

Autômatos, Linguagens e Algoritmos

(Versão 24-02-2021)

Prof. Pedro Paulo

pedrob@mackenzie.br

professor.mackenzie.br/pedrob

[Slides adaptados de original do Prof Luciano Silva, 2012]

Problema (I)

Existe um algoritmo para testar se um número inteiro x é par ou não ?

Problema (I)

Existe um algoritmo para decidir se um número inteiro x é par ou não ?

SIM!! e o algoritmo é o seguinte:

Se $x \% 2 = 0$

x é PAR

Senão x é ÍMPAR

Problema (I)

Existe um algoritmo para decidir se um número natural x é par ou não ?

SIM!! e o algoritmo é o seguinte:



PROBLEMA COMPUTÁVEL

Problema (I)

*Existe um algoritmo **eficiente** para testar se um número inteiro x é par ou não ?*

Problema (I)

*Existe um algoritmo **eficiente** para decidir se um número inteiro x é par ou não ?*

SIM!! e o algoritmo é o seguinte:

```
Se x % 2 = 0  
    x é PAR  
Senão x é ÍMPAR
```

$O(|x|)$

Problema (I)

Existe um algoritmo **eficiente** para decidir se um número inteiro x é

PROBLEMA COMPUTÁVEL

SIM!! e o

E
DE COMPLEXIDADE
POLINOMIAL

X É PAR

Senão x é ÍMPAR

Problema (II)

*Existe um algoritmo para decidir se um número inteiro x é composto ou não ? (**Fatoração de x**)*

x é composto \Leftrightarrow existem $y,z > 1$ tais que $x=yz$

Problema (II)

Existe um algoritmo para decidir se um número inteiro x é composto ou não ?

x é composto \Leftrightarrow existem $y, z > 1$ tais que $x=yz$

SIM!! e o algoritmo é o seguinte:

Para $y=2$ até x

 Para $z=2$ até x

 Se $x=y*z$

x é composto. PARE!

x não é composto.

Problema (II)

Existe um algoritmo para decidir se um número natural x é composto ou não ?

x é composto \Leftrightarrow existe $y, z \in \mathbb{N}$ tais que $x=yz$

SIM!! e o algoritmo é o seguinte:

Para $y=2$ até x

Para $z=2$ até y

Se

$x = yz$ entao x é composto. PARE!

x é primo.

PROBLEMA COMPUTÁVEL

Problema (II)

Complexidade do algoritmo clássico:

Comprimento do número a ser fatorado (em bits)	Tempo de fatoração por algoritmo clássico
512	4 dias
1024	100 mil anos
2048	100 mil bilhões de anos
4096	100 bilhões de quatrilhões de anos

Problema (II)

Complexidade do algoritmo clássico

Comprimento número a fatorado (em bits)	Tempo de cálculo (em anos)
512	100 mil bilhões de anos
1024	100 bilhões de quatrilhões de anos
2048	100 bilhões de quatrilhões de anos

Fatoração de grandes números é intratável !

Problema (II)

*Existe um algoritmo **eficiente** para decidir se um número inteiro x é composto ou não ?*

x é composto \Leftrightarrow existem $y, z > 1$ tais que $x=yz$

Problema (II)

*Existe um algoritmo **eficiente** para decidir se um número inteiro x é composto ou não ?*

