Indecidibilidade

Indecidibilidade

- Máquinas de Turing
  - Podem simular computadores reais
- Problemas
  - aqueles que tem um algoritmo
    - Máquina de Turing pára, quer aceite ou não sua entrada
  - aqueles que não tem um algoritmo
    - Máquinas de Turing que podem funcionar indefinidamente, sobre entradas que não aceitam

Indecidibilidade

- Linguagens RE:
  - Recursivamente Enumerável
  - Se L = L(M) para alguma TM M

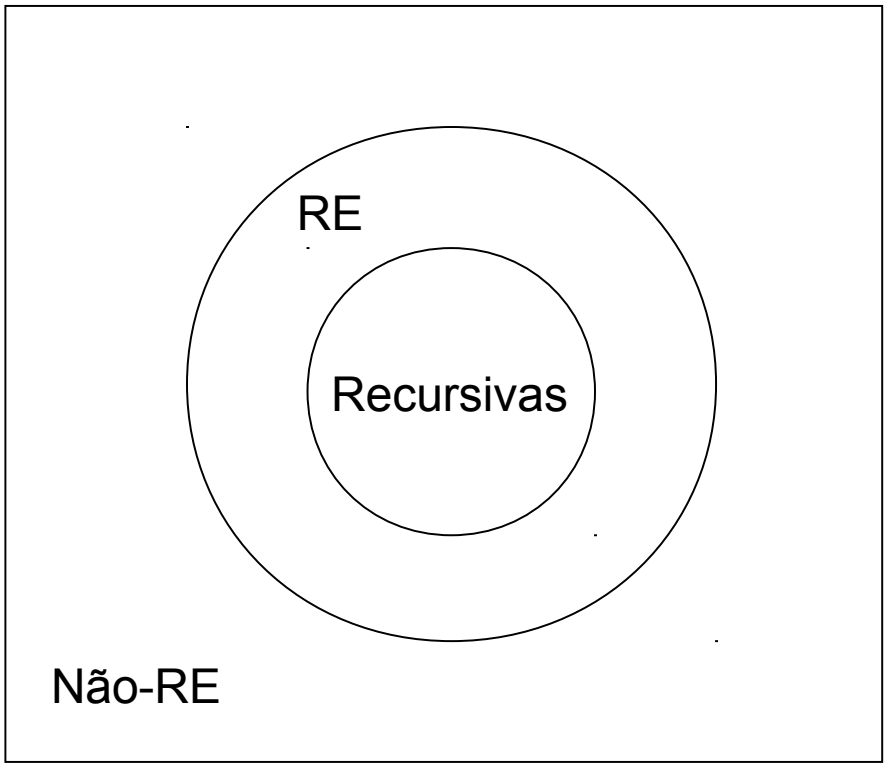
  - Conjunto de linguagens que podemos aceitar usando uma máquina de Turing

Indecidibilidade

- Linguagens Recursivas
  - Dizemos que L é recursiva se L = L(M) para uma máquina de Turing M:
    - Se w está em L, então M a aceita
      - E portanto para
    - Se w não está em L, então M pára eventualmente
      - Embora nunca entre em um estado de aceitação
  - Noção informal de um “algoritmo” que sempre termina e produz uma resposta

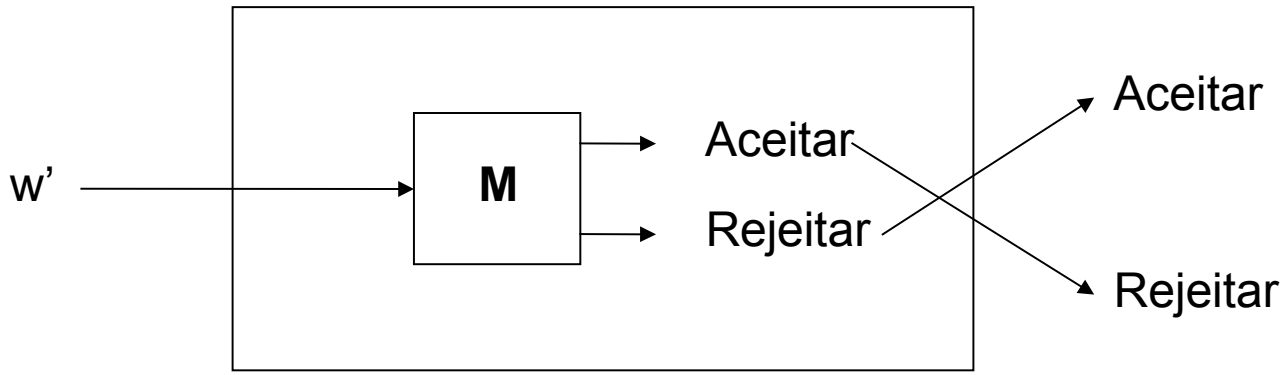
Indecidibilidade

- Linguagens:



Complemento

- Seja L = L(M) para alguma TM M
  - Definimos L’ como o conjunto de palavras não pertencente a L, do mesmo alfabeto
  - Assim, construímos M’, tal que L’ = L(M’)



Complemento

- 4 possibilidades:
  - L e L' são ambas recursivas
  - Nem L, nem L' são RE
  - L é RE mas não-recursiva e L' não é RE
  - L' é RE mas não-recursiva e L não é RE

Indecidibilidade

- Tese de Church
  - estabelece uma correspondência entre as noções de Algoritmo e Máquina de Turing
  - Ou seja, podemos pensar num algoritmo como uma máquina de Turing que sempre pára, para qualquer entrada, aceitando ou rejeitando.

  - Decidível → Recursivo

Exemplos

- Linguagem Ld (diagonalização)
  - Conjunto de strings de 0's e 1's
  - Quando interpretados como uma TM, não estão na linguagem dessa TM
- Teorema:
  - Ld não é uma Linguagem Recursivamente Enumerável

Codificação para TM

- Seja $M = (Q, \Sigma, \delta, q_0, ß, F)$, precisamos atribuir números naturais aos estados, aos símbolos e aos sentidos (E e D)
- podemos codificar o função de transição

$\delta(q_i, X_j) = (q_k, X_l, D_m)$, para naturais i, j, k, l e m.

Codificaremos essa regra pelo string

$$0^i 1 0^j 1 0^k 1 0^l 1 0^m$$

Obs.: Como todos os valores i, j, k, l e m são, pelo menos, iguais a 1, não temos dois, ou mais, 1´s consecutivos.

Codificação para TM

- Um código para a MT M inteira consiste em todos os códigos para as transições, em alguma ordem, separados por pares de 1´s:

$$C_1 11 C_2 11 \ldots C_{n-1} 11 C_n$$

Linguagem Ld

- Agora podemos ordenar todas as MT´s de acordo com o comprimento e as de mesmo comprimento por ordem lexicográfica.
- A linguagem da diagonalização, é o conjunto de strings w_i tais que w_i não está em $L(M_i)$.
- Teorema:
  - Ld não é uma Linguagem Recursivamente Enumerável

Linguagem Ld

- PROVA: Suponha que Ld seja T(M) para alguma máquina de Turing M.

Como Ld é uma linguagem sobre o alfabeto {0, 1}, M estaria na lista das máquinas de Turing, já que esta lista inclui todas as máquinas de Turing com alfabeto de entrada {0, 1}.

Portanto, há pelo menos um código para M, por exemplo, M = Mi. Agora será que wi está em Ld?

Linguagem Ld

- Se wi está em Ld, então Mi aceita wi . Mas então, pela definição de Ld, wi não está em Ld, porque Ld contém apenas aqueles wj tal que Mj não aceita wj.
- Similarmente, se wi não está em Ld, então Mi não aceita wi. Portanto, por definição de Ld, wi está em Ld.

– Como wi não pode estar e não estar em Ld ao mesmo tempo, há uma contradição na suposição de que M existe. Ou seja, Ld não é uma linguagem recursivamente enumerável.

Exemplos

- Linguagem Lu (universal)
  - Consiste em strings que são interpretados:
    - Como uma TM
    - Seguido por uma entrada
  - O string está em Lu se a TM aceita essa entrada
- Teorema:
  - Linguagem Lu é RE mas não recursiva.

Exemplos

- Prova:
  - Linguagem Lu é RE, pois existe uma TM M tal que Lu = L(M)
  - Será recursiva somente se existir uma TM M' para aceitar Lu'
    - Representa o mesmo que criar uma TM para aceitar uma linguagem Ld
    - Ld é não RE
  - Portanto Lu não é recursiva