Resumindo e Conectando a Linguagens, Máquinas à Complexidade Computacional (em Figuras)

Claudio Cesar de Sá
claudio.sa@udesc.br

Departamento de Ciência da Computação
Centro de Ciências e Tecnológias
Universidade do Estado de Santa Catarina

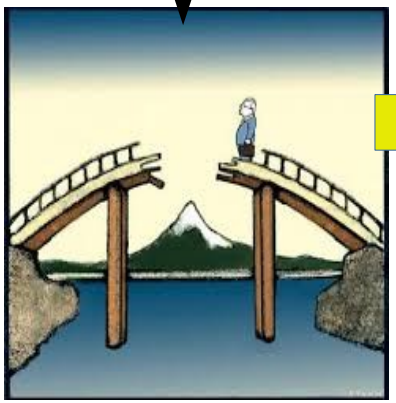
16 de dezembro de 2018

Contextualizando

- Algumas figuras vieram de vários autores *by Google*
- Propositalmente, as mantive originais!
- Cabeçalho e sequência dos *slides* – minha responsabilidade
- Comentários e ajuda: Cristiano Damiani
- Requisitos: finalizando um curso de LFA ou de TEC (preferenciamente)
- Esta apresentação disponível em **.tex** e **.pdf** em: `github.com/claudiosa/CCS/tree/master/linguagens_formais_LFA/extras_lfa_CCS`
- Programa de gravação utilizado: `http://www.maartenbaert.be/simplescreenrecorder/`
- Plataforma: Manjaro – Linux

Resolvendo Problemas

Validação dos modelos



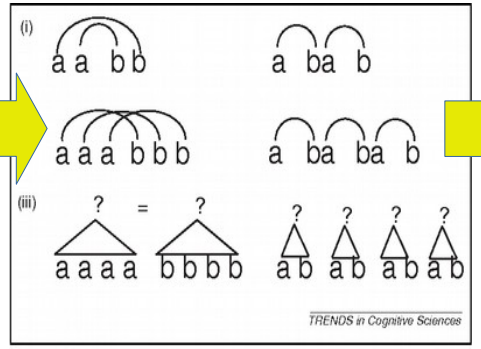


Figure 1. Strategies to check whether sentences are members of the formal languages $a^n b^n$ and $(ab)^n$ (see text for details).

Problema real

Visão conceitual do problema – modelo conceitual

Modelo Formal: alfabetos, palavras bem formadas, e teoremas! Uma teoria!

Implementação em uma Linguagem de Programação ou em Hardware

Figura: O quê o cientista busca?

O *Dilema* dos problemas é:

Problemas × Complexidade

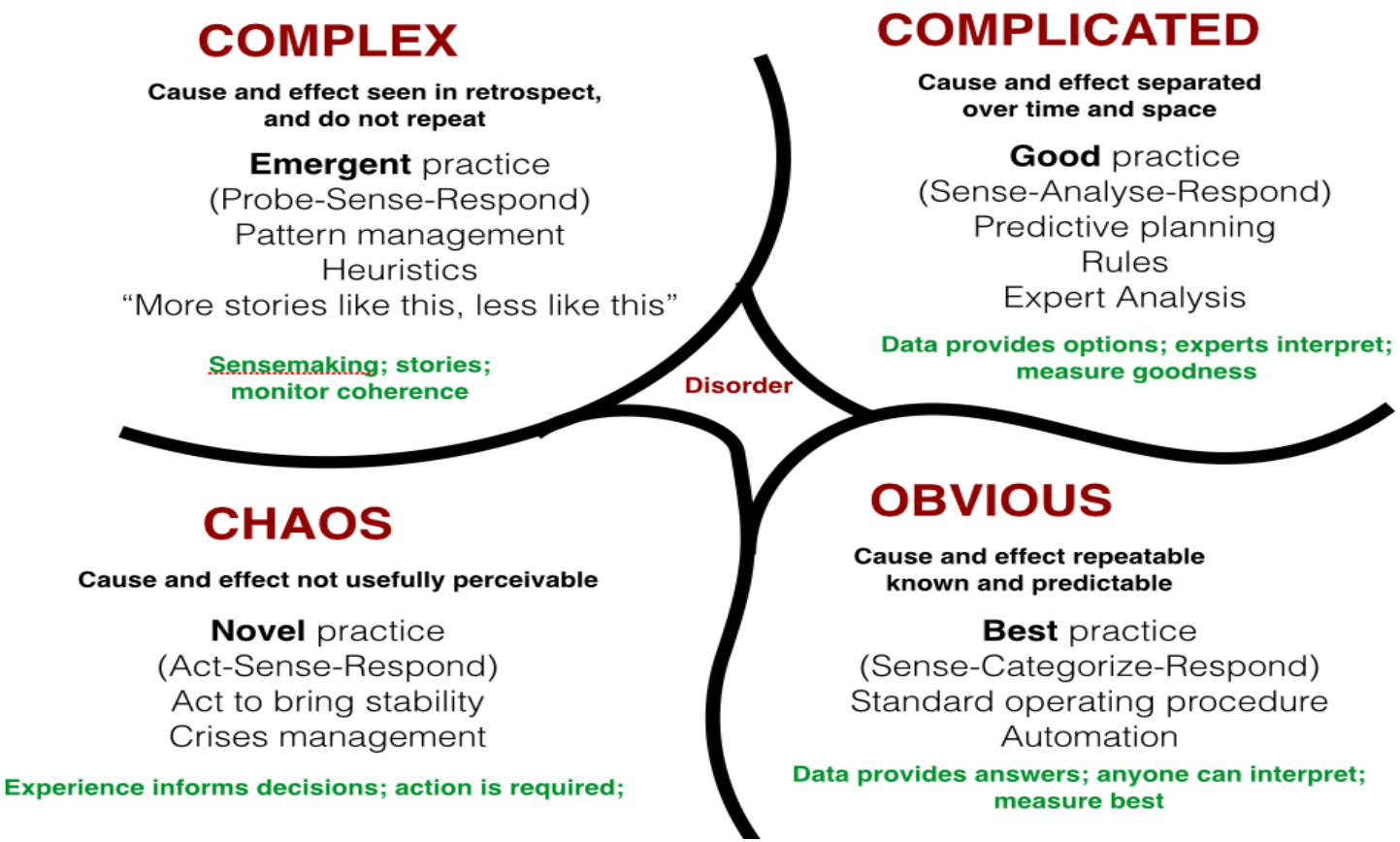


Figura: A área de Ciência da Computação (CC) se preocupa com os dificeis (complicados) e os simples (os ingênuos) – *alguma estruturação!*

A área da CC se preocupa com:

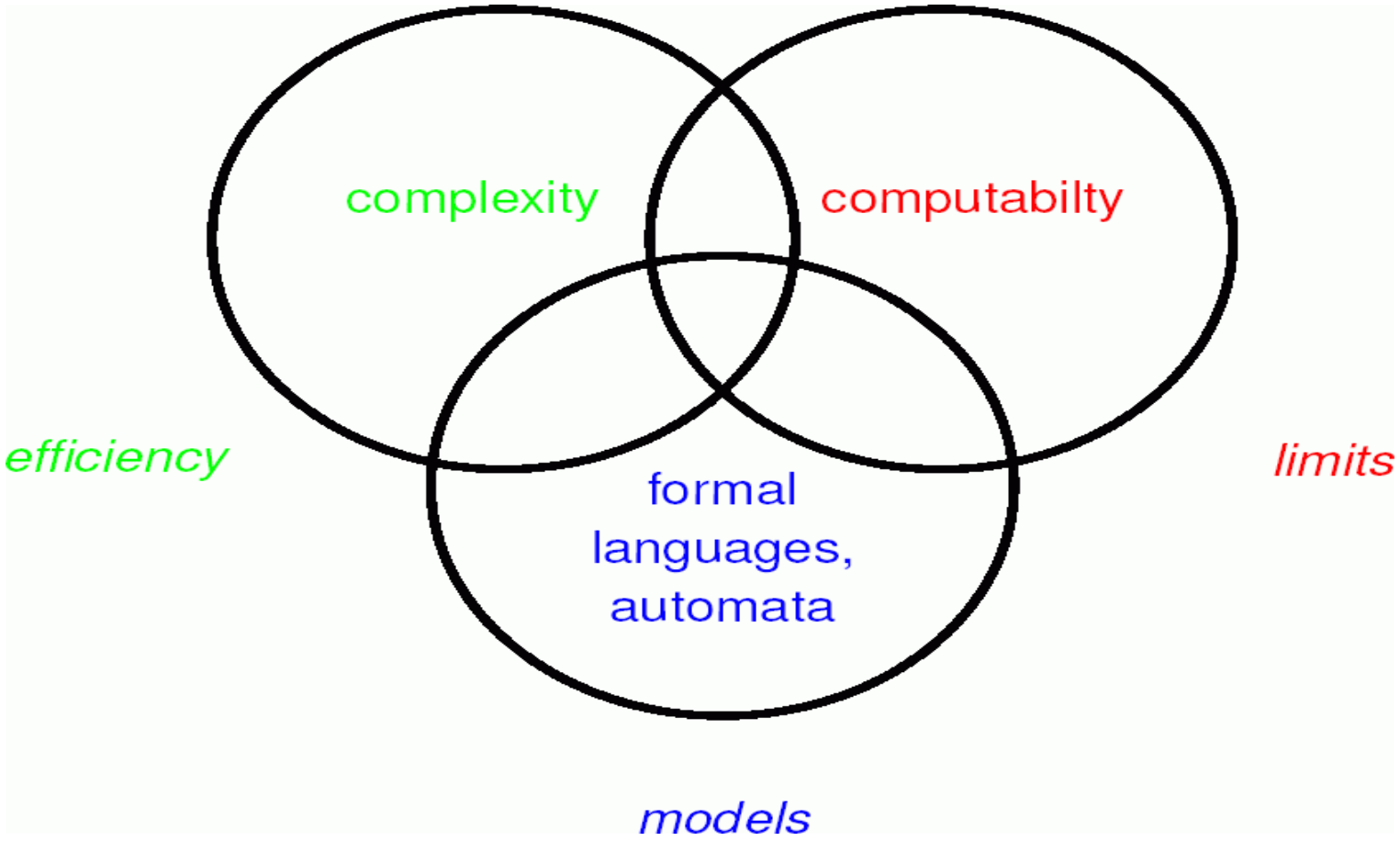


Figura: Precisamos construir modelos e medir seu desempenho!