x é composto \Leftrightarrow existem $y,z > 1$ tais que $x=yz$

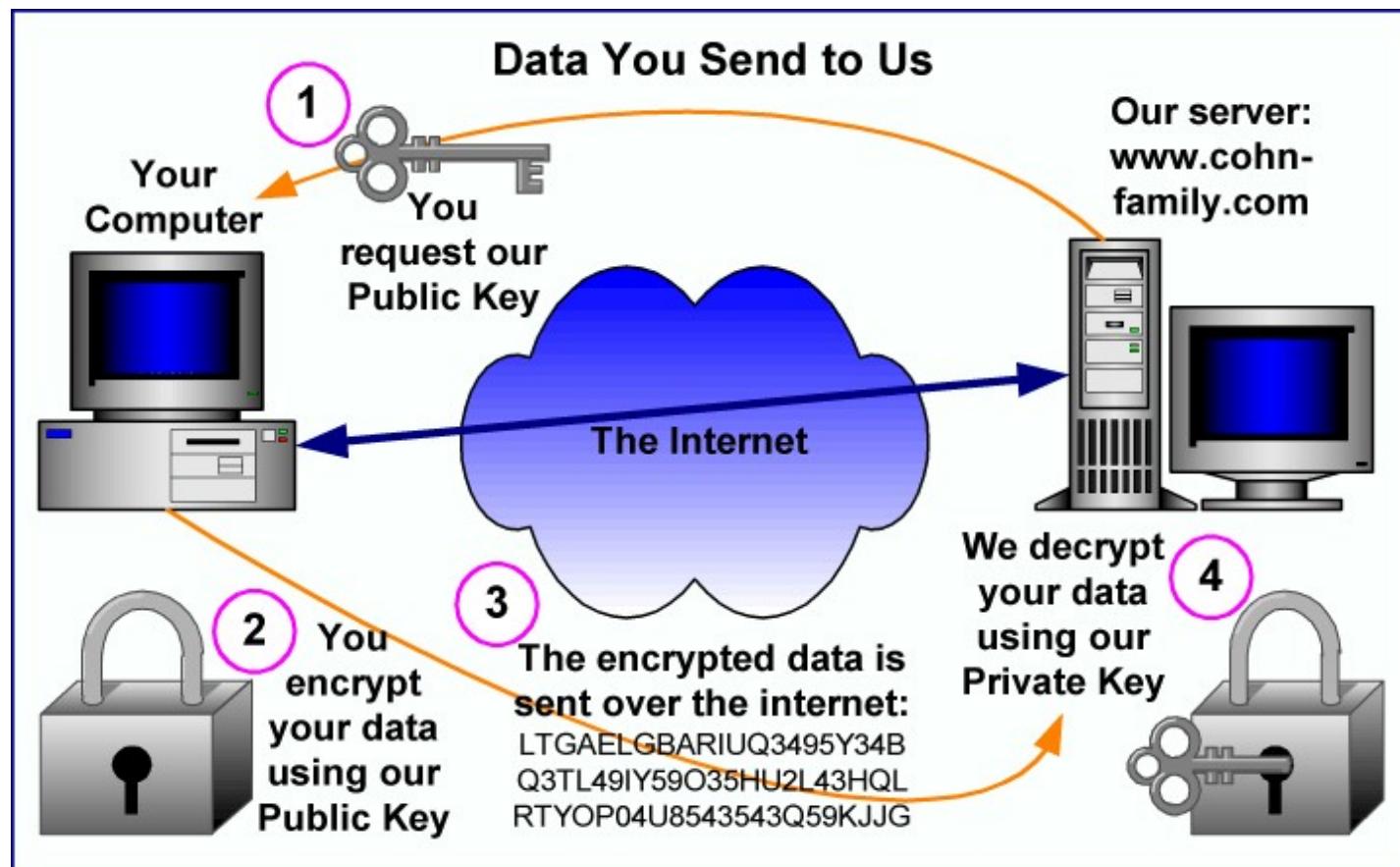
**Não se conhece tal
algoritmo no mundo
clássico!**

Não significa que ele não exista !

Problema (II)

Consequência de não se conhecer tal algoritmo...

Esquema de Criptografia RSA ainda é seguro!



Problema (II)

Oops... Algoritmo Quântico de Shor!

- | | |
|------------------------------------|--|
| 1. initial state | $ 0\rangle 1\rangle$ |
| 2. create superposition | $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} j\rangle 1\rangle$ |
| 3. apply $U_{x,N}$ | $\rightarrow \frac{1}{\sqrt{2^t}} \sum_{j=0}^{2^t-1} j\rangle x^j \bmod N\rangle$
$= \frac{1}{\sqrt{r2^t}} \sum_{s=0}^{r-1} \sum_{j=0}^{2^t-1} e^{2\pi i sj/r} j\rangle u_s\rangle$ |
| 4. apply inverse Fourier transform | $\rightarrow \frac{1}{\sqrt{r}} \sum_{s=0}^{r-1} \tilde{s/r}\rangle u_s\rangle$ |
| 5. measure first register | $\rightarrow s/r \pm \delta$ |
| 6. apply cont. frac. algorithm | $\rightarrow r$ |

Problema (II)

