Como fazer matemática com um computador?

Pablo

Agosto de 2021

Disclaimer

- Baseado em Machine-Assisted Proofs
- Hacker...

```
00110000100011101000011100011010111110101010011000
00010011011001101011110011011101011010000101010110
10111101100000110001110010100000101100111110100000
01001110001000101000110010000001100010001100110110
00100110000010011010010100000110101110110000101101
11011111111010010100110100001110100110001111001110
10101000000101000000000100101000000110001011001010
11001000101110010001010000101101001110110011100101
01010100011110010001010000111010101010000011001111
01100010111011110001111101011011100010000101111100
11011110010011011110001100010101110110100100100000
10110010101101110111011101001011110110011111001111
10100001010100010010111100001010011000010110111110
10011100111000010001011101010110101110001100000010
11011001001001110001000110110000110000111101110111
```

- Me interrompa!

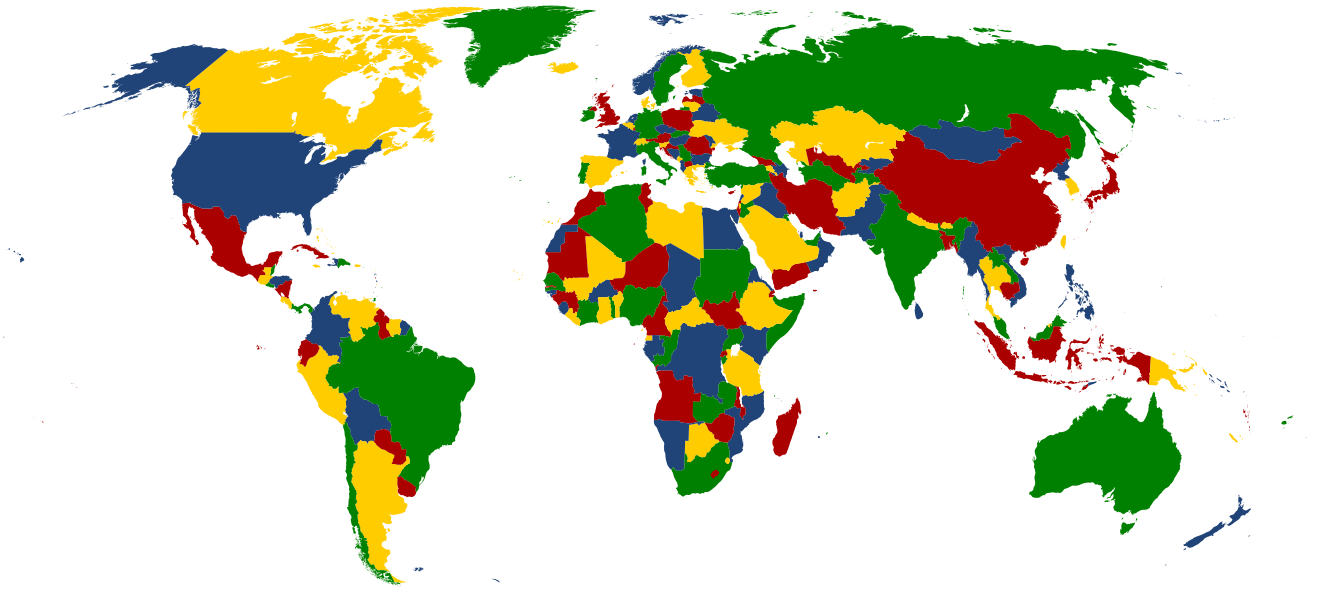
Como fazer matemática com um computador?

- Será que um computador pode nos ajudar a provar teoremas?
- As vezes provar teoremas acaba caindo em continhas
- As vezes são *muitas* continhas

- Será que um computador não consegue fazer as continhas pra mim?

