

A linguagem que traduz o Problema da Totalidade não é recursiva:

$$LT = \{ m \mid m = \text{código}(M) \text{ e } LOOP(M) = \varnothing \}$$

- Logo, é absurdo supor que o Problema da Totalidade é solucionável
  $\Rightarrow$ portanto, é não-solucionável.

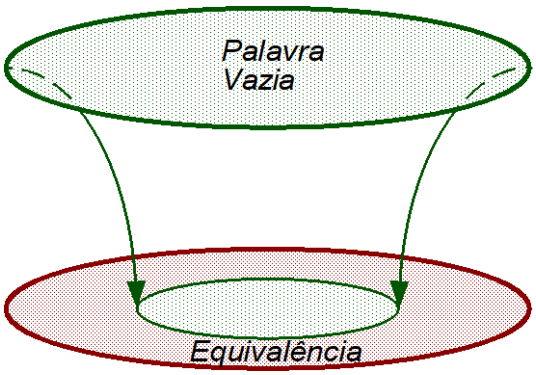
Definição 9.18
Problema da Equivalência.

O Problema da Equivalência é um problema de decisão (do tipo sim/não) que verifica a equivalência de duas máquinas universais.

Teorema 9.19
Problema da Equivalência é Não-Solucionável.

A linguagem que traduz o Problema da Equivalência não é recursiva:

$$LE = \{ (m, p) \mid m = \text{código}(M), p = \text{código}(P), ACEITA(M) = ACEITA(P) \text{ e } REJEITA(M) = REJEITA(P) \}$$



- Portanto, o Problema da Parada da Palavra Vazia é reduzido ao Problema da Equivalência
- Logo, o Problema da Equivalência é não-solucionável.

9.8 Problema da Correspondência de Post

O Problema da Correspondência de Post é definido sobre um *Sistema de Post*.

Definição 9.20
Sistema de Post.

Um Sistema de Post S definido sobre um alfabeto Σ é um conjunto finito e não-vazio de pares ordenados de palavras sobre Σ.

- É um conjunto da seguinte forma, onde $n>1$ e $x_i, y_i \in \Sigma^*$, para $i \in \{1,2, .., n\}$:

$$S = \{ (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n) \}$$

- Uma solução para um Sistema de Post é uma seqüência não-vazia de números naturais com valores em $\{ 1, 2, \ldots, n \}$: $i_1, i_2, \ldots, i_k$

  tal que: $x_{i1}x_{i2}\ldots x_{ik} = y_{i1}y_{i2}\ldots y_{ik}$

O *Problema da Correspondência de Post* é a investigação da existência de um algoritmo que analise qualquer Sistema de Post e determine se ele tem pelo menos uma solução.

Exemplo 9.2
Problema da Correspondência de Post.

Seja o Sistema de Post sobre $\Sigma = \{ a, b \}$ dado pelo seguinte conjunto:

$$S = \{ (b, bbb), (babbb, ba), (ba, a) \}$$

Uma solução de S é: **2, 1, 1, 3**

$\Rightarrow$ babbb b b ba = ba bbb bbb a

Exemplo 9.3
Problema da Correspondência de Post.

Seja o Sistema de Post sobre $\Sigma = \{ a, b \}$ dado pelo seguinte conjunto:

$$S = \{ (ab, abb), (b, ba), (b, bb) \}$$

S **não tem solução** pois, para qualquer par $(x_i, y_i) \in S$, tem-se que $|x_i| < |y_i|$.

Exemplo 9.4
Problema da Correspondência de Post.

Seja o Sistema de Post sobre $\Sigma = \{ a, b \}$ dado pelo seguinte conjunto:

$$S = \{ (a, ba), (bba, aaa), (aab, b), (ab, bba) \}$$

S **não tem solução** pois, para qualquer par $(x_i, y_i) \in S$, o primeiro símbolo de x_i é diferente do primeiro símbolo de y_i.

