Computabilidade e Decidibilidade

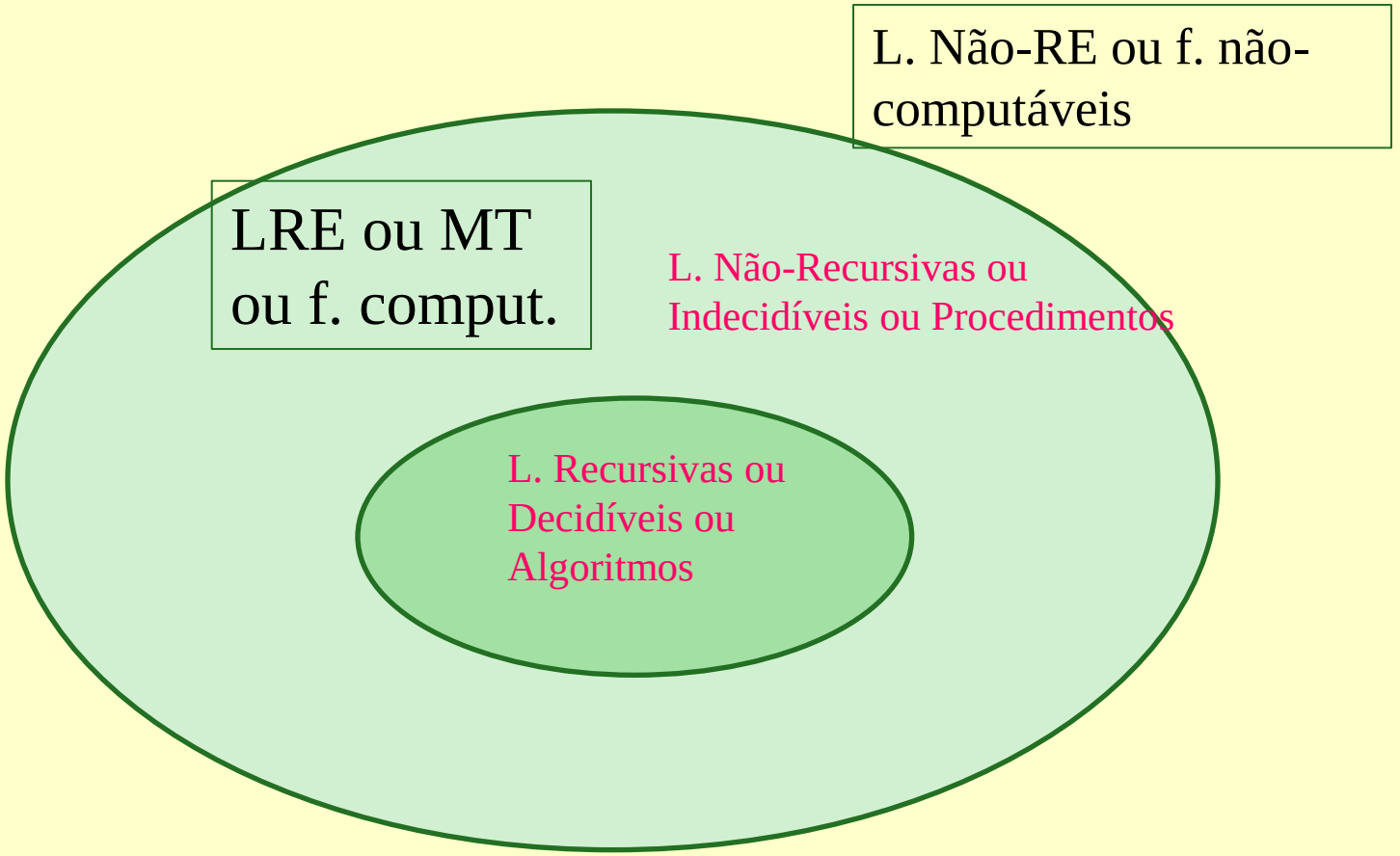
Problemas Computáveis

- Máquinas de Turing ou Funções Computáveis ou Linguagens Recursivamente Enumeráveis ***LRE*** podem ser divididas em 2 classes:

  (1) as MT que, para qualquer cadeia de entrada, sempre terminam, ou seja, sempre respondem se a cadeia faz parte ou não da linguagem. Em outras palavras, **decidem** a linguagem. Essas linguagens são chamadas Linguagens *Recursivas ou Decidíveis,* e essas MT correspondem aos **Algoritmos**.

