

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

Claudio Cesar de Sá  
claudio.sa@udesc.br

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

13 de dezembro de 2018

# Contextualizando

- Algumas figuras vieram de vários autores *by Google*
- Propositalmente, as mantive originais!
- Cabeçalho e sequência – do autor
- 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](https://github.com/claudiosa/CCS/tree/master/linguagens_formais_LFA/extras_lfa_CCS)

# Resolvendo Problemas

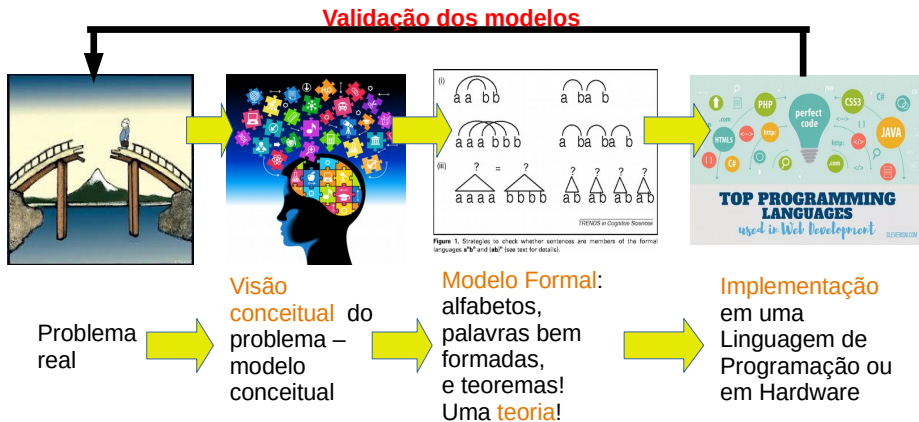
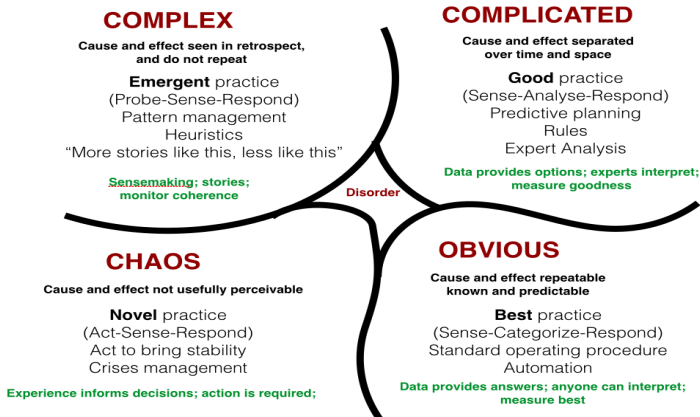


Figura: O quê o cientista busca?

O *Dilema* dos problemas é:



# Problemas × Complexidade



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

A área da CC se preocupa com:

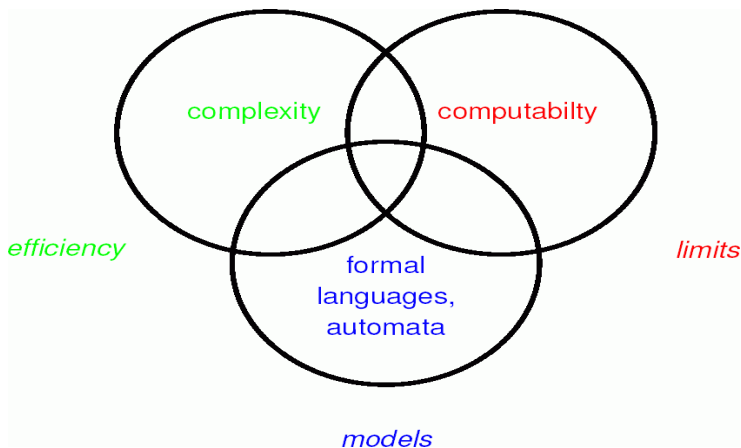


Figura: Precisamos construir modelos e medir seu desempenho!