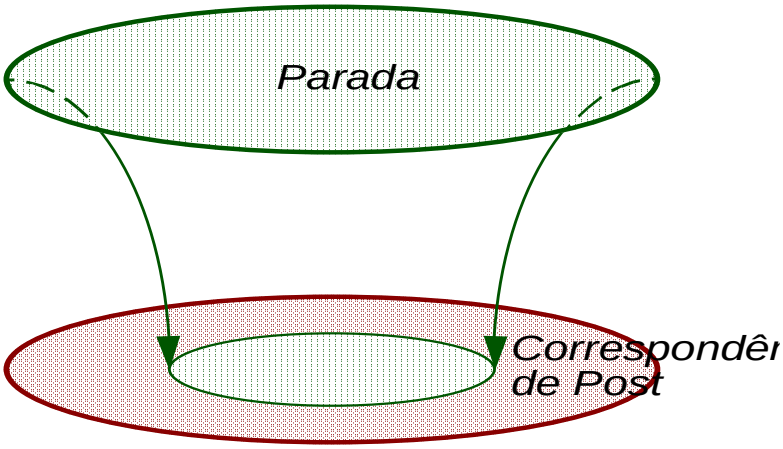
Teorema 9.21
Problema da Correspondência de Post é Não-Solucionável

A linguagem que traduz o Problema da Correspondência de Post *não* é recursiva:

LCP={s | s = código(S) e S é Sistema de Post com pelo menos uma solução }

PROVA

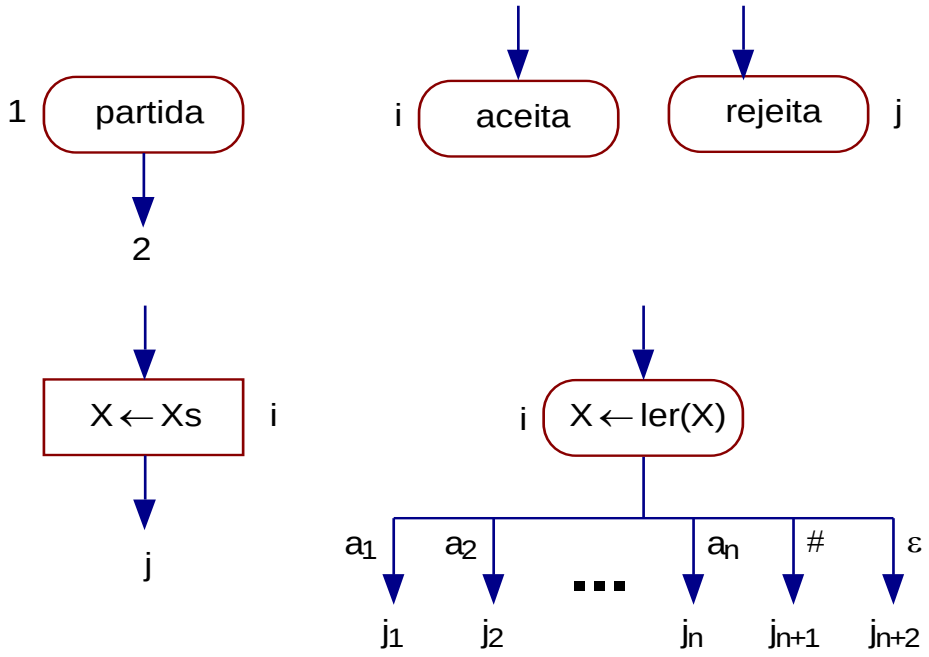
A partir de uma Máquina de Post M qualquer sobre o alfabeto Σ e de uma palavra w ∈ Σ* qualquer, constrói-se um Sistema de Post baseado na seqüência de comandos executados por M para a entrada w, de tal forma que o Sistema tenha solução se, e somente se, M pára para a entrada w.



- Portanto, o Problema da Parada é reduzido ao Problema da Correspondência de Post.
- A linguagem LP que traduz o Problema da Parada (que não é recursiva) é reduzida à linguagem LCP.