Um escopo de formalismos da CC:

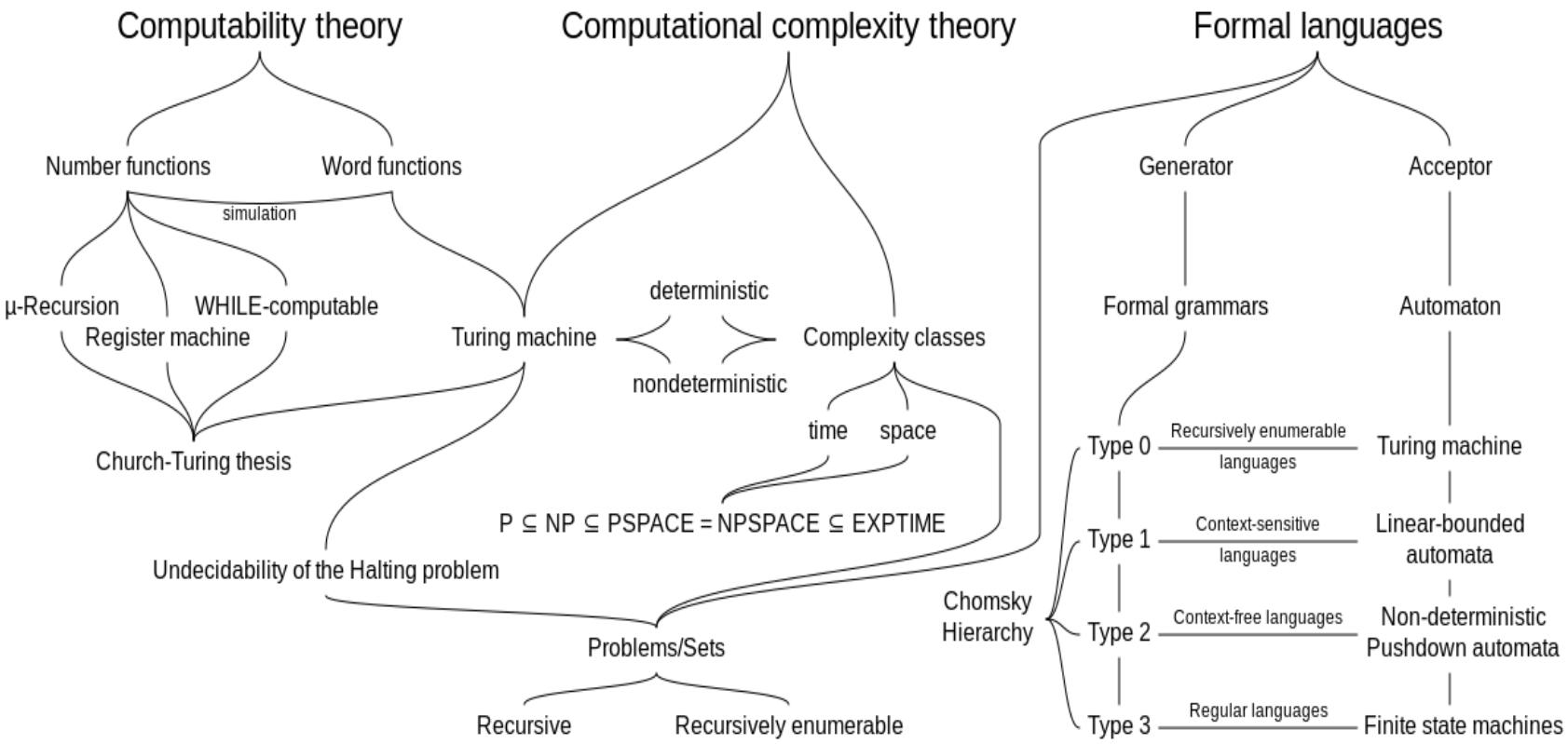
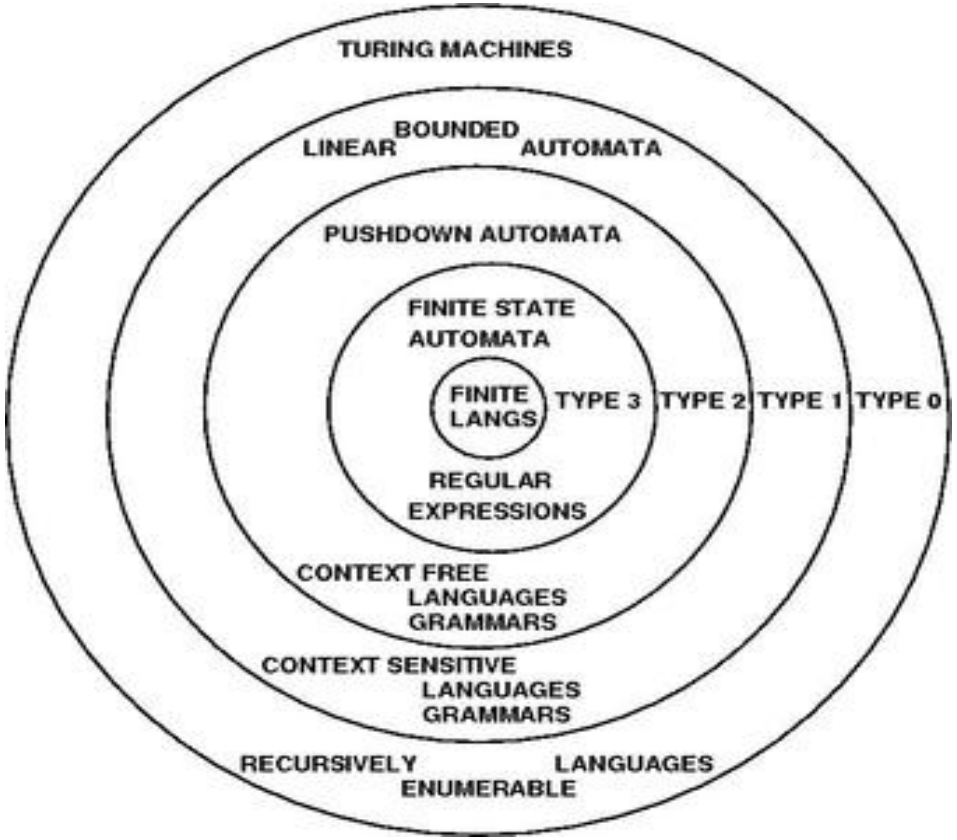
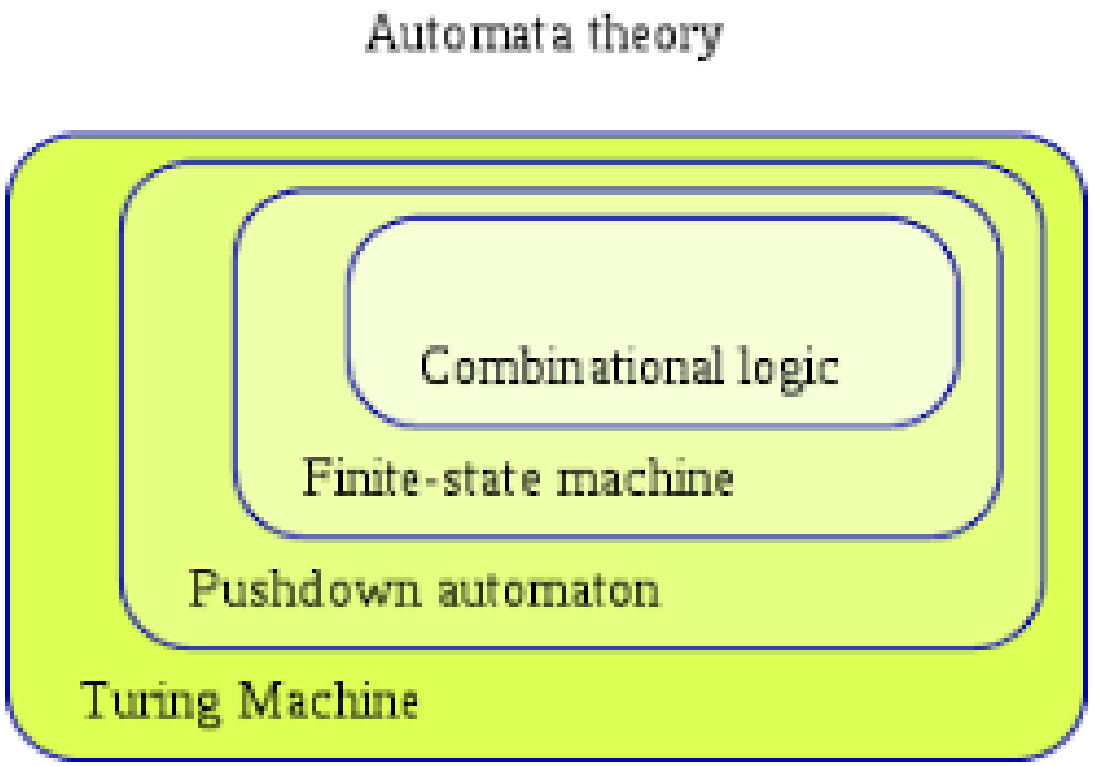


Figura: Basicamente um guia desta apresentação

LFA – Linguagens Formais



Máquinas que calculam estas linguagens:



Automatos de Estados Finitos

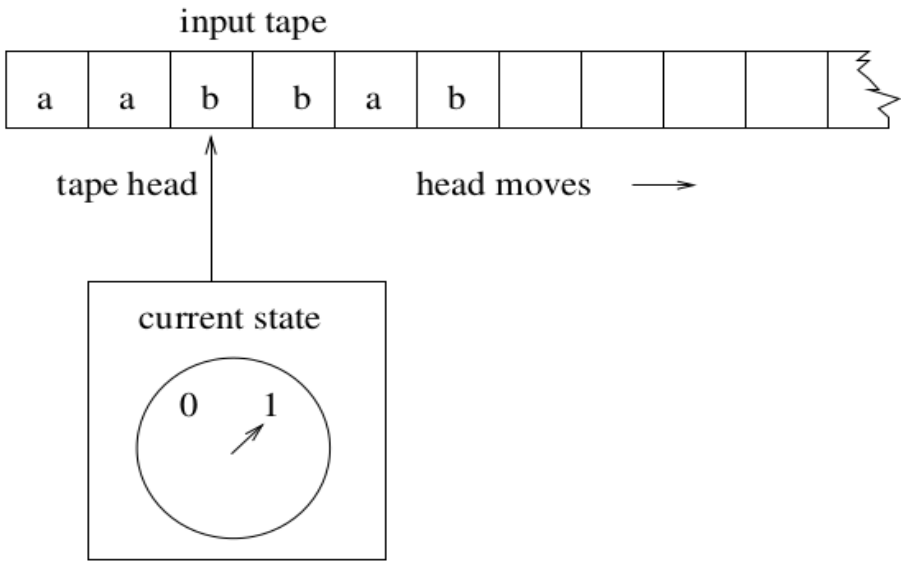


Figura: Uma unidade de controle com estados finitos

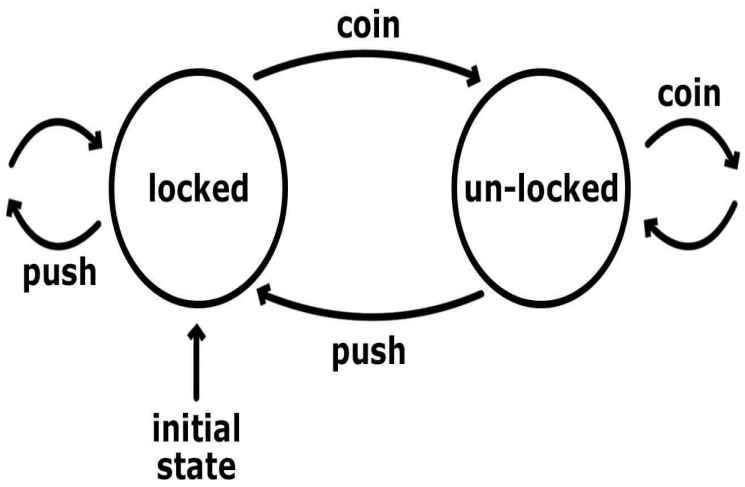


Figura: Exemplo: maleiro e moedas

Exemplo

Figura: Problema do Pastor

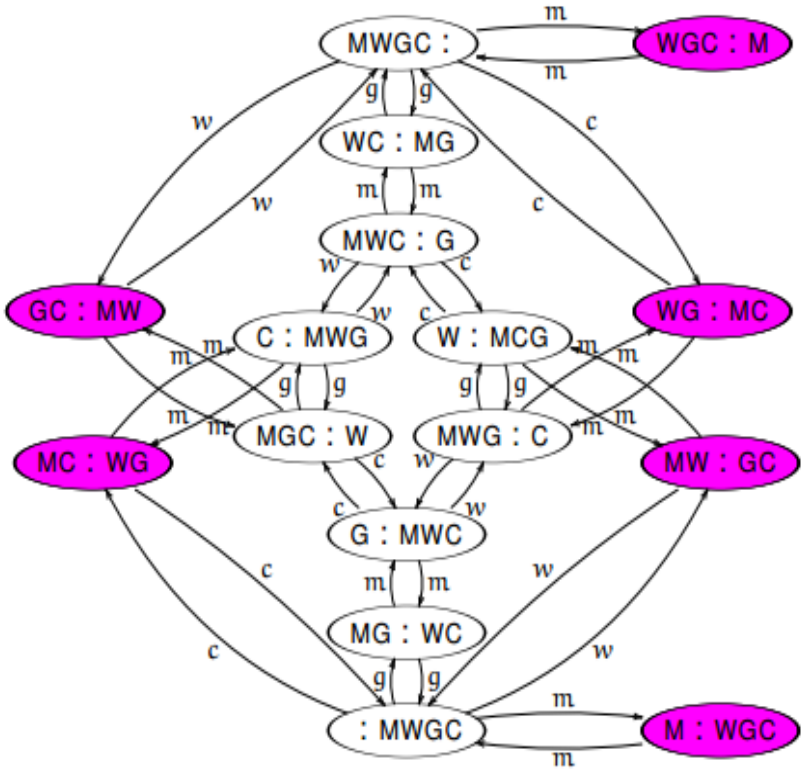
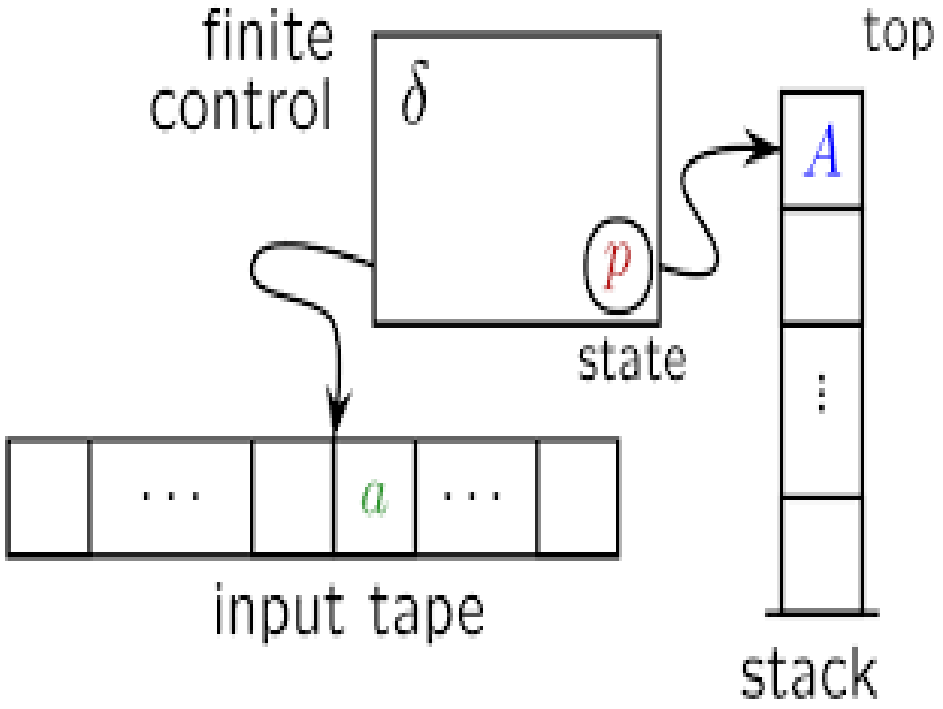
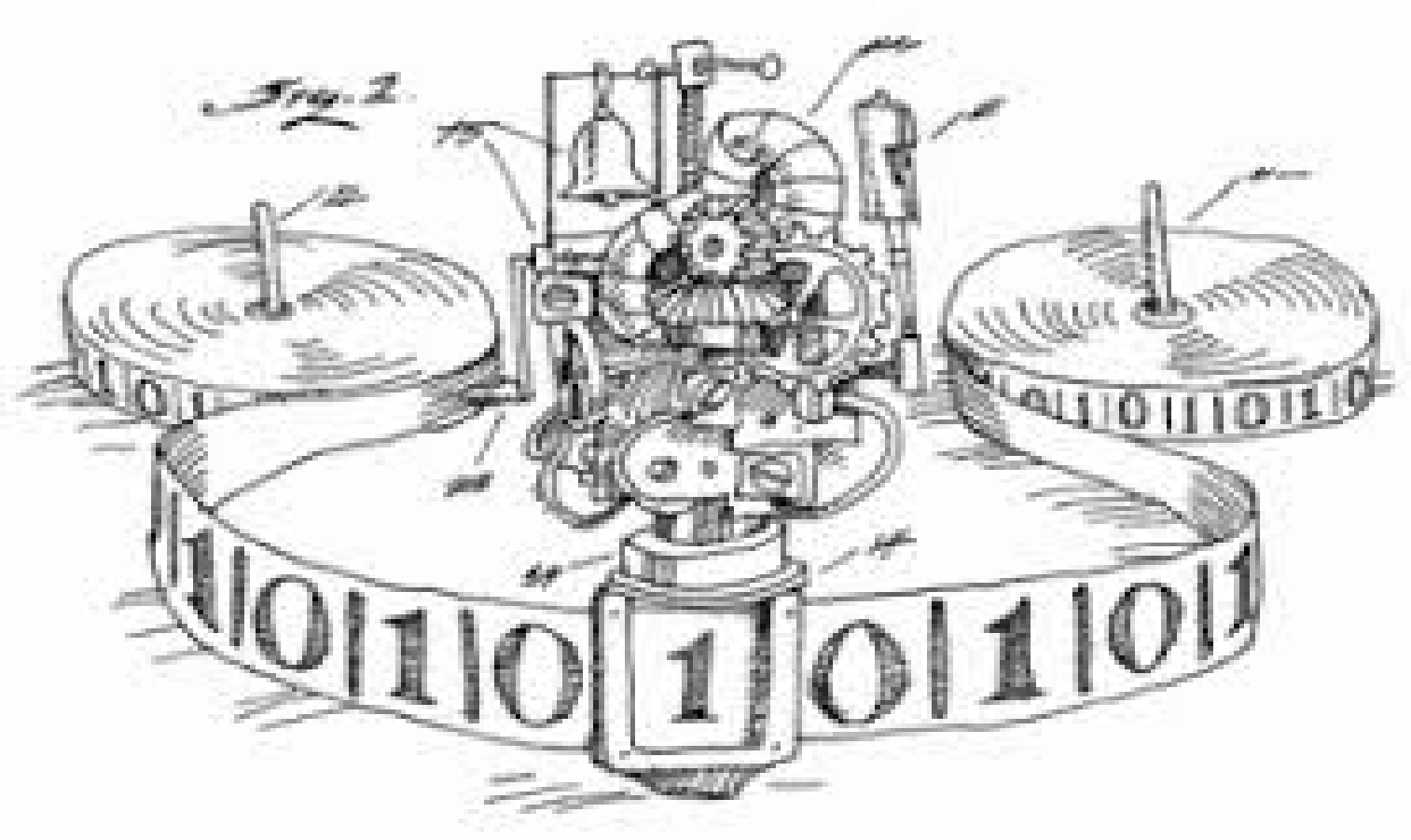


