O isomorfismo de Curry-Howard

Ou sobre a similaridade entre provas e programas.

Rodrigo Ribeiro

Logic side: Dedução natural

$$\frac{A \in \Gamma}{\Gamma \vdash A} \qquad \frac{\Gamma \vdash A \to B \quad \Gamma \vdash A}{\Gamma \vdash B}$$

$$\frac{\Gamma \cup \{A\} \vdash B}{\Gamma \vdash A \to B} \qquad \frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}$$

$$\frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A} \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}$$

Type theory side : λ-cálculo tipado simples

$$\frac{x : A \in \Gamma}{\Gamma \vdash x : A}$$

$$\frac{\Gamma \vdash \lambda x.e : A \to B \quad \Gamma \vdash e' : A}{\Gamma \vdash (e \;\; e') : B}$$

$$\frac{\Gamma \cup \{x : A\} \vdash e : B}{\Gamma \vdash \lambda x.e : A \to B}$$

$$\frac{\Gamma \vdash e : A \quad \Gamma \vdash e' : B}{\Gamma \vdash (e, e') : A \times B}$$

$$\frac{\Gamma \vdash e : A \times B}{\Gamma \vdash fst \;\; e : A}$$

$$\frac{\Gamma \vdash e : A \times B}{\Gamma \vdash snd \;\; e : B}$$

Type theory side : λ-cálculo tipado simples

$$\frac{A \in \Gamma}{\Gamma \vdash A}$$

$$\frac{\Gamma \vdash A \to B \quad \Gamma \vdash A}{\Gamma \vdash B}$$

$$\frac{\Gamma \cup \{A\} \vdash B}{\Gamma \vdash A \to B}$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \times B}$$

$$\frac{\Gamma \vdash A \times B}{\Gamma \vdash A}$$

$$\frac{\Gamma \vdash A \times B}{\Gamma \vdash B}$$

Então você percebe...