- A idéia básica da construção é enumerar os comandos da máquina M e, para cada ação sobre a variável X, gerar um par do Sistema de Post.
- Suponha que:
  - w ∈ Σ* uma palavra qualquer, onde $w = a_1a_2\ldots a_n$
  - o valor inicial de X é w
  - as componentes elementares do diagrama de fluxos de M são enumeradas sobre { 1, 2,…, m },

➢ **A relação entre os componentes elementares de M e os pares do correspondente Sistema de Post é como segue:**



- *Partida*. (1, 1 a1 a2…an 2)
- *Desvio ou Teste*. (i a1, j1), (i a2, j2),…, (i an, jn), (i #, jn+1), (i ε, jn+2)
- *Atribuição*. (i, s j)
- *Parada*. (i, ε)
- *Símbolo*. Cada símbolo s ∈ Σ∪{ # } (s, s)

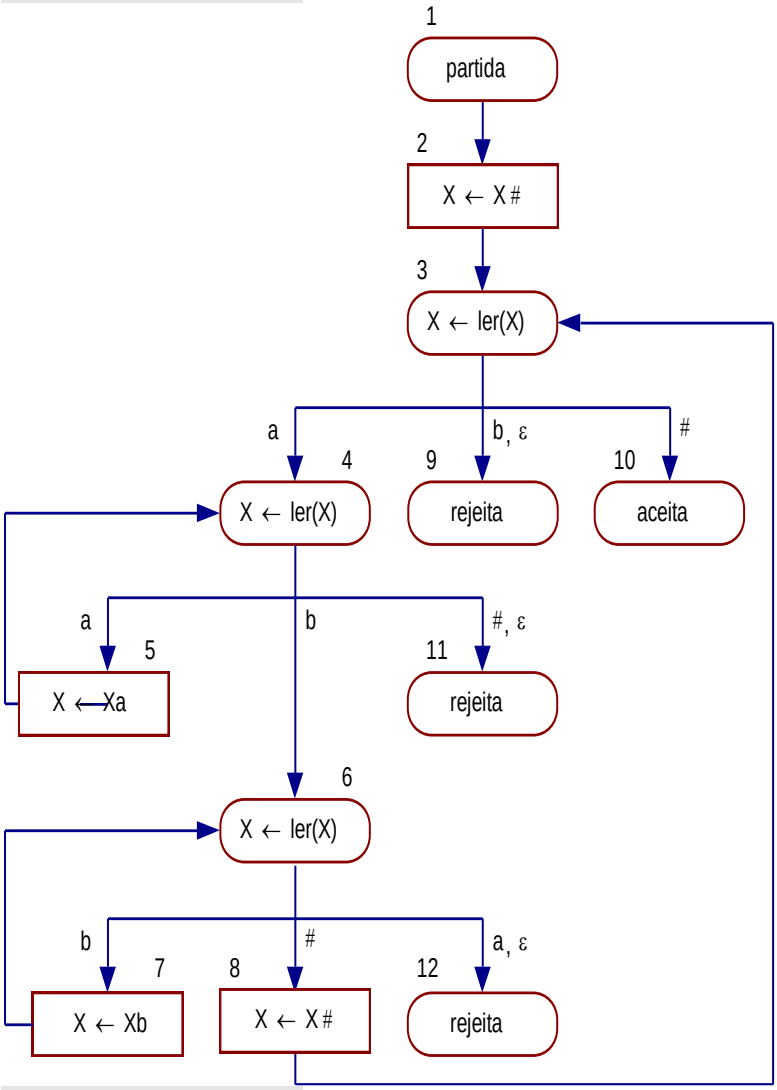
  - ♦ Deve-se reparar que, no par correspondente à partida, a segunda componente tem um número de instrução a mais que a primeira.
  - ♦ Essa diferença só é compensada no par correspondente à instrução de aceita/rejeita.
  - ♦ Assim, o Sistema Normal de Post somente tem solução se a máquina parar para a entrada w.