Comprimento do número a ser fatorado (em bits)	Tempo de fatoração por algoritmo clássico	Tempo de fatoração com o algoritmo de Shor
512	4 dias	34 segundos
1024	100 mil anos	4,5 minutos
2048	100 mil bilhões de anos	36 minutos
4096	100 bilhões de quatrilhões de anos	4,8 horas

Problema (II)

Comprimento do número a ser fatorado (em bits)	Tempo de fatoração por algoritmo clássico	Tempo de fatoração com o algoritmo de Shor
512	4 dias	34 segundos
1024	100 mil anos	4,5 minutos
2048	100 mil bilhões de anos	36 minutos
4096	100 bilhões de quatrilhões de anos	4,8 horas

Máquina de Turing

Máquina Quântica

Modelos de Computação

Problema (III)

O Problema da Parada (*Halting Problem*)

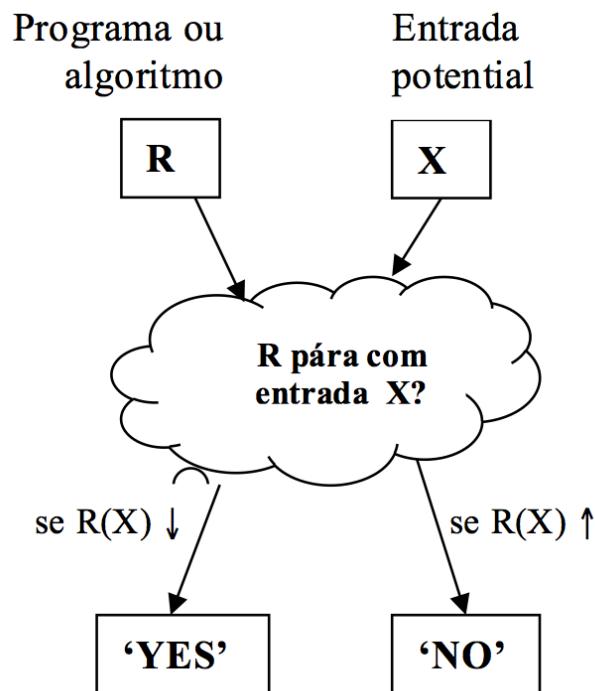
O programa R para com entrada X?

É possível escrever um programa que resolva o problema da parada?

Problema (III)

Problema da Parada

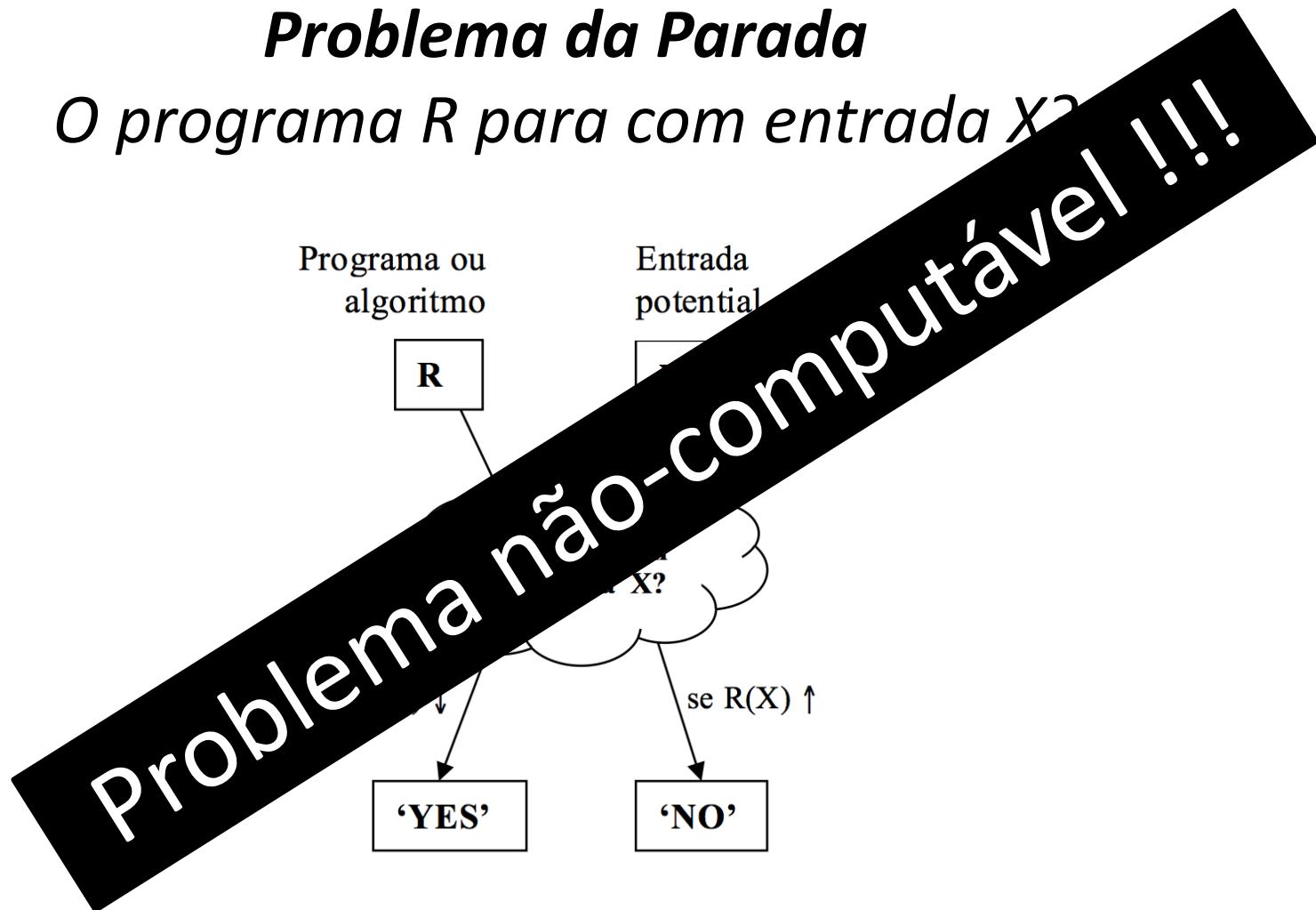
O programa R para com entrada X?