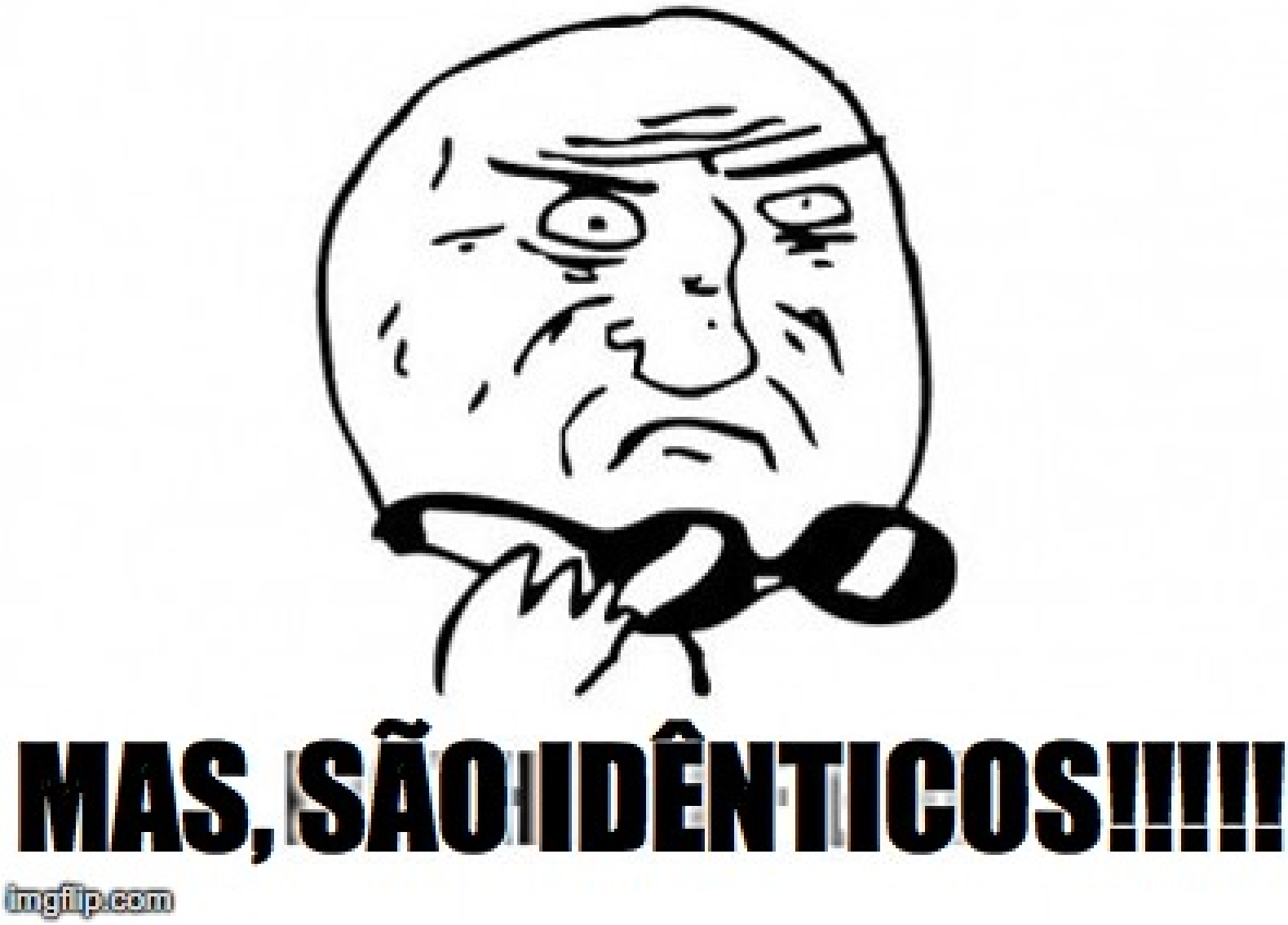

Figure 1: The truth

Uma outra visão da lógica.

- Lógica clássica: toda proposição é verdadeira ou falsa.
- Lógica intuicionista: Uma proposição é verdadeira somente se esta pode ser provada.
  - Mudança de paradigma: verdade sujeita a existência de evidência.
  - Lógica intuicionista é exatamente a lógica clássica sem o axioma do terceiro excluído e propriedades derivadas deste.
  - Ao contrário da lógica clássica, a semântica da lógica intuicionista é baseada na construção de provas, isto é, na dedução natural.

O isomorfismo de Curry-Howard

- Provas em um dado subconjunto da matemática **correspondem** a programas em uma dada linguagem de programação
  - Descoberto por Curry em '58 e por Howard em '69.
  - Esse "isomorfismo" é também conhecido como "proof-as-programs" correspondence.
- Teoremas nada mais são que tipos e o programa correspondente a prova.
  - Para isso, sua linguagem de programação deve ser expressiva.
  - Não tente provar teoremas usando Java, C/C++, Python... :)

The truth is out there...

Lógica	Computação
Provas	Programas
Fórmulas	Tipos
Axiomas	Primitivas de uma linguagem
A implica B	função de A em B
A e B	par formado por A e B
A ou B	tagged union de A e B
falso	tipo vazio
verdadeiro	tipo unit
$\exists x.P(x)$	um par formado por x e um valor de tipo $P(x)$
$\forall x \in A.P(x)$	uma função de $x : A$ em $P(x)$.

The truth is out there...

- Composição de funções

```
comp :: (B -> C) -> (A -> B) -> A -> C
comp = \ f -> \ g -> \ x -> f (g x)
```

The truth is out there...

$$\Gamma = \{f : B \to C, g : A \to B, x : A\}$$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{f : B \to C \in \Gamma}{\Gamma \vdash f : B \to C} \quad \dfrac{\dfrac{g : A \to B \in \Gamma}{\Gamma \vdash g : A \to B} \quad \dfrac{x : A \in \Gamma}{\Gamma \vdash x : A}}{\Gamma \vdash (g\ x) : B}}{\{f : B \to C, g : A \to B, x : A\} \vdash f\ (g\ x) : C}}{\{f : B \to C, g : A \to B\} \vdash \lambda x : A.f\ (g\ x) : A \to C}}{\{f : B \to C\} \vdash \lambda g : A \to B.\lambda x : A.f\ (g\ x) : (A \to B) \to A \to C}}{\emptyset \vdash \lambda f : B \to C.\lambda g : A \to B.\lambda x : A.f\ (g\ x) : (B \to C) \to (A \to B) \to A \to C}$$

The truth is out there...