➢ Portanto, é possível reduzir o Problema da Parada ao Problema da Correspondência de Post

➢ Logo, o Problema da Correspondência de Post é não-solucionável.

Exemplo 9.5
Máquina de Post → Sistema de Post.

- Considere a Máquina de Post Post-Duplo_Bal, a qual reconhece a linguagem Duplo_Bal = { $a^n b^n \mid n \geq 0$ }. Suponha a entrada w = ab.
- Sistema de Post correspondente conforme o ***Te 5.20*** resulta em 24 pares,



1: (1, 1ab2)
2: (2, #3)
3: (3a, 4)
4: (3b, 9)
5: (3#, 10)
6: (3ε, 9)
7: (4a, 5)
8: (4b, 6)
9: (4#, 11)
10: (4ε, 11)

11: (5, a4)
12: (6a, 12)
13: (6b, 7)
14: (6#, 8)
15: (6ε, 12)
16: (7, b6)
17: (8, #3)
18: (9, ε)
19: (10, ε)
20: (11, ε)
21: (12, ε)
22: (a, a)
23: (b, b)
24: (#, #)

Uma solução para esse sistema correspondente ao processamento da Máquina Post-Duplo_Bal para a entrada w = ab, é a seqüência de pares:

1, 22, 23, 2, 22, 23, 24, 3, 23, 24, 8, 24, 14, 17, 24, 5, 19

1 a b 2 a b # 3a b # 4b # 6# 8 # 3# 10 =

= 1ab2 a b #3 a b # 4 b # 6 # 8 #3 # 10 ε

- Repare que a informação contida entre dois números de instruções é o valor da variável X após o processamento do comando à esquerda e antes do comando à direita.
- Os pares 22, 23 e 24 possuem objetivo de propiciar a rotação do conteúdo de X.

9.9 Propriedades da Solucionabilidade

- **O complemento de uma linguagem recursiva é uma linguagem recursiva. Conseqüentemente, existe um algoritmo que sempre pára e que reconhece o complemento da linguagem;**

  - **Uma linguagem L é recursiva se, e somente se, L e seu complemento são enumeráveis recursivamente;**

    - **A Classe das Linguagens Recursivas está contida propriamente na Classe das Linguagens Enumeráveis Recursivamente.**

- **o complemento de um problema solucionável é solucionável;**

  - **um problema é solucionável se, e somente se, o problema e seu complemento são parcialmente solucionáveis;**

    - **a Classe dos Problemas Solucionáveis esta contida propriamente na Classe dos Problemas Parcialmente Solucionáveis.**

Exemplo:

Problema da Parada:

- o Problema da Parada é parcialmente solucionável;
- o Problema da Parada é não-solucionável;
- portanto, o *Problema da Não-Parada* é não-solucionável.

Teorema 9.22
Complemento da Linguagem Recursiva é uma Linguagem Recursiva.

Se uma linguagem L sobre um alfabeto Σ qualquer é recursiva, então o seu complemento Σ* - L também é uma linguagem recursiva.

PROVA

- Suponha L uma linguagem recursiva sobre Σ.
  - Então existe M, Máquina Universal, que aceita a linguagem e sempre pára para qualquer entrada. Ou seja:

    ACEITA(M) = L

    REJEITA(M) = Σ* - L

    LOOP(M) = ∅
- Seja M' uma Máquina Universal construída a partir de M, mas invertendo-se as condições de ACEITA por REJEITA e vice-versa.
- Portanto, M' aceita Σ* - L e sempre pára para qualquer entrada. Ou seja:

  ACEITA(M') = Σ* - L

  REJEITA(M') = L

  LOOP(M') = ∅

⇒ **Logo Σ* - L é uma linguagem recursiva.**

Teorema 9.23
Linguagem Recursiva × Linguagem Enumerável Recursivamente

Uma linguagem L sobre um alfabeto Σ qualquer é recursiva se, e somente se, L e Σ* - L são enumeráveis recursivamente.