  (2) as MT que, para qualquer cadeia de entrada, terminam aceitando a cadeia, se ela fizer parte da linguagem, ou podem funcionar indefinidamente sobre entradas que elas não aceitam. Em outras palavras, **aceitam** a linguagem. Tais linguagens são chamadas *Linguagens Indecidíveis,* e essas MT correspondem aos **Procedimentos**.

- Problemas ou Linguagens *Indecidíveis* são aqueles para os quais não existe nenhum *algoritmo*, ou seja, uma MT que sempre pára.



Pergunta

- O que caracteriza as funções indecidíveis (para as quais há procedimento, mas não algoritmo)?

Ou

- Que tipo de propriedade (característica da linguagem) pode ser decidida ou não?

Exemplos clássicos de funções indecidíveis

Problema 1.: Existe um procedimento – na verdade, um algoritmo - (p.ex., em Pascal) que toma como entrada um outro procedimento qualquer, *p*, e retorna *true* se *p* é um algoritmo, ou *false*, caso contrário?

Resposta: Não!

Prova: Por contradição

Suponha que tal procedimento exista. Vamos chamá-lo de ALG. Então a declaração de ALG é da forma:

function ALG (procedure p) : boolean;

<corpo da função>

Podemos, então, usar a função ALG para definir novos procedimentos:

```
procedure Problema (x: integer);
begin
        while ALG(Problema) do nil
end;
```

Pergunta: o procedimento "***Problema***" **é algoritmo?**

→ Suponha que **sim**. Então *ALG(Problema)* é *true* e o comando *while* nunca termina, e portanto, *Problema* nunca termina, e **não é algoritmo**. Contradição!

→ Suponha que **não**. Então *ALG(Problema)* é *false* e o comando *while* termina, e portanto, *Problema* termina, e **é um algoritmo**. Contradição!

→Portanto, *Problema termina se Problema não termina*

Logo, *ALG* não pode existir.

Problema 2. (da Parada): Existe um procedimento – na verdade, um algoritmo - HALT, que toma como entrada um procedimento *p* e um inteiro x, e retorna *true* se *p* termina com entrada x e *false*, se *p* não termina com entrada x?

Resposta: Não!

Prova: Por contradição.

Suponha que HALT exista. Então podemos escrever um procedimento Pascal D:

```
procedure D (x: integer);
begin
        while HALT(D, x) do nil
end;
```

Pergunta: D termina com entrada x?

→ Suponha que **sim.** Então *HALT(D, x)* é *true* e D **não termina com entrada x.** Contradição!

→ Suponha que **não**. Então *HALT(D, x)* é *false* e D **termina com entrada x**. Contradição!

Portanto, *D termina com entrada x se D não termina com entrada x.*

Logo, HALT não pode existir!

Propriedades Indecidíveis

- Logo, as propriedades de procedimentos:
  - É algoritmo ou não (Problema 1)
  - Termina para uma entrada x (Problema 2)

são indecidíveis – ou seja, não há algoritmos que as decidam.

Propriedades Indecidíveis

Corolário: Se uma propriedade **P é indecidível**, então a negação desta propriedade, ¬**P, também é indecidível.**

Se queremos verificar ¬P num procedimento A, temos que decidir P executando A, e quando A retorna *true*, a saída é *false*, e quando A retorna *false*, a saída é *true*.