Figura: Modelagem via AFD – determinístico e finito: $\lessdot$ exponencial $\gtrdot$

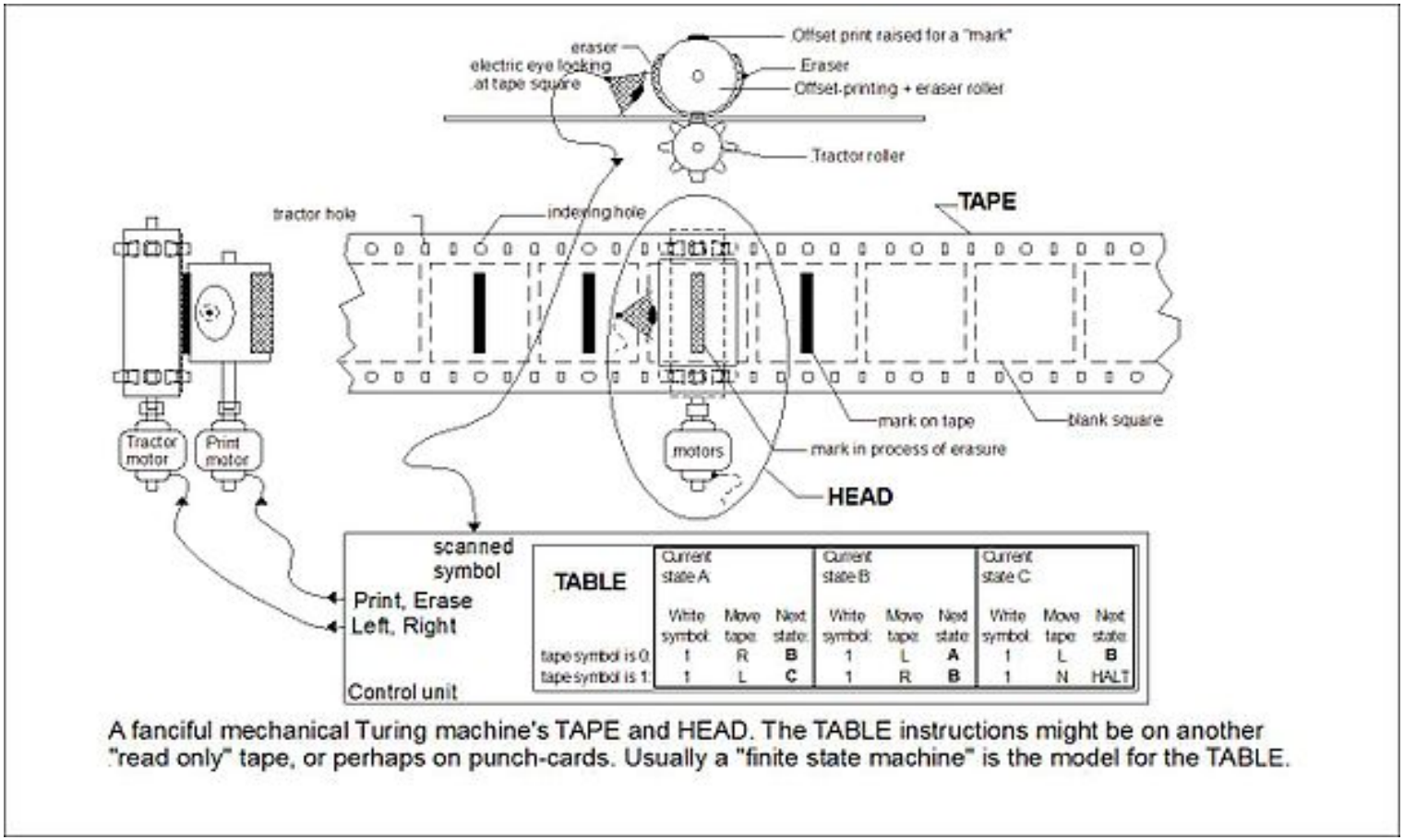
Automatos de Pilha



Uma máquina *forte* e robusta: Máquina de Turing



Máquina de Turing e seus conceitos computacionais



A fanciful mechanical Turing machine's TAPE and HEAD. The TABLE instructions might be on another "read only" tape, or perhaps on punch-cards. Usually a "finite state machine" is the model for the TABLE.

Figura: Memória, entrada, saída, programa armazenado, escrita, etc.

Máquina de Turing Universal : calcula tudo?

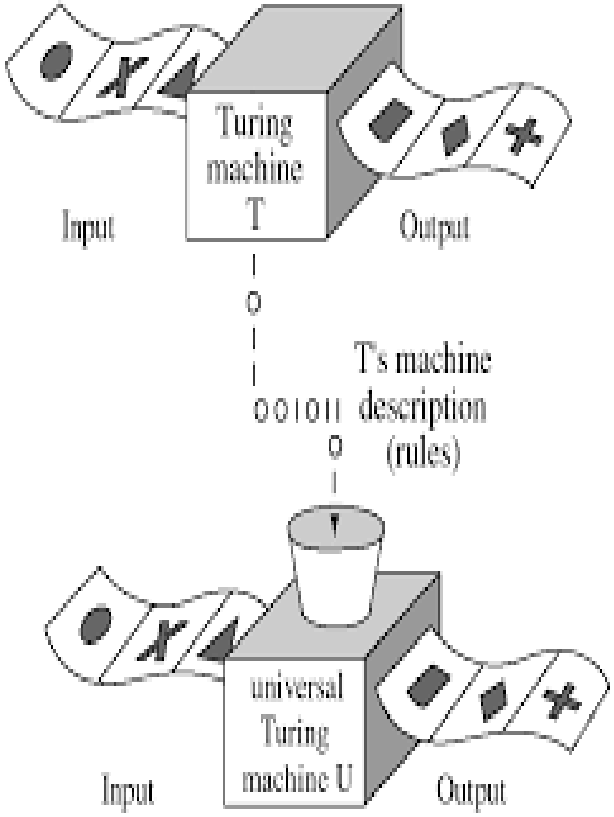
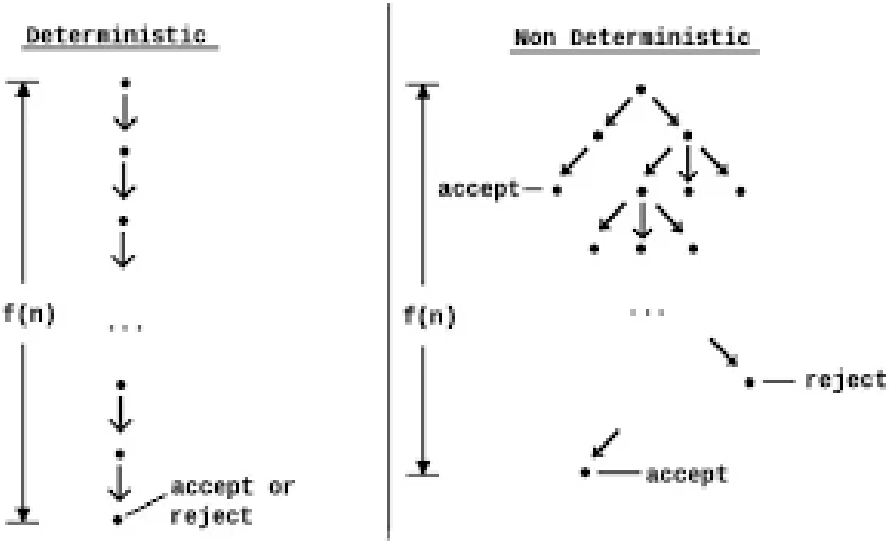
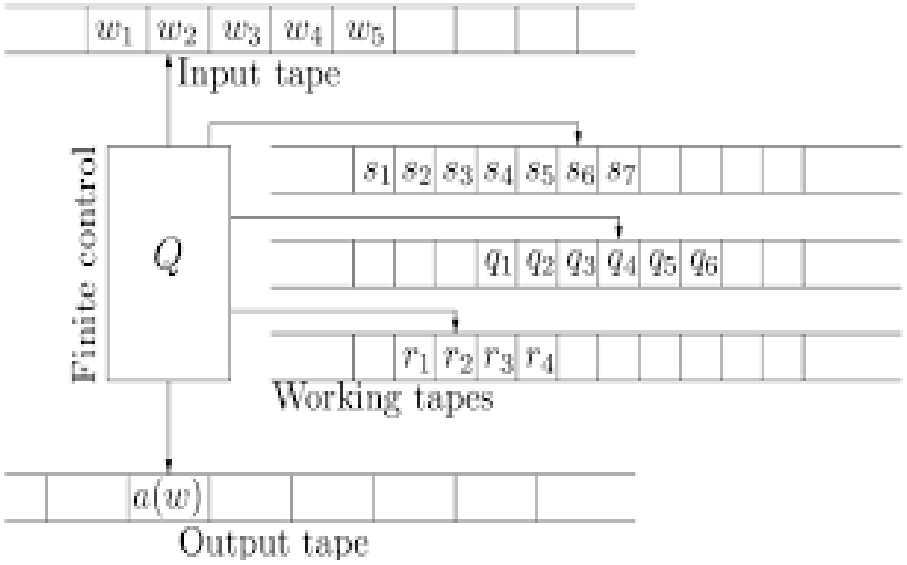


Figura: Se calcula tudo, então vai saber se uma outra máquina vai **parar** ou **não** sob um dada entrada?