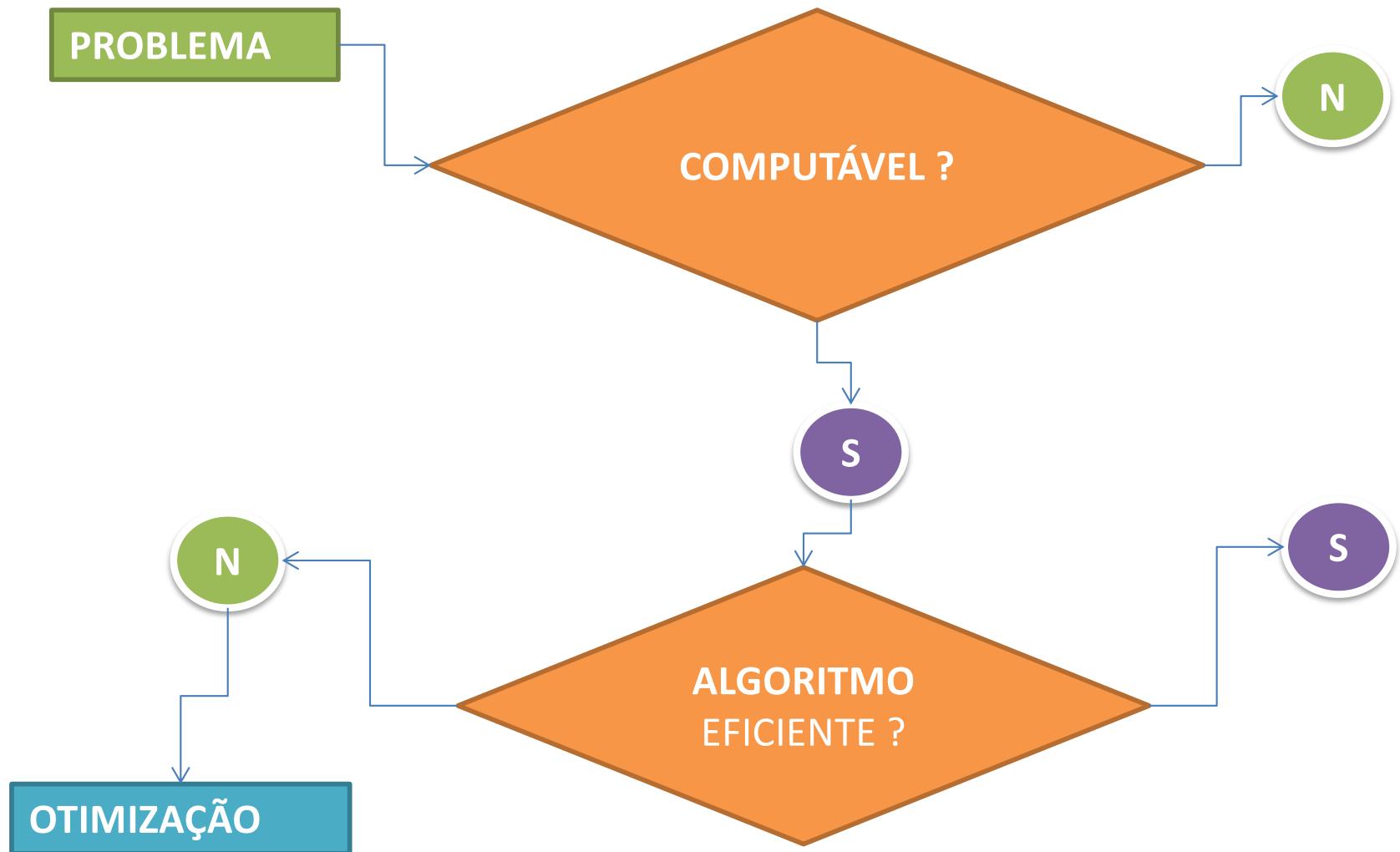
Problema (III)