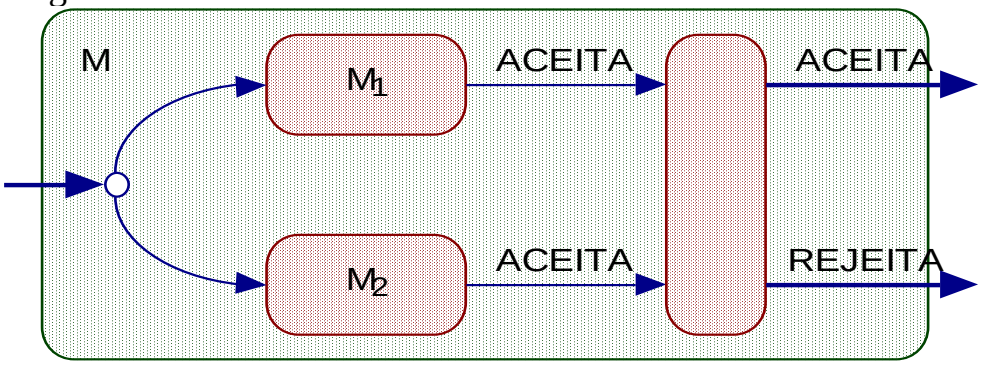
PROVA

a) Suponha L uma linguagem recursiva sobre Σ.

- Então, como foi mostrado no Teorema 5.22, Σ* - L é recursiva.
- Como toda linguagem recursiva também é enumerável recursivamente,
  ⇒ **então L e Σ* - L são enumeráveis recursivamente;**

b) Suponha L uma linguagem sobre Σ tal que L e Σ* - L são enumeráveis recursivamente.

- Então existem M1 e M2, Máquinas Universais tais que:
  ACEITA(M1) = L
  ACEITA(M2) = Σ* - L
- Seja M Máquina Universal não-determinística definida conforme esquema ilustrado na figura.



- Para qualquer palavra de entrada, M aceita-a se M1 aceitá-la e M rejeita-a se M2 aceitá-la.

⇒ **Portanto, claramente, M sempre pára.**
⇒ **Logo, L é recursiva.**

Teorema 9.24

Linguagens Recursivas ⊂ Enumeráveis Recursivamente

A Classe das Linguagens Recursivas está contida propriamente na Classe das Linguagens Enumeráveis Recursivamente.

PROVA

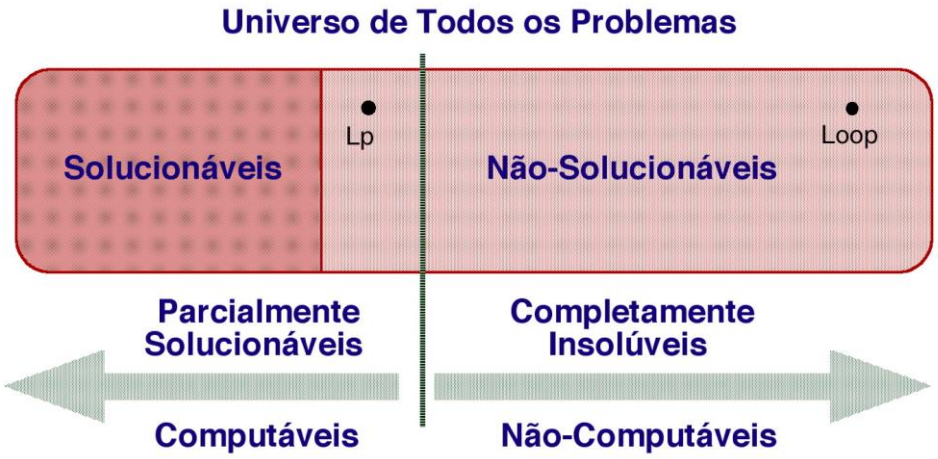
- Para mostrar que a inclusão é própria, basta mostrar que existe pelos menos uma linguagem enumerável recursivamente que não é recursiva.