# Um escopo de formalismos da CC:

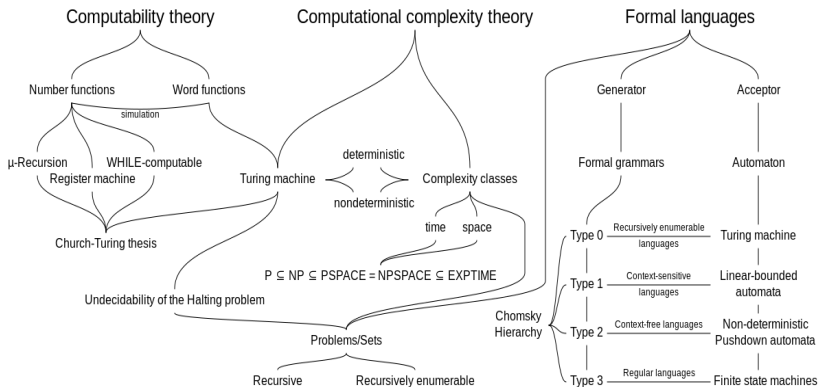
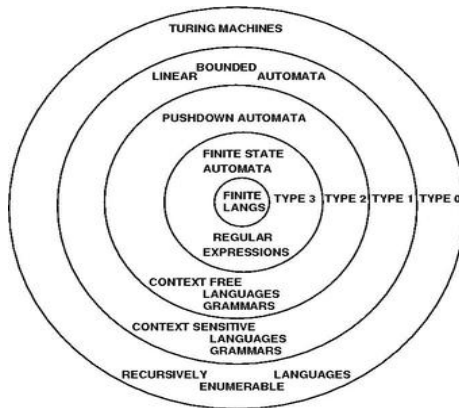


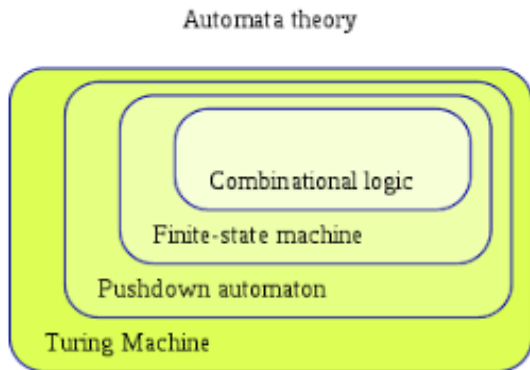
Figura: Basicamente um guia desta apresentação