Problema da Parada

O programa R para com entrada X?



Ambiente da Teoria da Computação



TEORIA DA COMPUTAÇÃO

COMPUTABILIDADE

O que é possível computar?

COMPLEXIDADE

Com que eficiência de tempo e espaço se consegue computar?

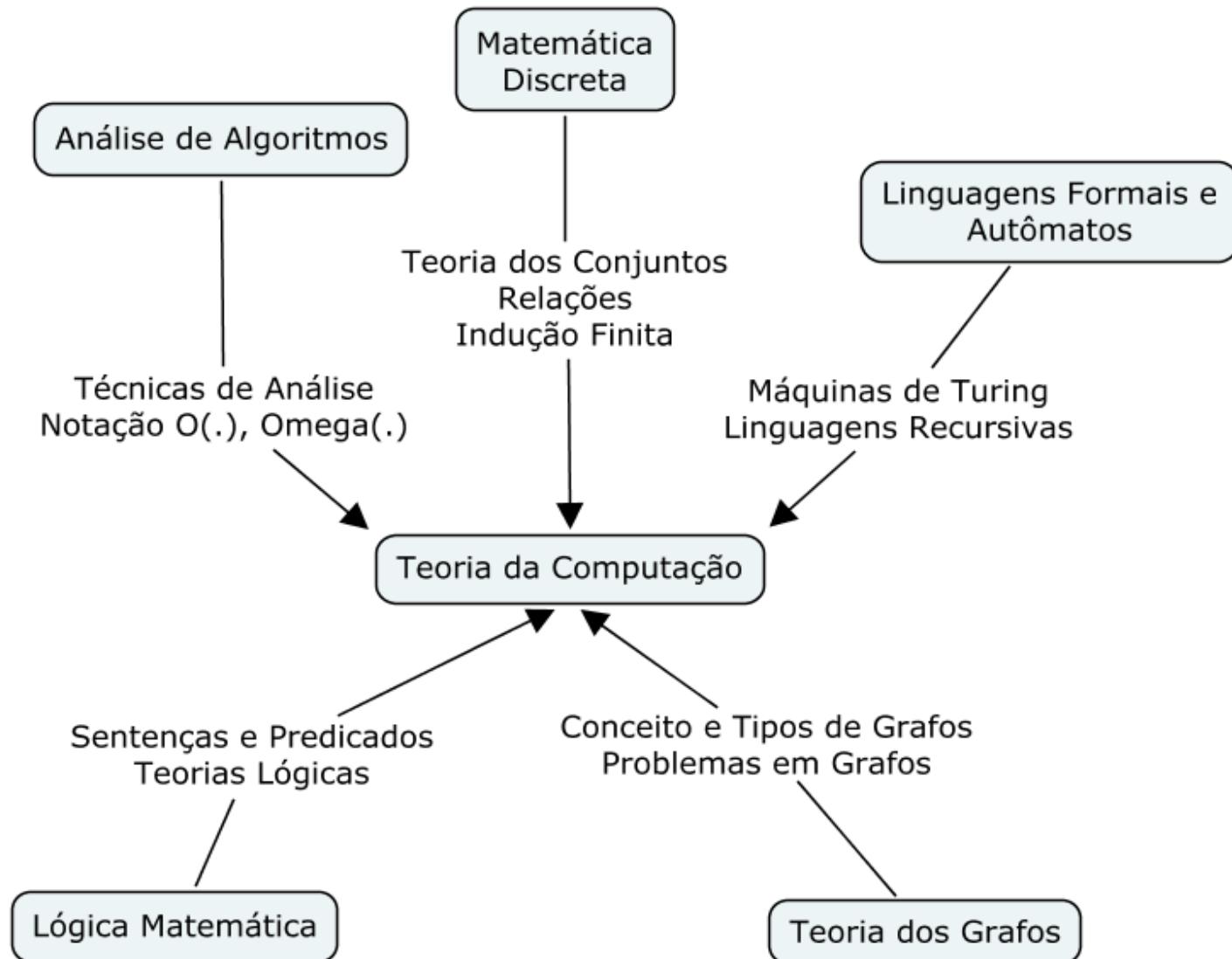