Lembre-se que:

- o Problema da Parada é parcialmente solucionável, ou seja, é enumerável recursivamente
- o Problema da Parada é não-solucionável, ou seja, não é recursivo.
  - Portanto, o Problema da Parada é não recursivo.

Logo, a Classe das Linguagens Recursivas está contida propriamente na Classe das Linguagens Enumeráveis Recursivamente. □

- ***Importantes resultados podem ser obtidos a partir dessas propriedades como, por exemplo, sobre o Problema da Parada***
  - o Problema da Parada é parcialmente solucionável;
  - o Problema da Parada é não-solucionável;
  - portanto, o seu complemento que é o *Problema da Não-Parada* (Loop) é não-solucionável e completamente insolúvel, ou seja, não computável.



Uma conseqüência imediata desse resultado é a inexistência de um algoritmo genérico para identificar *loops* infinitos em sistemas.

- ***Da mesma forma, prova-se que o Problema da Não Correspondência de Post (complemento do Problema da Correspondência de Post) é Completamente Insolúvel.***
  - o Problema da Correspondência de Post pode ser reduzido ao Problema da Parada,
  - Problema da Parada, o qual é Parcialmente Solucionável e, também, não é Solucionável.
    - Em termos de linguagens, é Enumerável Recursivamente, mas não é Recursivo.
    - Se ele fosse Parcialmente Solucionável, estaria satisfeita a hipótese da segunda propriedade e implicaria que ambos seriam solucionáveis, caindo em contradição.
    - Logo, o Problema da Não Correspondência de Post é Não Computável.

Observação 9.25
Cardinalidade das Classes de Problemas

Considerando que:
*0 os problemas computáveis são caracterizados pela existência de um algoritmo implementado na forma de um programa para uma Máquina Universal;
*1 é possível estabelecer uma função bijetora entre o conjunto de todos os programas e o conjunto dos números naturais;

conclui-se que o cardinal da Classe dos Problemas Computáveis é igual ao cardinal dos números naturais, ou seja, é contável.

o cardinal da Classe dos Problemas Computáveis é contável.

É fácil definir um conjunto não-contável, como, por exemplo, o conjunto de todas as funções nos inteiros (que caracterizam o universo "discreto") e o conjunto de todas as funções nos reais (que caracterizam o universo "contínuo"). Assim:

existem classes de problemas que não são contáveis.

Portanto, o cardinal da Classe dos Problemas Não Computáveis é não-contável. Assim, informalmente, pode-se afirmar que:

Existem muito mais problemas não computáveis do que computáveis.

9.10 Conclusões

O estudo da computabilidade objetiva determinar a solucionabilidade de problemas (existência de algoritmos). Portanto, investiga os limites do que pode ser implementado em um computador evitando pesquisa de soluções inexistentes.

A abordagem é centrada nos problemas de decisão (do tipo sim/não). A investigação da solucionabilidade é suportada por:

- Princípio da Redução, o qual consiste na investigação da solucionabilidade de um problema a partir de outro, cuja classe de solucionabilidade é conhecida;

- Propriedades da solucionabilidade, as quais podem ser aplicadas para inferir a solucionabilidade ou não, conhecidas determinadas condições.

e permite, entre outros resultados, identificar classes de problemas:

- **Solucionáveis**: existe um algoritmo capaz de responder sim ou não e que sempre pára;
- **Parcialmente solucionáveis** (computáveis): existe um algoritmo capaz de responder sim, embora, eventualmente, possa ficar processando indefinidamente para uma resposta que deveria ser não;
- E os respectivos complementos **não-solucionáveis** e **completamente insolúveis** (não computáveis).

Um importante resultado deste estudo é que, informalmente, a **Classe dos Problemas Não-Computáveis** é "**muito maior**" que o da **Classe dos Problemas Computáveis**, ou seja, existem muito mais problemas não-computáveis do que computáveis.