- Esta máquina hipotética tem suas limitações
- Por exemplo, dizer se ela própria vai **parar** ou não, quando sua entrada for a sua própria especificação!
- Este é o famoso problema da parada da Máquina de Turing (*the Halting Turing*)
- Assim os problemas são **decidíveis** ou **reconhecíveis** (as vezes não param)
- Contudo, as MTs ajudaram a construir modelos computacionais robustos e **consistentes** para computação!

Máquina de Turing: Deterministica x Não-Determinística





Buscas $\lessdot$ <u>exponenciais</u> $\gtrdot$ na fita da MT, da ordem: $\gtrapprox 2^{f(n)}$

Os algoritmos consomem **tempo** e **memória** sob uma determinada máquina (modelo formal – abstrato ou físico).

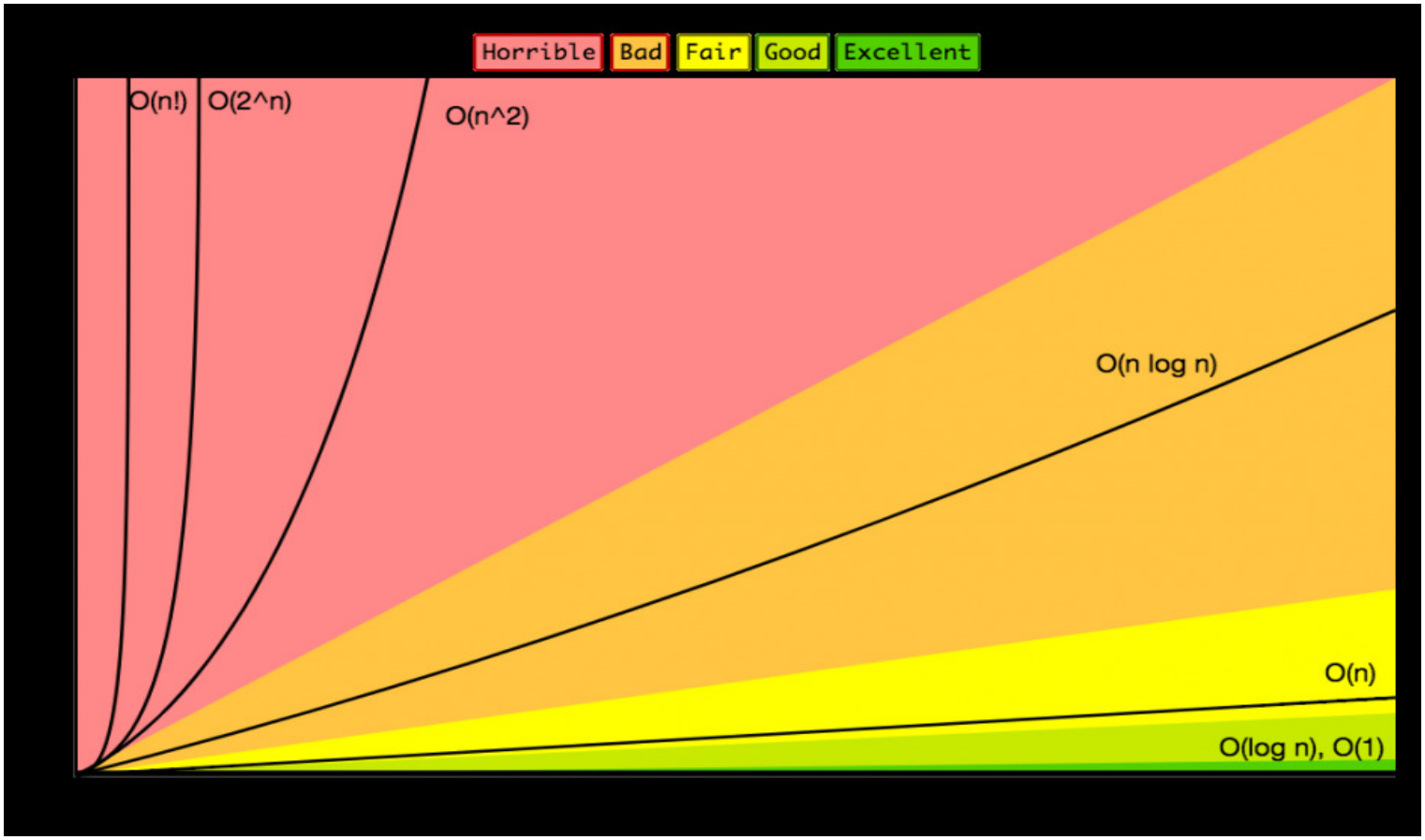


Figura: Complexidade dos algoritmos quanto ao **tempo**.
Tempo **não** pode ser reusado, já memória: **sim**!

Máquinas que calculam sobre linguagens são resumidas em:

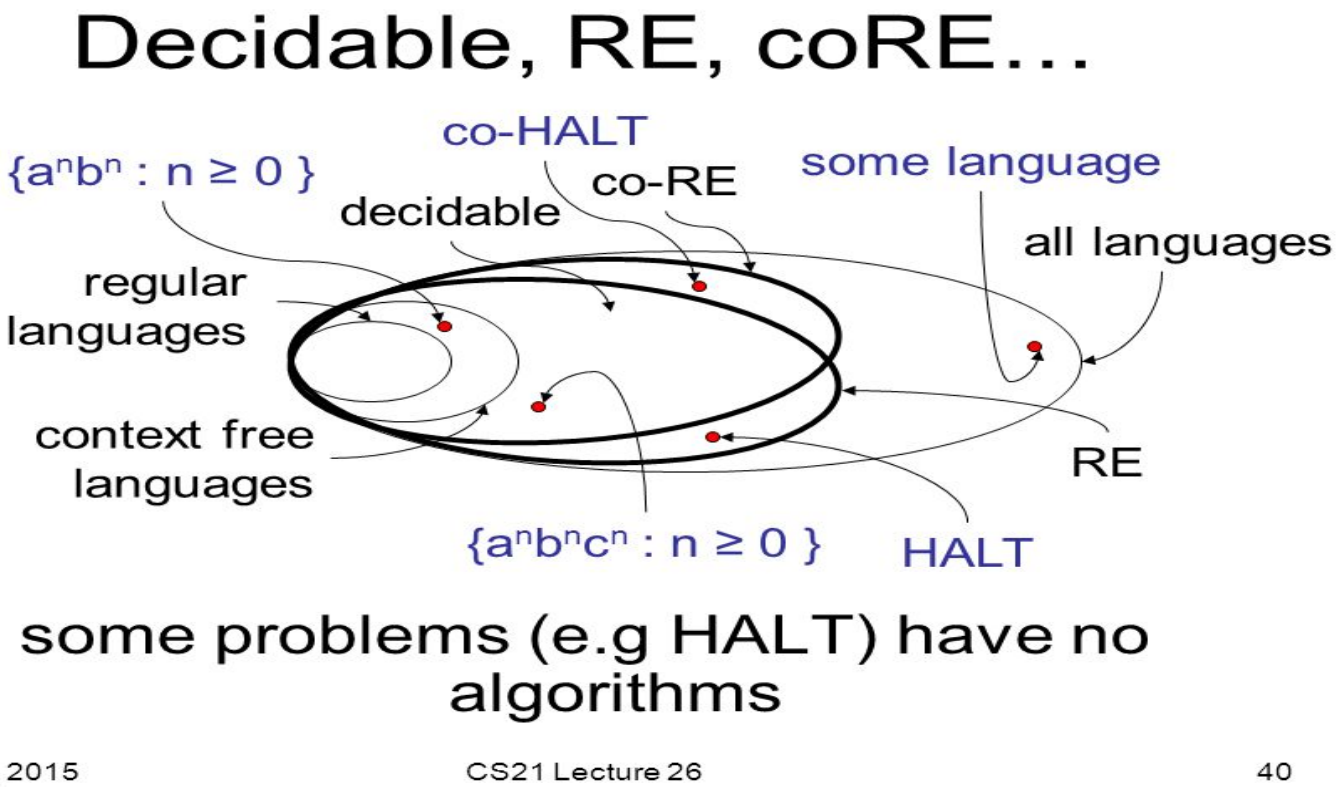


Figura: RE: reconhecíveis $\approx$ linguagens finitas ou não, mas sem garantias da existência de um algortimo que **SEMPRE** pare!