# LFA – Linguagens Formais





# Máquinas que calculam estas linguagens:



# Automatos de Estados Finitos

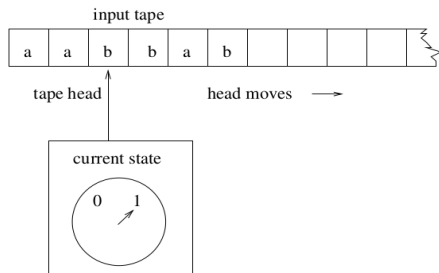


Figura: Uma unidade de controle com estados finitos

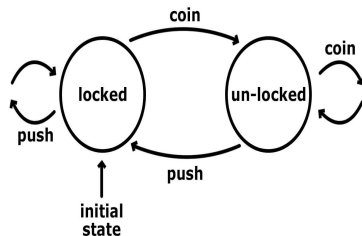


Figura: Exemplo: maleiro e moedas

# Exemplo



Figura: Problema do Pastor

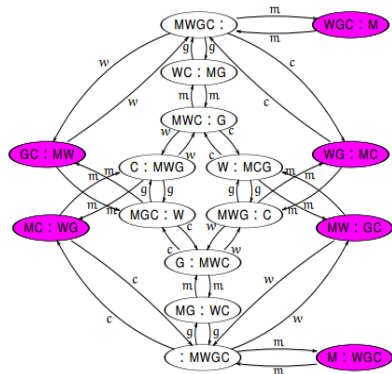
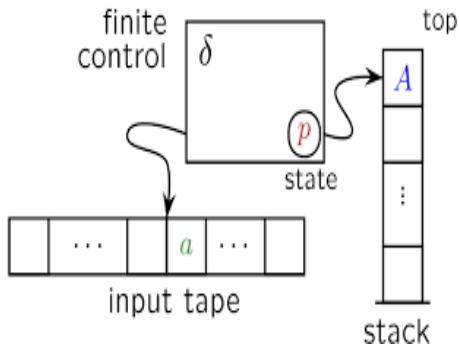
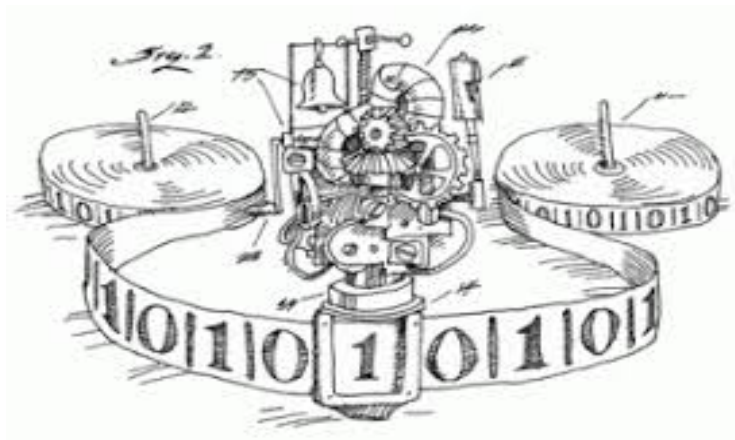


Figura: Modelagem via AFD –  
determinístico e finito: < exponencial >

# Automatos de Pilha



# Uma máquina *forte* e robusta: Máquina de Turing



# Máquina de Turing e seus conceitos computacionais

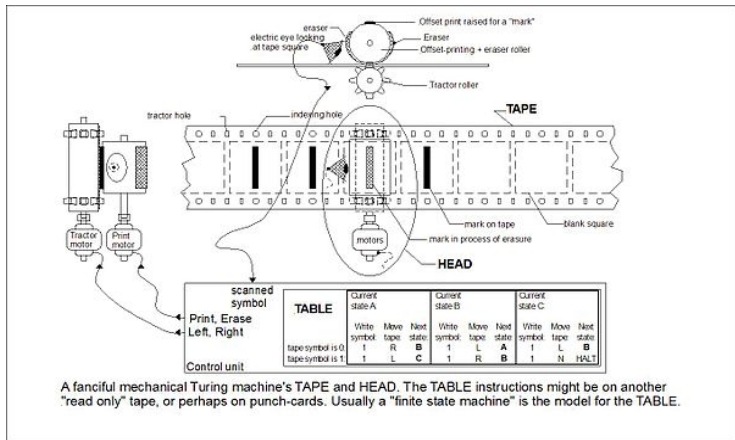
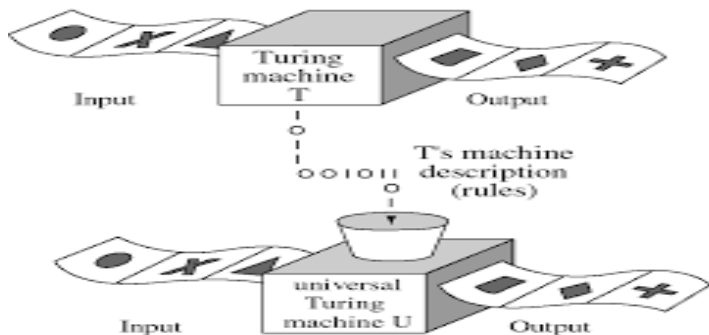