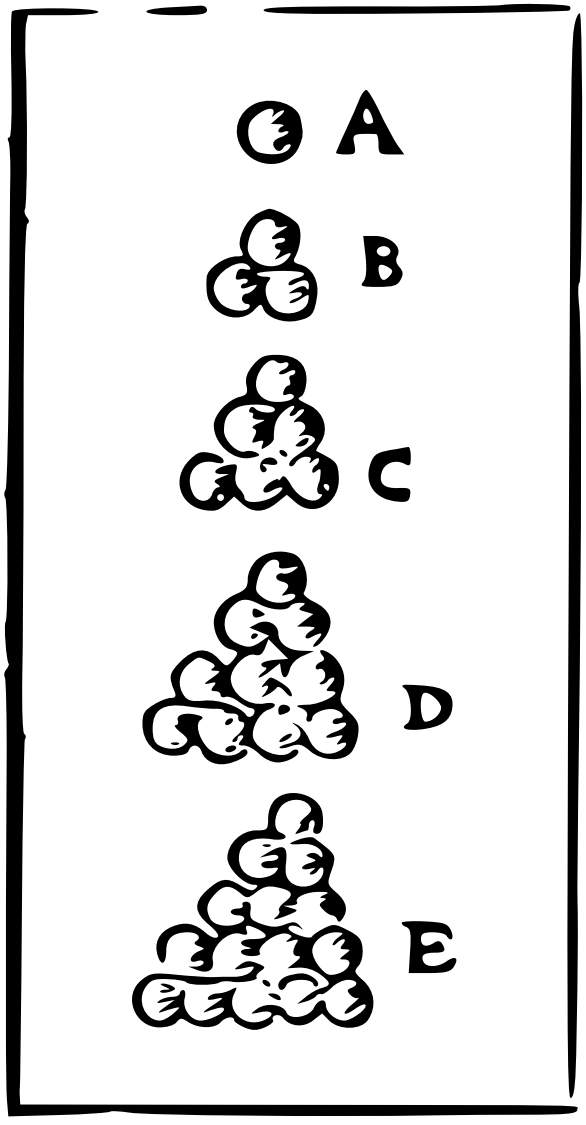
Teorema (Teorema das Quatro Cores)

Todo mapa pode ser pintado de quatro cores distintas de modo dois países adjacentes não compartilham a mesma cor.



- Provado em 1976 por Kenneth Appel e Wolfgang Haken usando o método *discharging*

Provar Por Exaustão



- Se eu tenho um número finito de casos...posso testar todos os casos!

```
for x in X:
    assert p(x)
```

Teorema (Conjectura de Kepler)

Nenhum arranjo de esferas em três dimensões tem uma densidade média maior do que o arranjo cúbico.

- Provado em 1998 por Thomas Hales via exaustão
- Limitações
  - Pouco acessível
  - E se tiver um bug no programa que eu escrevi?
  - E se o número de casos não for finito?

Métodos Não-Numéricos?

- E se as contas que eu preciso fazer não são *numéricas*?
  - Indução é conta
  - Continhas em algebra abstrata

$$n! \cdot m! \mid (n+m)!$$

- Antes de tentar provar meu teorema, eu preciso enunciar ele...

$$n! = \begin{cases} 1 & \text{se } n = 0 \\ n \cdot (n-1)! & \text{se } n > 0 \end{cases}$$

```
def f(n):
    if n == 0:
        return 1
    else:
        return n * f(n - 1)

    # ???
```

$$n! \cdot m! \mid (n+m)!$$

- Temos termos, mas não temos formulas!