9.11 Exercícios

Exercício 9.1
Elabore uma linha de tempo sobre o desenvolvimento do conceito de computabilidade (use como base o correspondente exercício do Capítulo 1 – Instrução).

Exercício 9.2
Defina a relação entre as seguintes classes de problemas:
a) Solucionáveis;
b) Parcialmente Solucionáveis (Computáveis);
c) Não-Solucionáveis;
d) Completamente Insolúveis (Não-Computáveis).

Exercício 9.3
Descreva uma sistemática genérica para traduzir problemas em linguagens.

Exercício 9.4
Qual a importância do Problema da Auto-Aplicação no estudo da solucionabilidade de problemas?

Exercício 9.5
Quais as idéias básicas do princípio da redução?

Exercício 9.6
Desenvolva um algoritmo de decodificação (função inversa da codificação).

Exercício 9.7
Esboce um programa que recebe como entrada p = código_bij(P) e simula P para a entrada p.

Exercício 9.8
Demonstre o **teorema 9.15** Problema da Parada da Palavra Vazia é Não Solucionável. É parcialmente solucionável? E o seu complemento, como se classifica?

Exercício 9.9
Demonstre o **Teorema 9.17** Problema da Totalidade é Não-Solucionável. É parcialmente solucionável? E o seu complemento, como se classifica?

Exercício 9.10

Demonstre o **Teorema 9.19** Problema da Equivalência é Não-Solucionável. É parcialmente solucionável? E o seu complemento, como se classifica?

Exercício 9.11

Demonstre que o *Problema da Parada da Palavra Constante* é não-solucionável.

Exercício 9.12

Escolha um dos problemas abaixo e mostre que ele é parcialmente solucionável:

a) Problema da Parada;
b) Problema da Parada da Palavra Vazia;
c) Problema da Totalidade;
d) Problema da Equivalência.

Exercício 9.13

O *Problema da Aceitação da Palavra* é definido como segue: dada uma Máquina Universal M qualquer e uma palavra w qualquer pertencente a Σ^*, M aceita w? Ou seja, investiga-se se uma palavra é aceita por uma Máquina Universal.

a) A linguagem correspondente a esse problema é conhecida como a *Linguagem Universal*. Qual é essa linguagem?
b) Prove que é um problema parcialmente solucionável;
c) Prove que é um problema não-solucionável.

Exercício 9.14

No estudo dos problemas do tipo sim/não, são válidas todas as operações lógicas como e, ou, se-então, negação, etc. Para os itens abaixo, considere as operações e, ou e negação:

a) Interprete o significado dessas operações sobre problemas;
b) Qual o resultado ao aplicar essas operações sobre problemas solucionáveis? Por quê?
c) Idem para não-solucionáveis;
d) Idem para parcialmente solucionáveis.

Exercício 9.15

Suponha M uma Máquina Universal. Seja M' uma Máquina Universal construída a partir de M, mas invertendo-se as condições de ACEITA por REJEITA e vice-versa. Como essa inversão pode ser implementada? A resposta pode ser específica para qualquer dos formalismos estudados.

Exercício 9.16

Seja M uma Máquina Universal não-determinística definida conforme esquema ilustrado na **figura 9.19**. Como seria a definição de M? A resposta pode ser específica para qualquer dos formalismo estudados.

Exercício 9.17

Sobre o Problema da Correspondência de Post:

a) Encontre a menor solução para os seguintes Sistemas de Post:
   S1 = { (b, bbb), (babbb, ba), (ba, a) }
   S2 ={ (ε, a), (a, b), (b, aa), (aa, ab), (ab, ba), (ba, bb), (bb, ε) }
b) Encontre uma solução para o seguinte Sistema de Post:
   S3 = { (aab, a), (ab, abb), (ab, bab), (ba, aab) }

Observação: a menor solução é uma seqüência de 66 índices.

c) Mostre que o seguinte Sistema de Post não tem solução:
   S4 = { (ba, bab), (abb, bb), (bab, abb) }

Exercício 9.18

Suponha S um Sistema de Post com solução. Desenvolva um algoritmo que determine a menor seqüência de índices que seja solução de S.