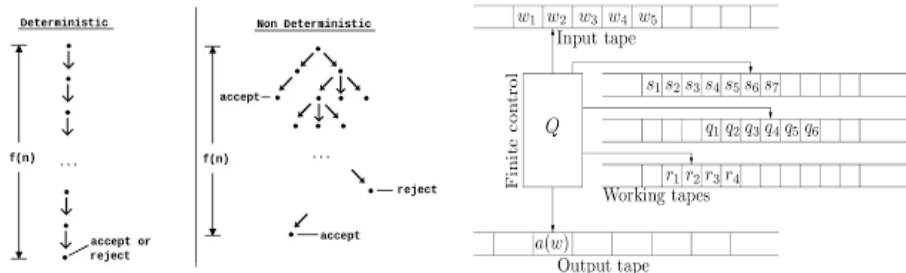
Figura: Memória, entrada, saída, programa armazenado, escrita, etc.

# 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?

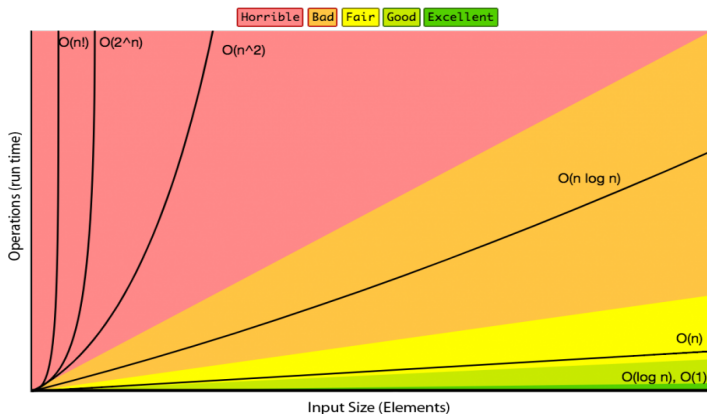
# Máquina de Turing: Determinística x Não-Determinística



Buscas  $\leq$  exponenciais  $\geq$  na fita da MT, da ordem:  $\approx 2^{f(n)}$



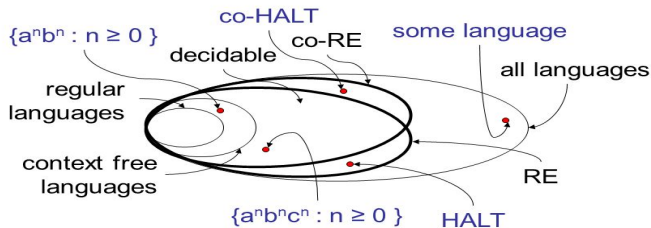
Os algoritmos consomem **tempo** e **memória** sob uma determinada máquina (modelo formal – abstrato ou físico).



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

# Máquinas que calculam sobre linguagens são resumidas em:

## Decidable, RE, coRE...



some problems (e.g HALT) have no algorithms

March 9, 2015

CS21 Lecture 26

40

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

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

## The Extended Chomsky Hierarchy Reloaded

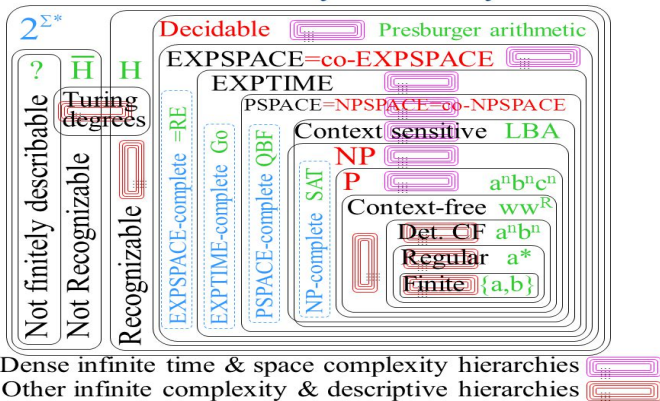


Figura: Modelos, formalismos, máquinas e complexidade!