Termos	Fórmulas
Um número n	"$n > m$"
Um conjunto X	"$x \in X$"
Pontos, retas, planos, ...	O quinto postulado de Euclides

A Correspondência de Curry-Howard

- Provar que vale A é o mesmo que mostrar que $\{\text{provas de } A\} \neq \emptyset$
- Proposições como tipos

$$\begin{aligned} A &\quad\leftrightsquigarrow\quad \{\text{provas de } A\} \\ A \text{ e } B &\quad\leftrightsquigarrow\quad \{\text{provas de } A\} \times \{\text{provas de } B\} \\ A \text{ ou } B &\quad\leftrightsquigarrow\quad \{\text{provas de } A\} \cup \{\text{provas de } B\} \\ A \Rightarrow B &\quad\leftrightsquigarrow\quad \{\text{provas de } A\} \longrightarrow \{\text{provas de } B\} \end{aligned}$$

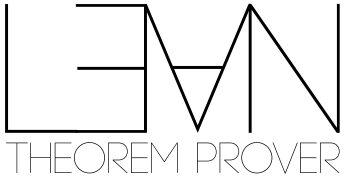
- Mas como eu represento isso num computador?
- Tipos

```
class AndAB:
    """A e B <-> AndAB"""
    def __init__(self, a: A, b: B):
        self.left = a
        self.right = b
```

- Se eu conseguir contruir um valor to tipo **AndAB** eu provei que vale A e B
- Mas o Python não sabe checar se o valor que eu construi é de fato do tipo **AndAB**

Provadores de Teoremas

- Linguagens estaticamente tipadas
- Fixo certas coisas que quero impor que sejam verdade
  - Dados x e y, existe o tipo `x = y` das provas de que $x = y$
  - O tipo `x = x` tem um único elemento `rfl`
- O tipo `x = y` varia com x e y
- Provadores de Teoremas
  - Coq (https://coq.inria.fr/)
  - Idris (https://www.idris-lang.org/)
  - Lean (https://leanprover.github.io/)
  - Isabelle (https://isabelle.in.tum.de/)



Lean

```
theorem ex : Or p q -> Or q p := by
  intro h
  match h with
  | Or.inl _  =>
    apply Or.inr
    assumption
  | Or.inr h2 =>
    apply Or.inl
    exact h2
```

- mathlib
  - Algebra linear
  - Algebra abstrata
  - Topologia
  - Análise
  - Geometria diferencial
  - Teoria dos números
  - Combinatória
  - ...

- A *mathlib* ainda não tem suporte para tudo na matemática...
- A *mathlib* "nunca foi usada para provar algo novo"...
- Pra que tudo isso então?

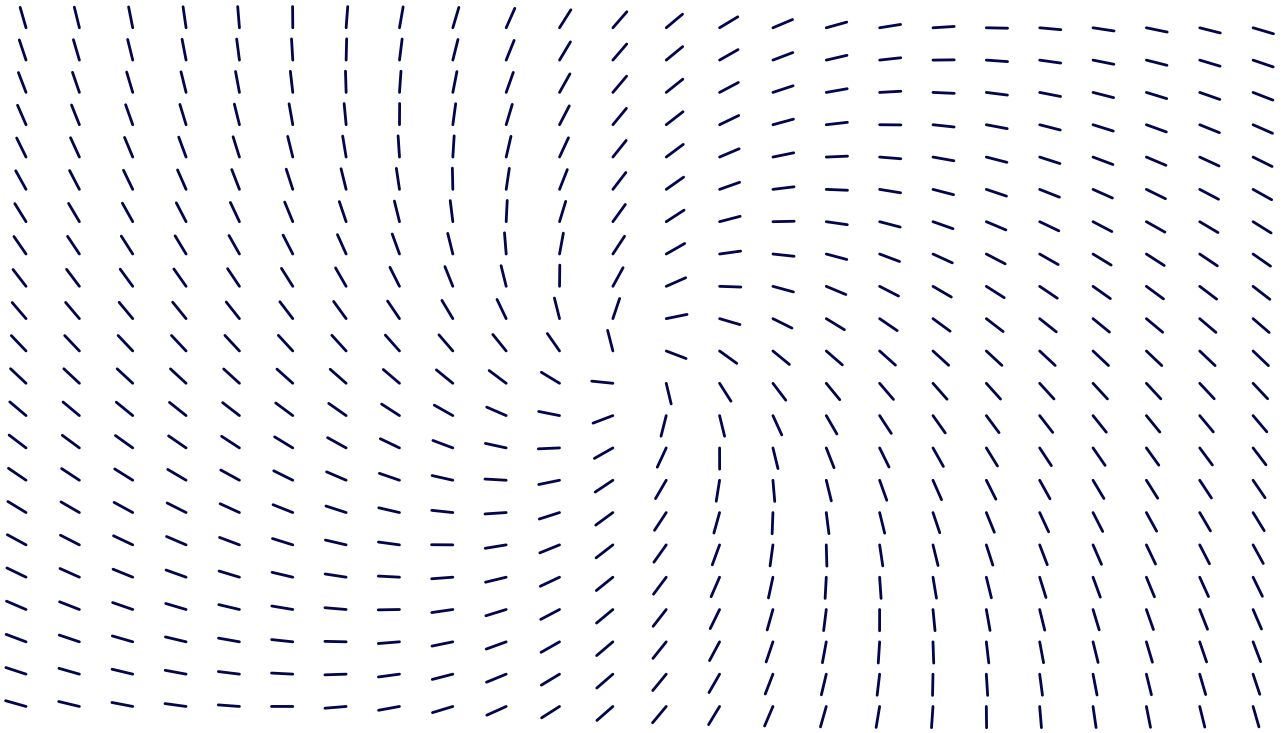
Pra que tudo isso?