Daí, as propriedades:

"não termina para alguma entrada" e "não termina para entrada x" são ambas **indecidíveis**.

Propriedades Semi-Decidíveis

Um atributo menos rigoroso de propriedades de procedimentos é introduzido por:

DEF.: Uma propriedade de procedimento P é dita **semi-decidível** se existe um procedimento que, quando dado um procedimento *p*, resulta *true*, se *p* tem a propriedade P. (nada se espera se ele não tiver a propriedade P)

Propriedades Semi-Decidíveis

Obs.1:

- A noção de semi-decidibilidade é mais fraca que a de decidibilidade. Se uma propriedade P é **decidível**, então sempre se pode dizer se um procedimento **tem ou não tem** a propriedade P. Já se P é semi-decidível, pode-se dizer apenas se um procedimento tem a propriedade P.

- **Corolário:** Se P é decidível, certamente ela é semi-decidível.

Propriedades Semi-Decidíveis

Teorema: A propriedade de procedimento "*termina para entrada x"* é semi-decidível.

Prova: O procedimento pode ser expresso em Pascal como:

```
function TERM (procedure f): boolean;
begin
        f(x);
        TERM:= true
end;
```

Repare que, se f(x) terminar, TERM também termina; se f(x) não terminar, TERM não termina.

Entretanto, existem muitas propriedades que não são sequer semi-decidíveis.

Resultado: Se P é semi-decidível e $\neg$ P é semi-decidível, então P é decidível.

Prova: Assuma que ambos P e $\neg$ P são semi-decidíveis.

Sejam $\mathbf{p_1}$**:** o procedimento que resulta *true*, se P é decidível;

e $\mathbf{p_2}$**:** o procedimento que resulta *true*, se $\neg$ P é decidível.

Podemos, então, construir um procedimento (algoritmo) *p* que executa ou simula p_1 e p_2 em paralelo e espera que um dos 2 retorne *true*. Desde que P é ou *true* ou *false*, exatamente um dos 2 procedimentos **deve** retornar com o valor *true*.

Se p1 retornar *true* → *p* retorna *true;*

Se p2 retornar *true* → *p* retorna *false*.

Portanto, ***p*** **sempre termina e decide P.**

Este resultado é útil quando queremos mostrar que uma propriedade não é semi-decidível. Por exemplo:

- Sabemos que a propriedade P, *"procedimento p termina para entrada x"*, é semi-decidível.
  Se ¬P, *"procedimento p não termina para entrada x"*, for semi-decidível, então, pelo Resultado anterior, teríamos que a propriedade P, "*procedimento p termina para entrada x"*, é decidível – o que sabemos ser falso.
- Concluímos, então, que ¬P, *"procedimento p não termina para entrada x"*, não é semi-decidível.

Outros problemas indecidíveis

- **Problema da equivalência de programas:** Não existe um algoritmo que decide se dois procedimentos dados P e Q são equivalentes; mais precisamente, não existe um programa Eq(P, Q) tal que Eq pára com quaisquer dados de entrada, e Eq(P, Q) = T se os procedimentos P e Q calculam a mesma função e Eq(P, Q) = F em caso contrário. Note que P e Q calculam a mesma função se para qualquer entrada ou ambos não param, ou ambos param com a mesma resposta.

- **Problema da Satisfatibilidade:** É indecidível se uma expressão lógica, formada com os conectivos e quantificadores lógicos $\neg$, $\wedge$, $\vee$, $\Rightarrow$, $\forall$, $\exists$, é satisfatível, ou seja, tem valor lógico verdadeiro para quaisquer valores de seus símbolos.

- É indecidível se uma expressão formada com os símbolos 0, 1, +, *, =, conectivos lógicos $\neg$, $\wedge$, $\vee$, $\Rightarrow$, variáveis e quantificadores lógicos $\forall$ e $\exists$, é um Teorema da Aritmética.