# Como se relacionam estas linguagens e/ou problemas?

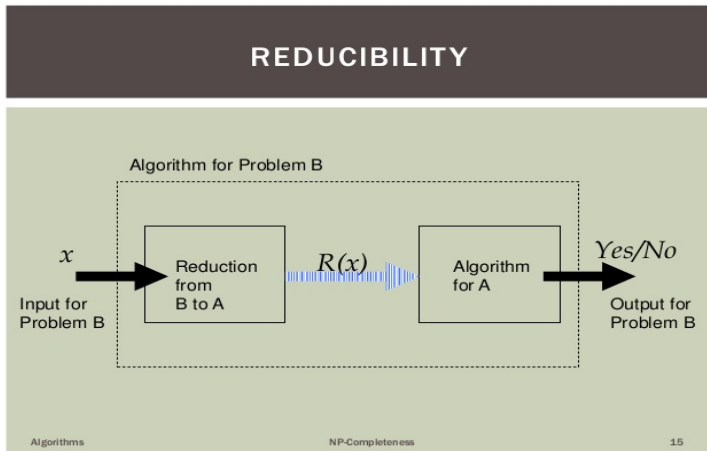


Figura: Problema B se reduz a A ou  $B \leq A$  (Reduções de Karp)

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

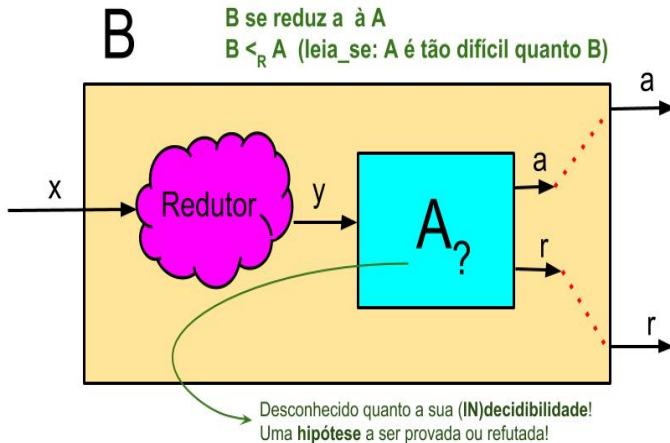


Figura: Problema B se reduz a A ou  $B \leq A$

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

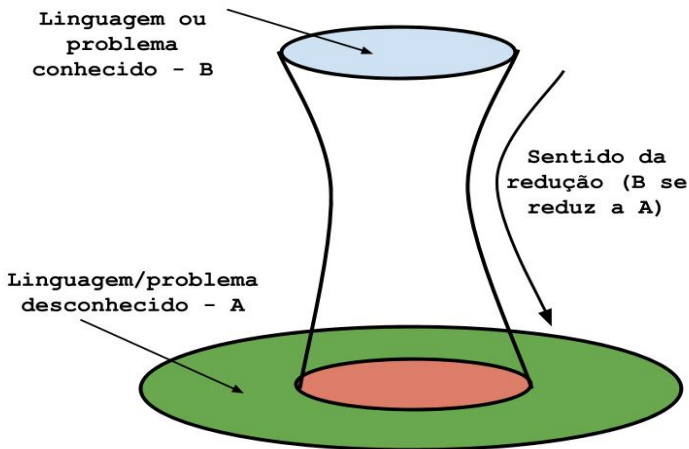


Figura: Problema B se reduz a A ou  $B \leq A$

## Exemplo de *redução*:

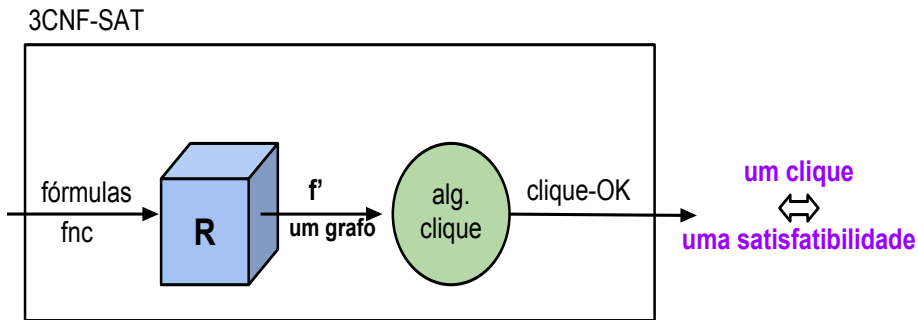


Figura: Há uma *engrenagem* (uma redução específica) nesta redução!