- Verificação formal de software e hardware
  - O bug do Pentium de 1994
  - A queda do voo 302 da Ethiopian Airlines em 2019
- Provas de alta complexidade
  - Classificação dos grupos simples finitos
  - A prova de Mochizuki da conjetura abc
  - A prova de Hales da conjectura de Kepler de 2015
  - *Liquid Tensor Experiment*
- Limitações
  - Cadê a intuição?
  - *Who watches the Watchmen?*

Uso Exploratório

- Plotar coisas

$$\frac{\mathrm{d}x}{\mathrm{d}t} = x + y \qquad \frac{\mathrm{d}y}{\mathrm{d}t} = -x + y$$

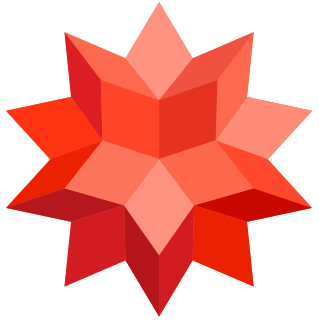


Uso Exploratório

- Continhas bobas

$$\int_0^{\pi} \sin t \cdot \cos^2 t \, \mathrm{d}t = \frac{2}{3}$$

$$\int_0^{\pi} \sin t \cdot \cos^2 t \, \mathrm{d}t \approx 0.66667$$



- Como o Wolfram resolve essa integral?

$$\int_0^{\pi} \sin t \cdot \cos^2 t \, \mathrm{d}t = \left(-\cos^3 t\right)\Big|_{t=0}^{\pi} - 2\int_0^{\pi} \sin t \cdot \cos^2 t \, \mathrm{d}t$$

$$\int_0^{\pi} \sin t \cdot \cos^2 t \, \mathrm{d}t = \frac{\left(-\cos^3 t\right)\Big|_{t=0}^{\pi}}{3}$$

$$\int_0^{\pi} \sin t \cdot \cos^2 t \, \mathrm{d}t = \frac{2}{3}$$

Computação Simbólica

- Calculadora *simbólica*
  - Abre as mesmas contas que abrimos na mão
  - Usa todos os truques do baralho
  - Deixa *coisas indeterminadas* expressas

```
math.sqrt(2) ** 2 # 2.0000000000000004
```

- Computer Algebra System (CAS)
  - Sage (https://www.sagemath.org/)
  - GAP (https://www.gap-system.org/)
  - Mathics (https://mathics.org/)
  - GeoGebra (https://geogebra.org/)