Exercício 9.19

Marque a alternativa correta:

a) Um problema não-solucionável é aquele que não pode ser computável; (errado, pois o problema não-solucionável pode ser parcialmente solucionável, ou seja, pode ser computável)
b) Um problema é computável se existe um programa, em uma máquina de Turing, que para qualquer entrada. A computação é sempre finita, pois sempre pára;
c) Um problema parcialmente solucionável é aquele que tem partes que não se consegue resolver; (errado, pois não é totalmente solucionável)
d) Um problema é dito parcialmente solucionável se existe um programa em uma máquina de Turing que solucione o problema tal que pára quando a resposta esperada é negativa (rejeitada);
e) Um problema é computável quando ele é parcialmente solucionável.

Exercício 9.20

Considere o seguinte:

- "O problema A é solucionável";
- "O problema B é parcialmente solucionável. O complemento de B é A. O problema C pode ser reduzido ao problema B".

O que pode dizer sobre C? Marque a alternativa correta:

a) É solucionável; (pois B é solucionável e B é um pedaço de C)
b) É não-solucionável;
c) É parcialmente solucionável, mas não solucionável;
d) Seu complemento é não-solucionável;
e) Não faz parte do complemento de A.

Exercício 9.21

Numere a segunda coluna de acordo com a primeira:

I. Classe dos Problemas Solucionáveis	( ) Cardinal não-contável
II. Classe dos Problemas Parcialmente Solucionáveis	( ) Linguagens recursivas
III. Classe dos Problemas Completamente Insolúveis	( ) Linguagens enumeráveis recursivamente

A numeração correta, de cima para baixo, é:

a) I, II, III
b) III, II, I
c) III, I, II
d) II, I, III
e) II, III, I

Exercício 9.22

Analise as seguintes afirmações e marque cada uma com V (verdadeira) ou F (falsa):

( ) A união da Classes dos Problemas Solucionáveis com o complemento da Classe dos Problemas Completamente Insolúveis é o universo de todos os problemas; F, pois os insolúveis não englobam os problemas não - solucionáveis que são parcialmente solucionáveis

( ) A união das Classes dos Problemas Solucionáveis, dos Problemas Computáveis, dos Problemas Parcialmente Solucionáveis com os Problemas Não-Computáveis é o universo de todos os problemas; V

( ) Todo problema solucionável é parcialmente solucionável; V, trivial

( ) Não existem problemas não-solucionáveis que possuam solução parcial. F, pois existem problemas não-solucionáveis com solução parcial

A marcação correta, de cima para baixo, é:

a) V, V, V, F
b) F, V, V, F
c) V, F, F, V
d) F, V, F, F
e) F, F, V, V

Exercício 9.23

Analise as seguintes afirmações:

I. A Classe dos Problemas Solucionáveis é equivalente à Classe das Linguagens Recursivas; V, pois os problemas solucionáveis são bem definidos e não ficam em loop, ou seja, seu loop é vazio, linguagens recursivas.

II. Existe um algoritmo genérico que sempre pára capaz de comparar quaisquer dois reconhecedores de linguagens livres do contexto, como Pascal, e verificar se são equivalentes, ou seja, se de fato, reconhecem a mesma linguagem; F, problema da equivalência, não-computável

III. Dados um programa e uma entrada quaisquer, não existe algoritmo genérico capaz de verificar se o programa vai parar ou não para a entrada. V, pois para qualquer programa e para qualquer entrada não existe algoritmo genérico que verifique se o programa vai parar sempre.

Marque a alternativa correta:

a) Apenas a I está correta;
b) Apenas a II está correta;
c) Apenas a III está correta
d) Apenas I e III estão corretas
e) I, II e III estão corretas.

Referência de problemas Insolúveis:

http://www.amazon.com/Algorithmics-Spirit-Computing-David-Harel/dp/0321117840#reader_0321117840