## Muitos problemas podem ser reduzidos:

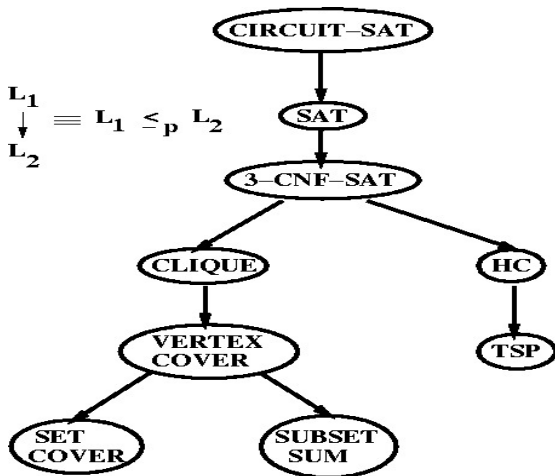
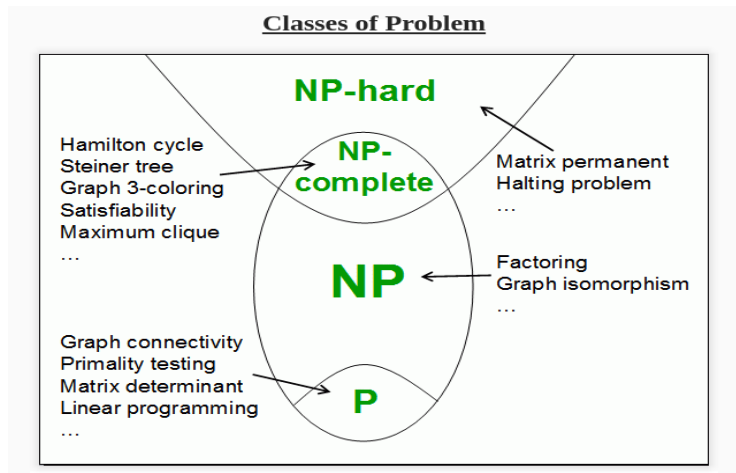


Figura: Há uma *engrenagem* (uma redução específica) nesta redução!



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



# Referências

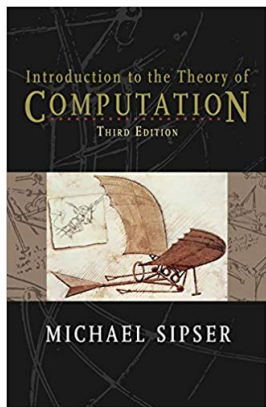


Figura: 3a. Edição: algumas provas claras!

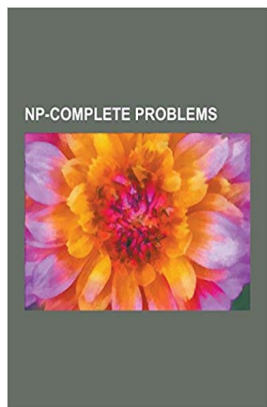


Figura: Lista muitos problemas NP-completos. Conteúdo compilado da Wikipédia.

# 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](mailto:claudio.sa@udesc.br)
- Canal do Youtube: Claudio Cesar de Sá – [ccs1664@gmail.com](mailto:ccs1664@gmail.com)