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{B \to C \in \Gamma}{\Gamma \vdash B \to C} \quad \dfrac{\dfrac{A \to B \in \Gamma}{\Gamma \vdash A \to B} \quad \dfrac{A \in \Gamma}{\Gamma \vdash A}}{\Gamma \vdash B}}{\{B \to C, A \to B, A\} \vdash C}}{\{B \to C, A \to B\} \vdash A \to C}}{\{B \to C\} \vdash (A \to B) \to A \to C}}{\emptyset \vdash (B \to C) \to (A \to B) \to A \to C}$$

The truth is out there...

O termo

$$\lambda f.\lambda g.\lambda x.f\ (g\ \ x)$$

pode ser considerado uma representação da derivação, em dedução natural, da fórmula

$$(B \to C) \to (A \to B) \to A \to C$$

, que é o tipo da função anterior.

The truth is out there...

- Não é só isso: provas por indução são funções recursivas!
- Teorema: Para todo $n \in \mathbb{N}$, existe $p \in \mathbb{N}$ tal que $n = 2p$ ou $n = 2p + 1$.
  - Caso $n = 0$. Imediato.
  - Caso $n = m + 1$. Pela I.H. temos que existe p tal que $m = 2p$ ou $m = 2p + 1$.
    - Caso $m = 2p$: temos que $n = 2p + 1$.
    - Caso $m = 2p + 1$: temos que $n = 2(p + 1)$

The truth is out there...

- Não é só isso: provas por indução são funções recursivas!

```
div2 :: Int -> (Int, Bool)
div2 n
     | n == 0      = (0,True)
     | otherwise = if even then (p ,false)
                           else (p + 1, true)
        where
             (p,even) = div2 (n - 1)
```

A new hope...

$$
\begin{array}{ccc}
\text{Novos axiomas} & \sim & \text{Novas primitivas} \\
\Updownarrow & & \Updownarrow \\
\text{Representação lógica} & \sim & \text{Transformação de programas}
\end{array}
$$

A new hope...

- Princípios da lógica clássica, usados cotidianamente por programadores JS

Novos axiomas		Novas primitivas
$A \vee \neg A$	$\sim$	`callcc`
$\Updownarrow$		$\Updownarrow$
Representação lógica		Transformação de programas
double negation	$\sim$	callback-passing style

Novas perspectivas de pesquisa

- A relação entre lógica e computação é um campo frutífero de pesquisa!
- Homotopy type theory
- Provas como paths, tipos são homotopy spaces
- Novas perspectivas sobre a definição de igualdade em assistentes de provas.

O assistente de provas Coq

- Ferramenta que explora o isomorfismo de Curry-Howard.
- Desenvolvido pelo Inria, França desde 1984.
- Vencedor do ACM Software System Award, 2013

Afinal, onde isso é utilizado?

- Matemática
  - Teorema das 4 cores.
  - Teorema de Feit-Thomson.
  - Formalizações da homotopy type theory.
- Computação
  - Compilador de C/C++ (Compcert)
  - Ferramenta para criação de blogs
  - Bibliotecas para verificação de programas C usando separation logic.

Moral da história

- Tipos são fórmulas da lógica, programas são provas!
- Verificar uma prova nada mais é que o processo de verificação de tipos realizado por um compilador.
- Em essência, lógica e computação são visões diferentes de um mesmo fenômeno.

Slides e materiais para o curso

Todo o material deste curso está disponível no site

http://rodrigogribeiro.github.io/coqcourse