Ampliando a visão anterior tem-se: linguagens, máquinas – modelos abstratas versus problemas resolvidos (total e parcial):

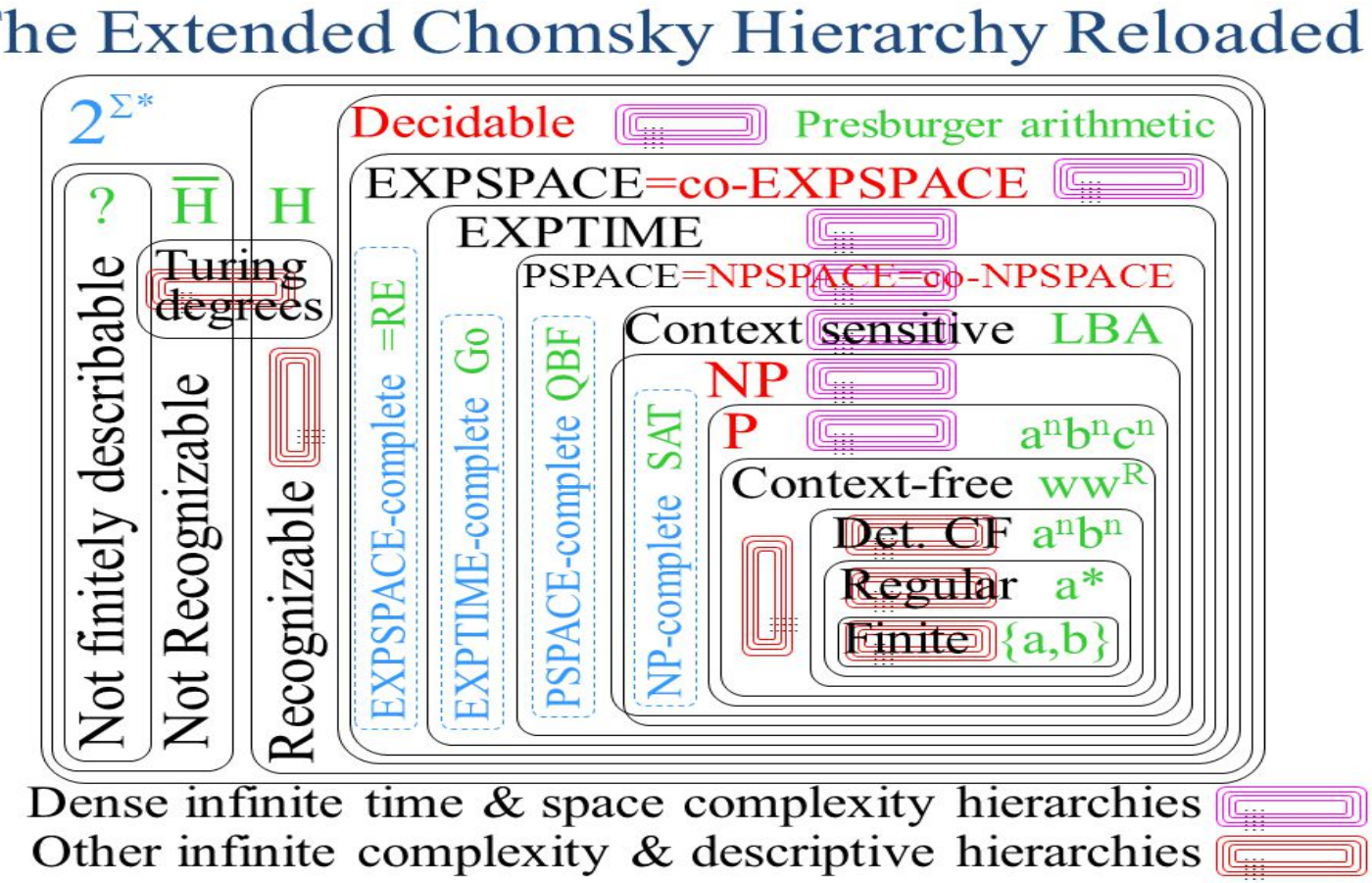


Figura: Modelos, formalismos, máquinas e complexidade!

Como se relacionam estas linguagens e/ou problemas?

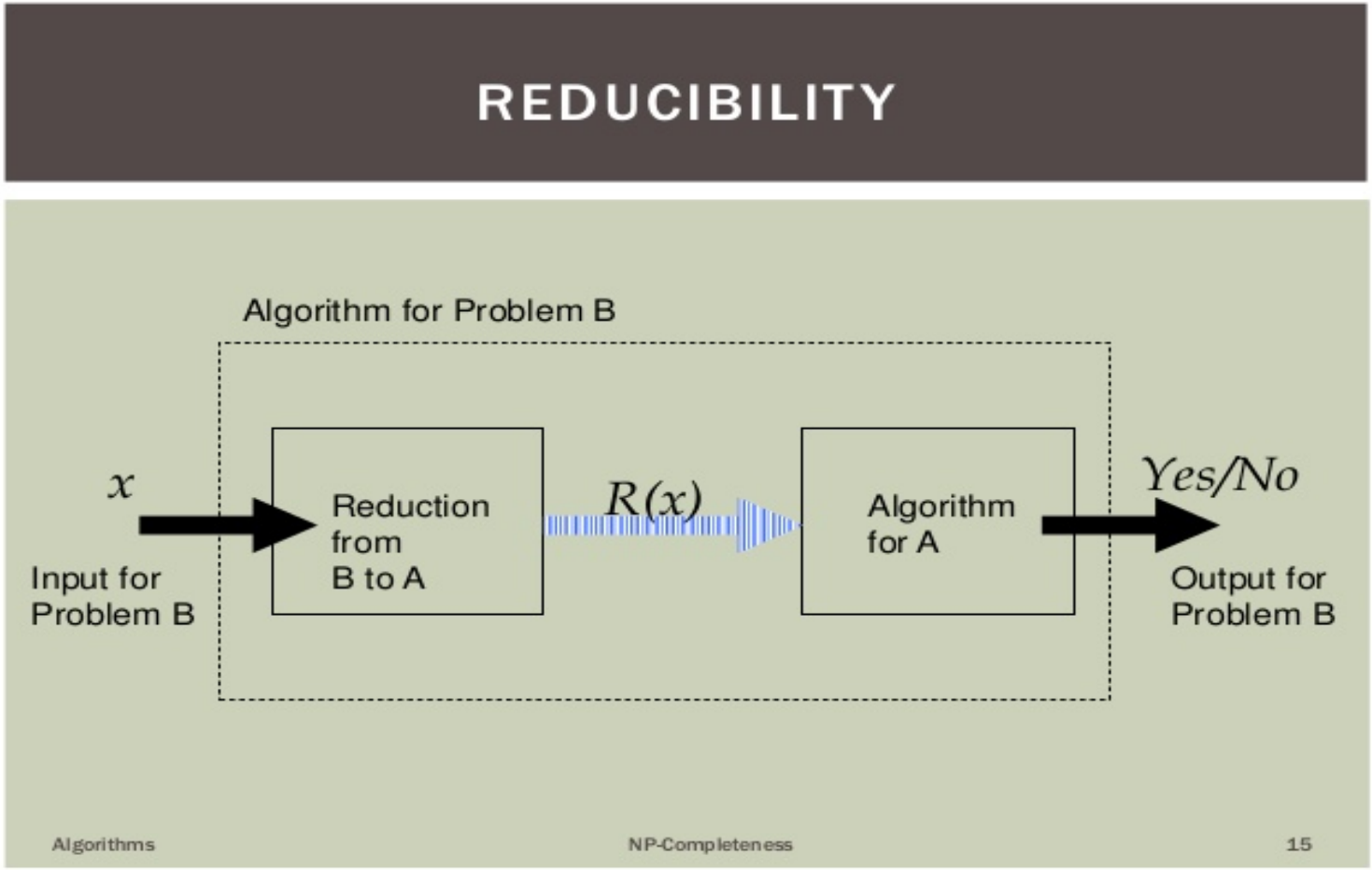


Figura: Problema B se reduz a A ou $B \leq A$ (Reduções de Karp)

A *redução* é a **engrenagem** de se estabelecer propriedades, a existência e dificuldade de soluções entre problemas:

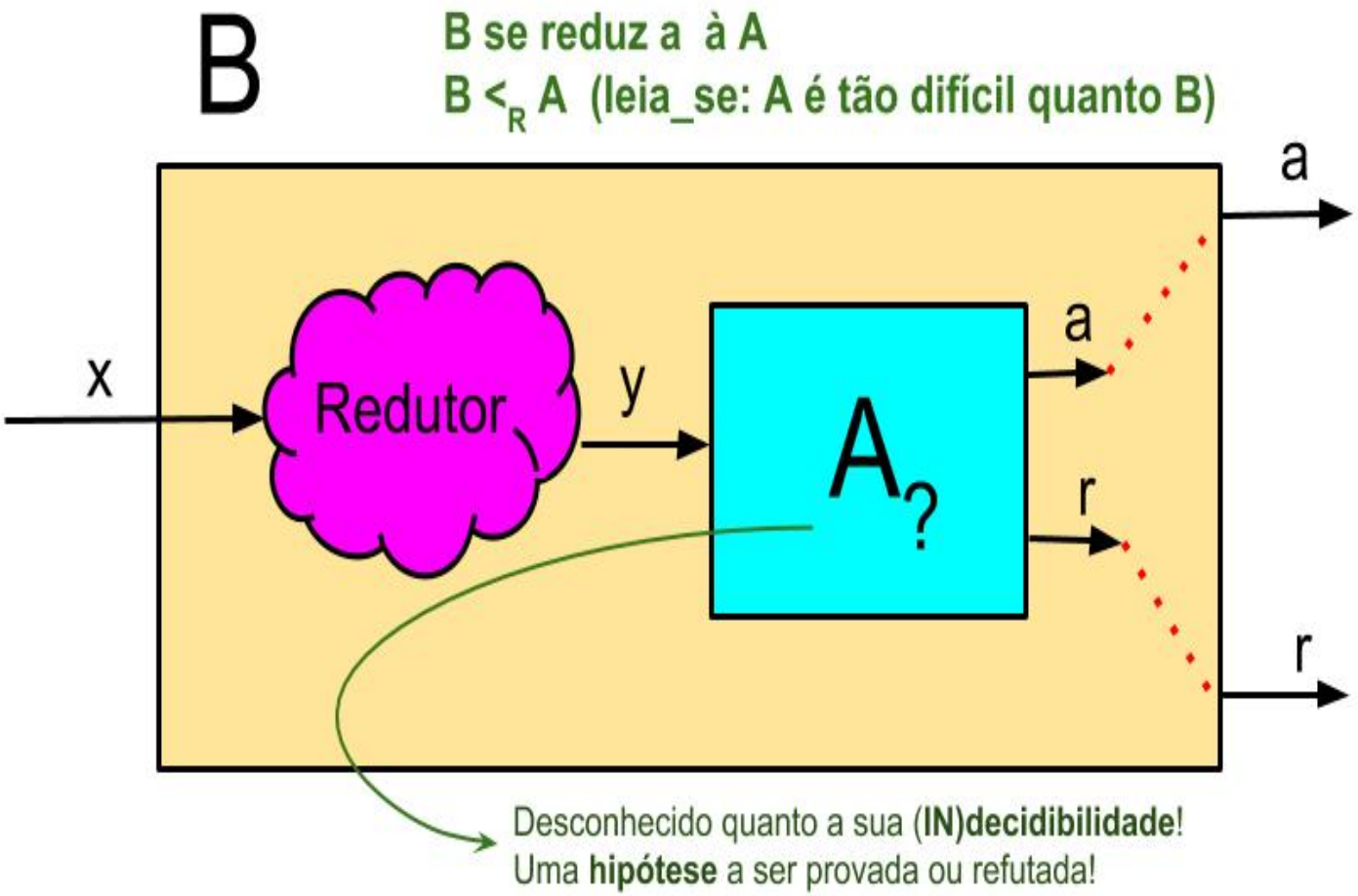


Figura: Problema B se reduz a A ou $B \leq A$

O objetivo é tornar esta *redução* (uma **engrenagem**) entre os problemas sem muita complexidade:

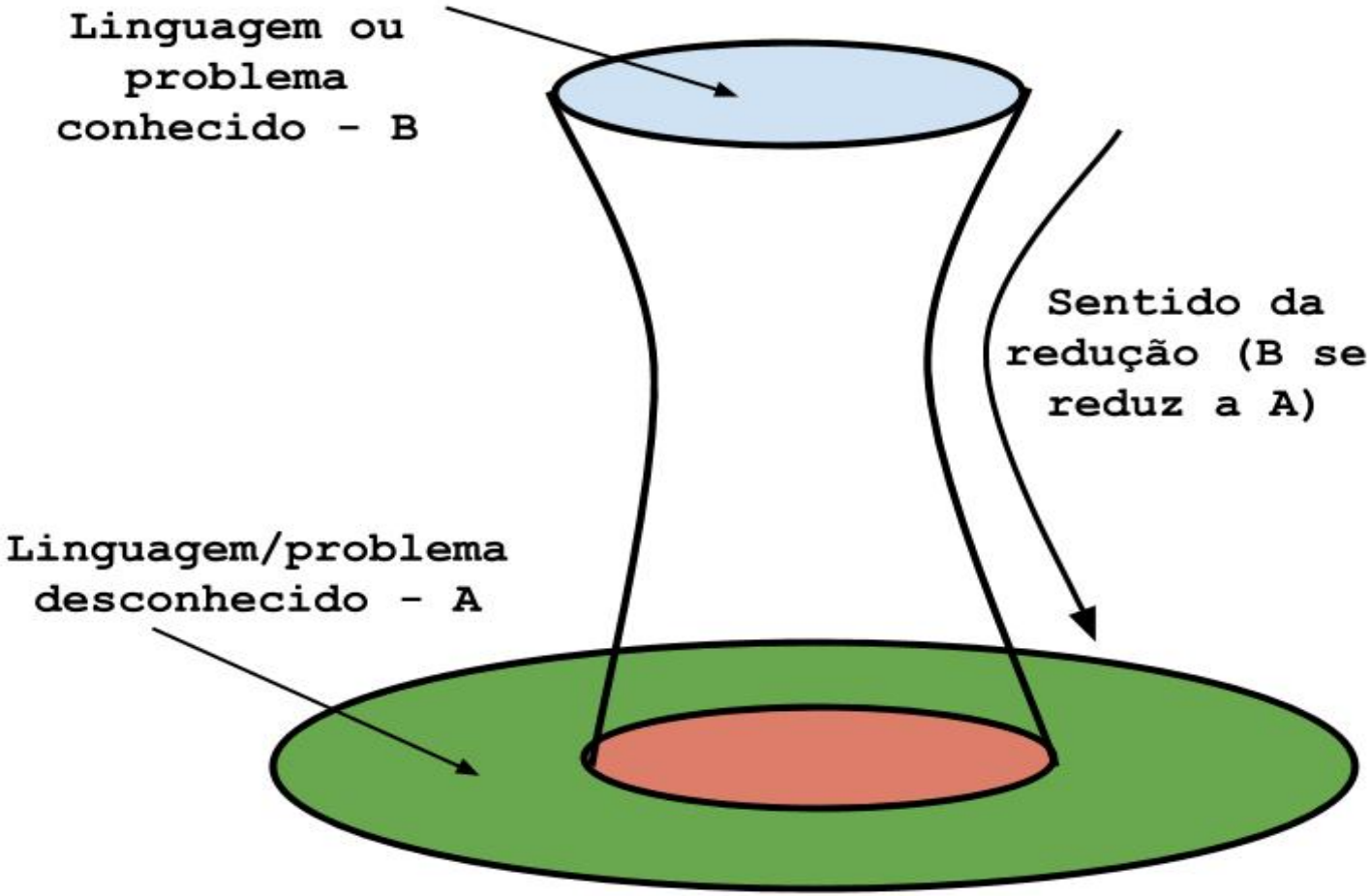


Figura: Problema B se reduz a A ou $B \leq A$

Exemplo de *redução*:

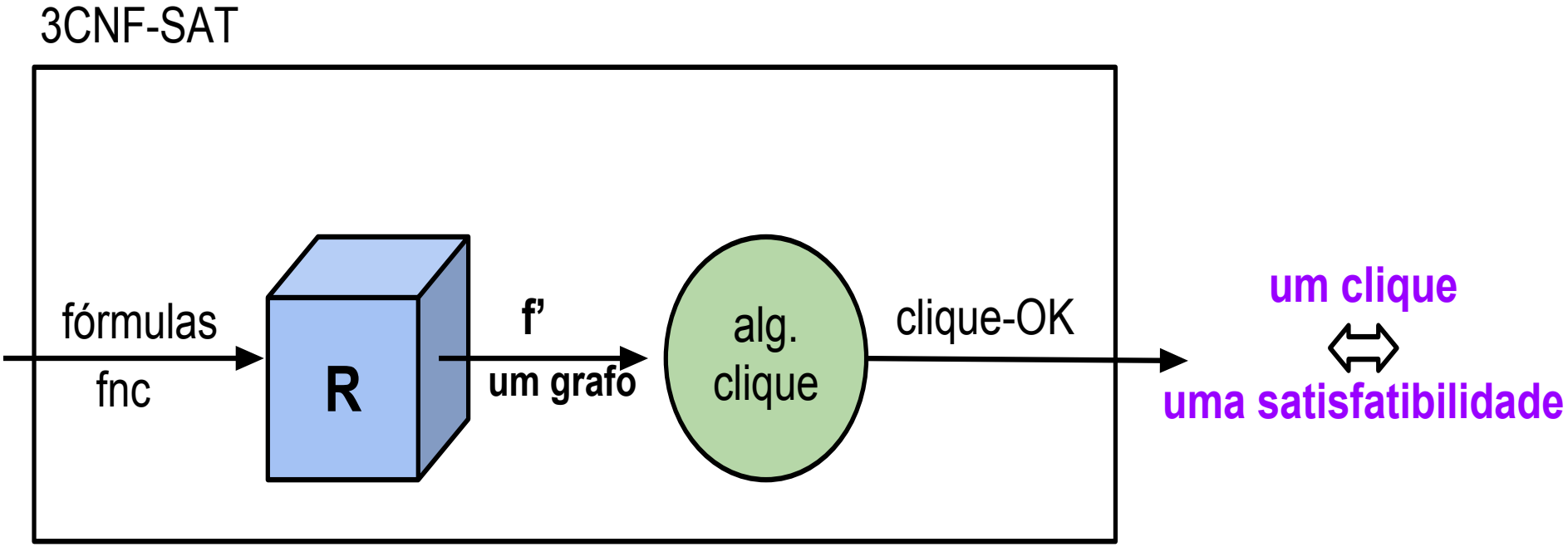


Figura: Há uma *engrenagem* (uma redução específica) nesta reduçao!

Muitos problemas podem ser reduzidos:

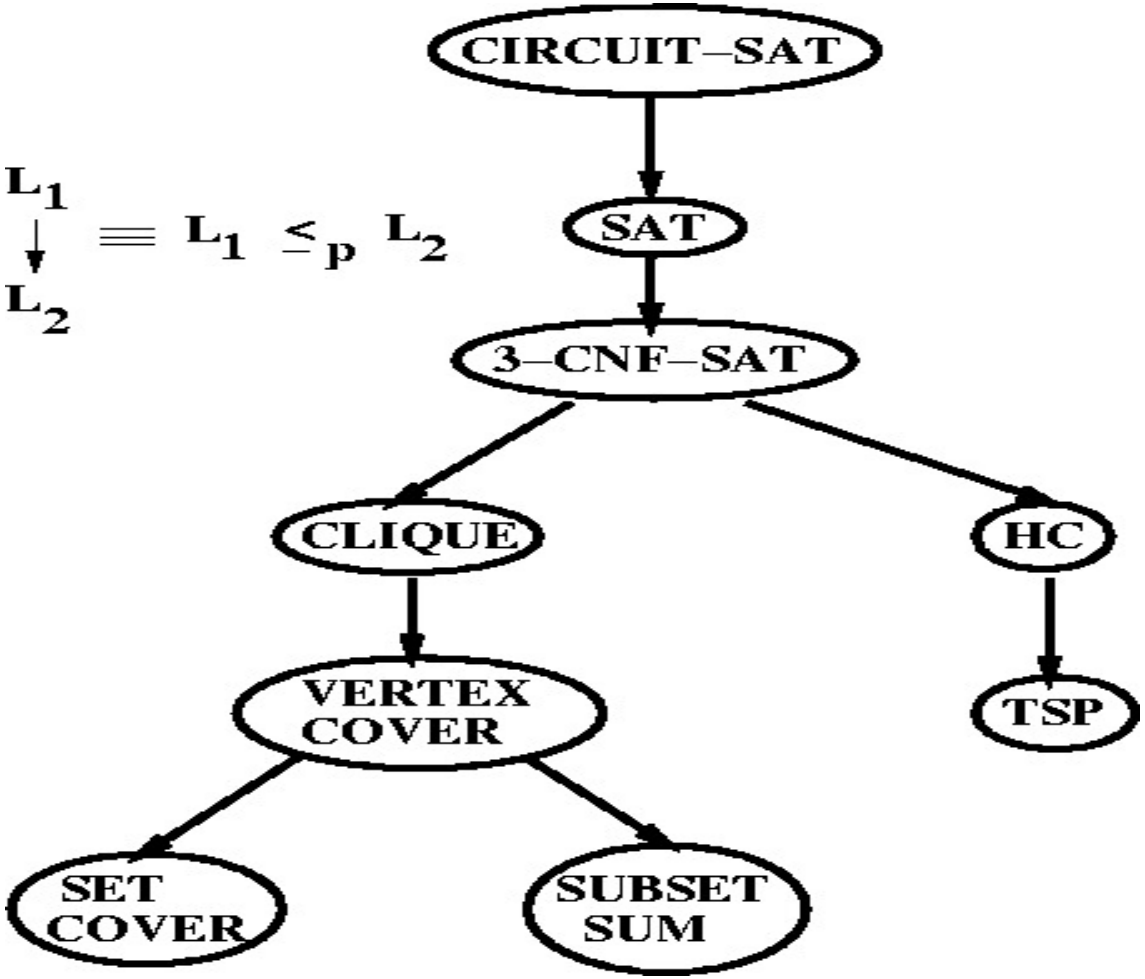
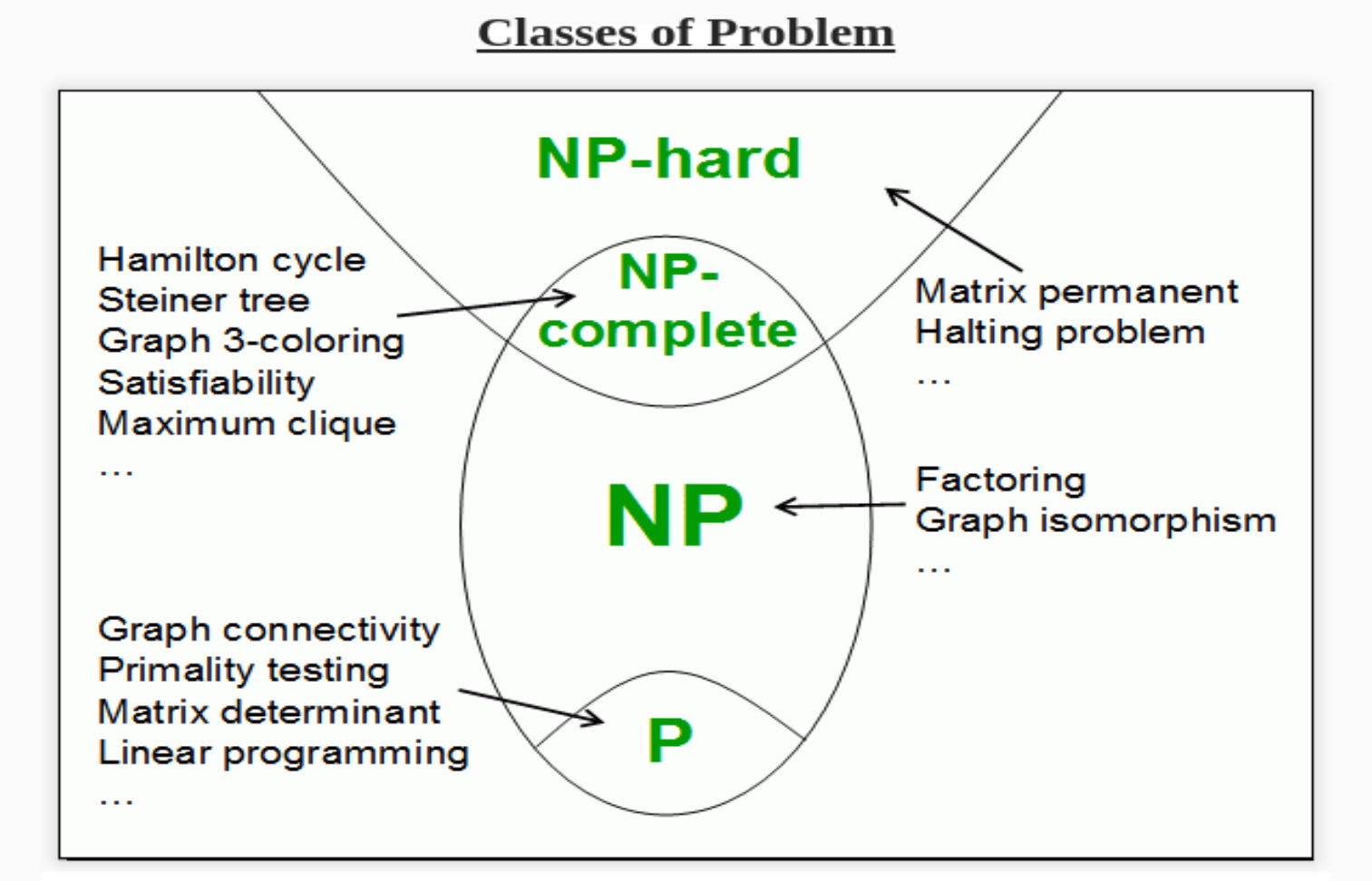


Figura: Há uma *engrenagem* (uma redução específica) nesta reduçao!

Em geral, o cientista da CC está focado em reconhecer e resolver problemas de *classes*:



Referências

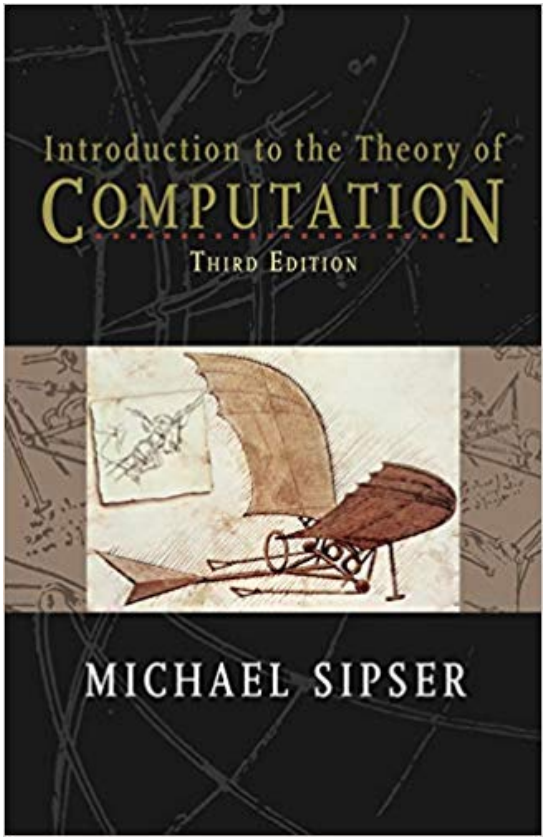


Figura: 3a. Edição: algumas provas claras!

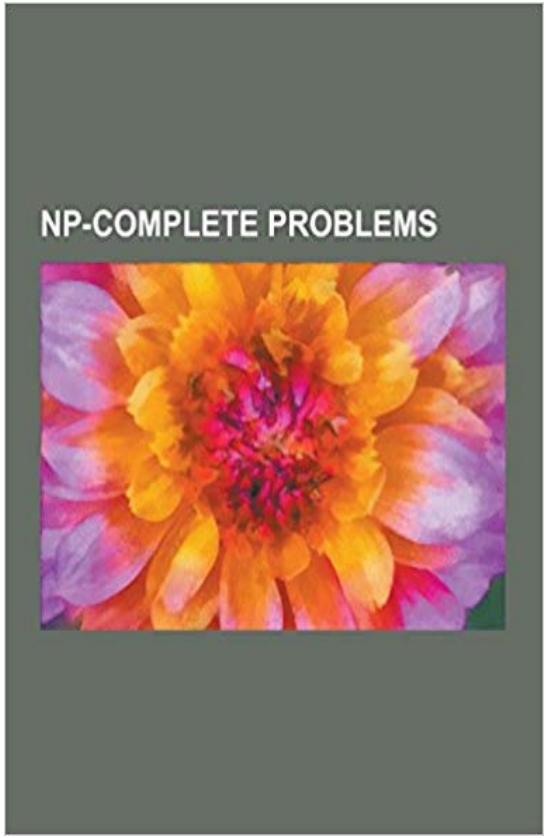


Figura: Lista muitos problemas NP-completos. Conteúdo compilado da Wikipédia.

Conclusões

- Tudo começou com alfabetos de símbolos, usados para montar palavras e estas definem uma linguagem
- Linguagens devem ser computadas sob máquinas abstractas
- Linguagens $\Leftrightarrow$ problemas (se equivalem)
- Estas precisam computadas sob máquinas abstractas, as quais definem a complexidade em se computar um problema
- Há uma *engrenagem* em se descobrir relações entre problemas: a *redução*
- .....

Conclusões

- Tudo começou com alfabetos de símbolos, usados para montar palavras e estas definem uma linguagem
- Linguagens devem ser computadas sob máquinas abstractas
- Linguagens $\Leftrightarrow$ problemas (se equivalem)
- Estas precisam computadas sob máquinas abstractas, as quais definem a complexidade em se computar um problema
- Há uma *engrenagem* em se descobrir relações entre problemas: a *redução*
- .....
- Área teórica da CC é paradoxalmente complicada–difícil, mas, bela e sedutora!

Agradecimentos

- Obrigado
- Comentários e dúvidas são muito bem-vindos
- Email: claudio.sa@udesc.br
- Canal do Youtube: Claudio Cesar de Sá – ccs1664@gmail.com