LINGUAGENS FORMAIS

Caracterização de capacidades computacionais

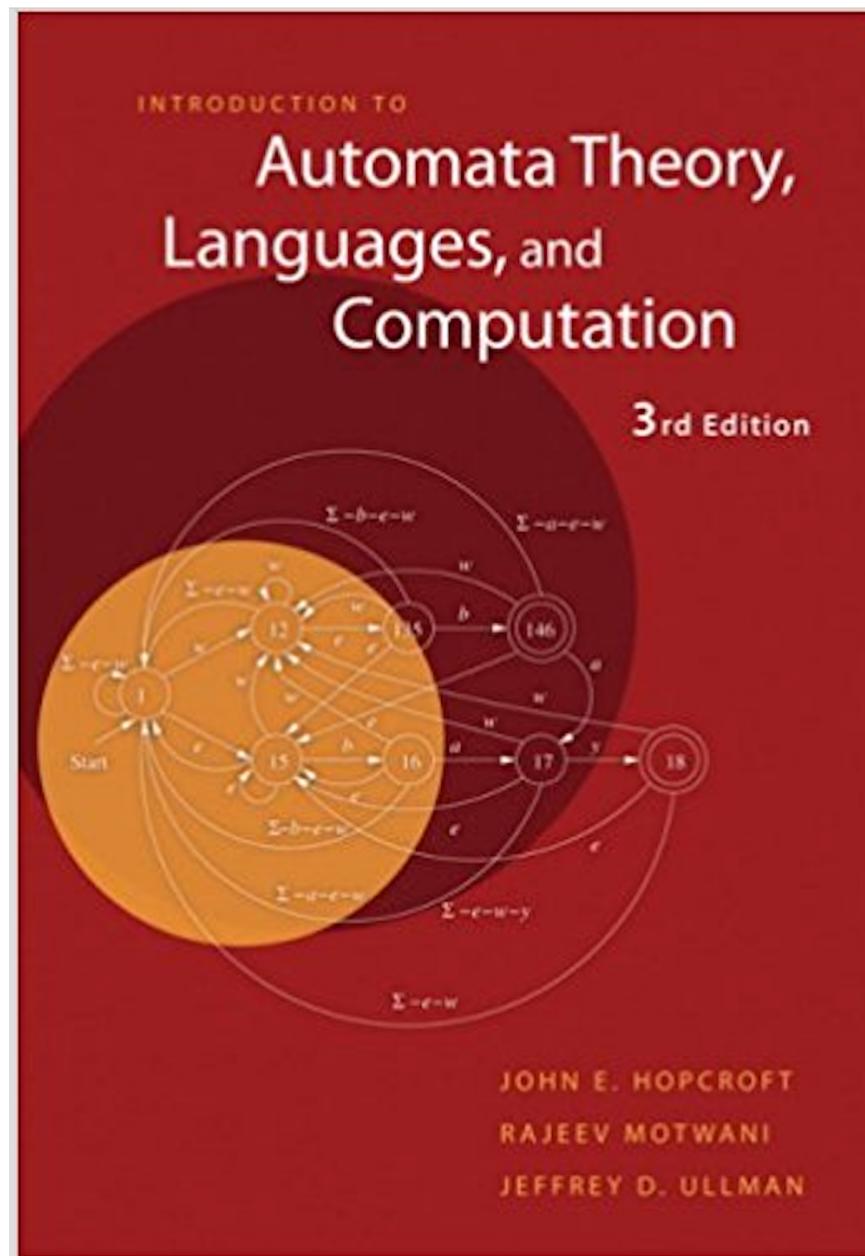
Objetivos da Disciplina

- Formalizar as noções de **problema** e **algoritmo**
- Classificar problemas quanto à **computabilidade**
- Classificar problemas quanto à **complexidade**

Relacionamento entre Disciplinas



Bibliografia



INTRODUCTION
TO
AUTOMATA THEORY,
LANGUAGES,
AND
COMPUTATION

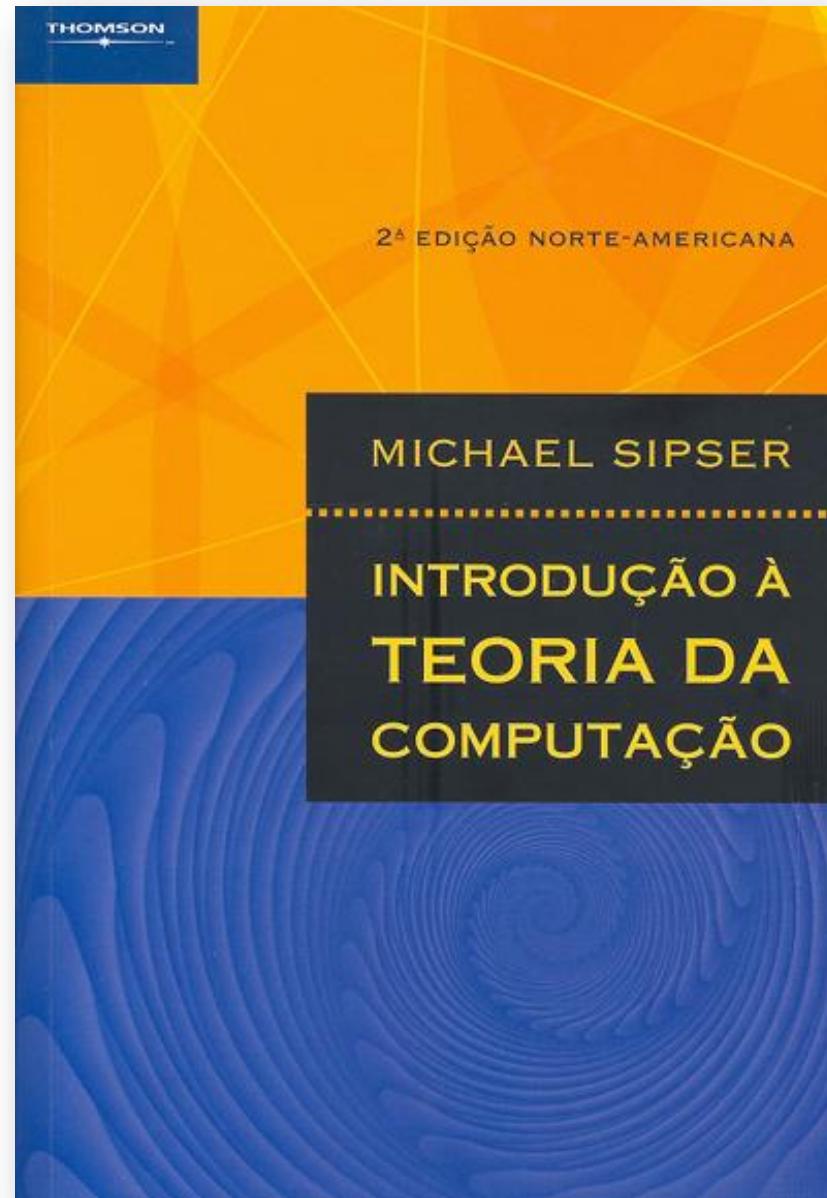
JOHN E. HOPCROFT
JEFFREY D. ULLMAN



Introduction to the Theory of
COMPUTATION
THIRD EDITION



MICHAEL SIPSER



Algorithmics

The Spirit of Computing

THIRD EDITION

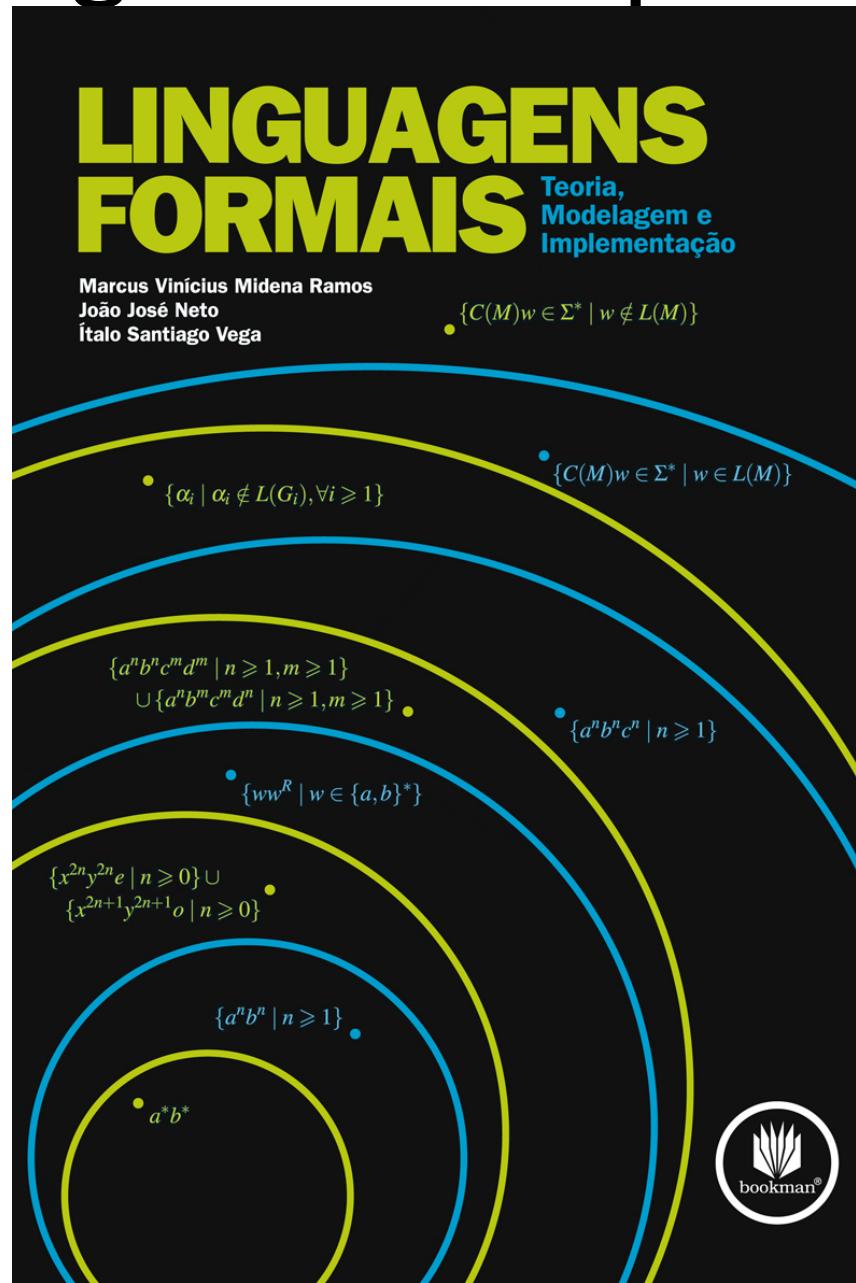
David Harel

with

Yishai Feldman



Bibliografia Complementar



5

** Série livros didáticos informática ufrgs **

teoria da computação:
máquinas universais
e computabilidade

** 3^a edição

• tiarajú asmuz diverio
• paulo blauth menezes



ROBERT W. FLOYD

RICHARD BEIGEL

THE LANGUAGE OF MACHINES

An Introduction to
Computability and
Formal Languages



