The Curry-Howard isomorphism

for dummies

Pierre-Marie Pédrot

PPS/πr^2

17th February 2015

All you ever wanted to know and were afraid to ask!

(or did not care about)

Overview

In this talk

- A gentle introduction to the Curry-Howard isomorphism
- Bird's eye view, no technical stuff
- A bit of advertising

A brief, biased history of Logic: The Dark Ages

- **4$^{\text{th}}$ c. B.C.** The greek philosopher Aristotle writes the *Organon*.

  "Socrates is a man, all men are mortal, etc."

A brief, biased history of Logic: The Dark Ages

- **4$^{\text{th}}$ c. B.C.** The greek philosopher Aristotle writes the *Organon*.

  "Socrates is a man, all men are mortal, etc."

- **1453.** Fall of Constantinople.

  Meanwhile, heated discussion between byzantine scholars about the sex of angels.

A brief, biased history of Logic: The Dark Ages

- **4$^{\text{th}}$ c. B.C.** The greek philosopher Aristotle writes the *Organon*.

  "Socrates is a man, all men are mortal, etc."

- **1453.** Fall of Constantinople.

  Meanwhile, heated discussion between byzantine scholars about the sex of angels.

- **1901.** Russel shows that there is no set of all sets.

  "The 20th century is the century of mathematics."

A brief, biased history of Logic: The Age of Enlightenment

From the 30's onwards, foundations of mathematics and computer science become intertwingled.

Formal logic	**Computer science**
1929. Completeness theorem	**193X.** λ-calculus
1931. Incompleteness theorem	**1936.** Turing Machines
1936. Undefinability theorem	**1936.** Halting problem

A brief, biased history of Logic: The Age of Enlightenment

From the 30's onwards, foundations of mathematics and computer science become intertwingled.

Formal logic	**Computer science**
1929. Completeness theorem	**193X.** λ-calculus
1931. Incompleteness theorem	**1936.** Turing Machines
1936. Undefinability theorem	**1936.** Halting problem

... and there is a good reason for that.

Through the looking glass

The mathematician

Theorem. For all $n \in \mathbb{N}$, there exists $p \in \mathbb{N}$ such that $n = 2p$ or $n = 2p + 1$.

Proof. By induction on n.

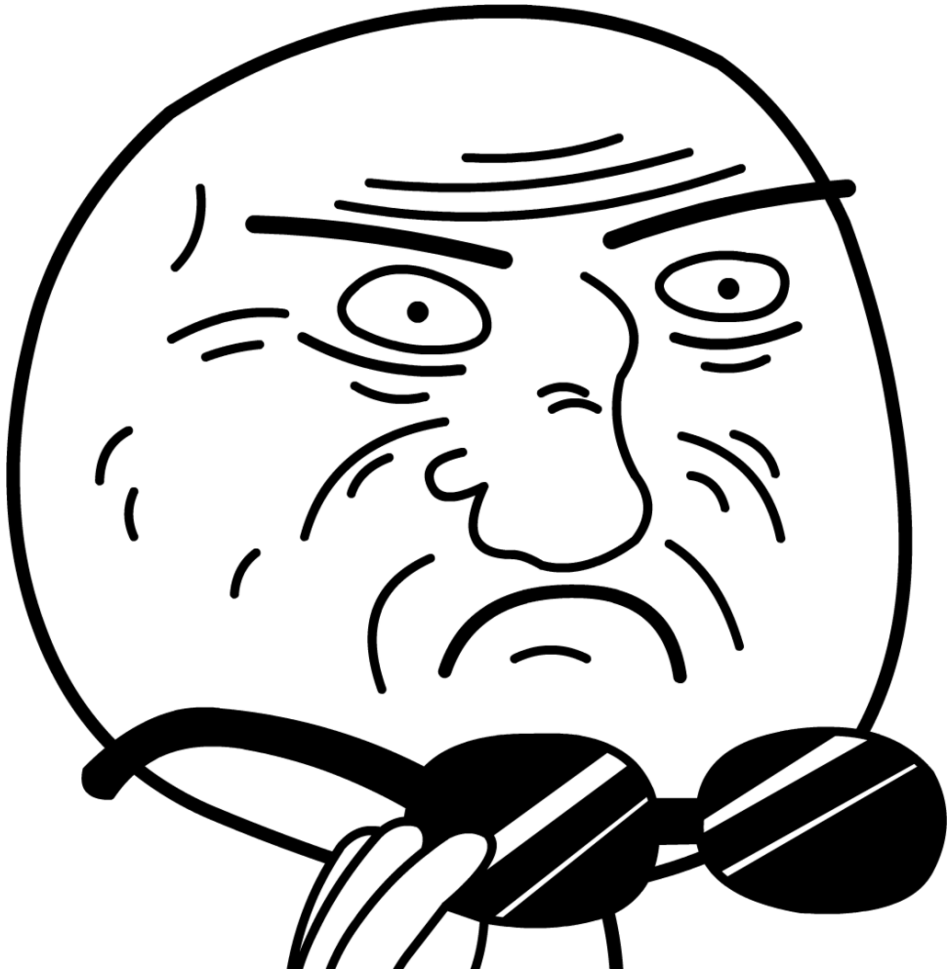
- If $n = 0$ then this is obvious.
- Otherwise, assume that $n = m + 1$. By the induction hypothesis, we know that there exists some p such that $m = 2p$ or $m = 2p + 1$.
  - In the first case, $n = 2p + 1$.
  - Otherwise $n = 2(p + 1)$.

The programmer

```
val div2 : int -> int * bool
(* [div2 n] returns the integer
division by 2 of [n] together with
a boolean indicating if [n] is
even. *)

let rec div2 n = match n with
| 0 -> (0, true)
| m + 1 ->
  let (p, even) = div2 m in
    if even then (p, false)
    else (p + 1, true)
```

An epiphany



That's the same thing.

The Curry-Howard isomorphism

Proofs in a given subset of mathematics are **exactly** programs from a particular language.

The Curry-Howard isomorphism

Proofs in a given subset of mathematics are **exactly** programs from a particular language.

- Discovered by Curry in '58, then by Howard in '69
- a.k.a. the "proof-as-program" correspondence
- Proofs compute! ♡

The Curry-Howard isomorphism

Proofs in a given subset of mathematics are **exactly** programs from a particular language.

- Discovered by Curry in '58, then by Howard in '69
- a.k.a. the "proof-as-program" correspondence
- Proofs compute! ♡

Furthermore, the statement of a theorem correspond to the type of the corresponding program.

- We need our programming language to be rich enough
- Programming languages in the wild have crappy type systems (if any)

> "Do not try to do maths with C++/Java/Python at home, kids."

A Rosetta's Stone

Logic	**CS**
Proofs	Programs
Formula	Types

A Rosetta's Stone

Logic	**CS**
Proofs	Programs
Formula	Types
A implies B	function from A to B
A and B	pair of A and B
A or B	tagged union of A and B
falsity	empty type
truth	singleton type
for all $x \in A$, $B(x)$	dependent product from A to B

A Rosetta's Stone

Logic	**CS**
Proofs	Programs
Formula	Types
A implies B	function from A to B
A and B	pair of A and B
A or B	tagged union of A and B
falsity	empty type
truth	singleton type
for all $x \in A$, $B(x)$	dependent product from A to B
Axiom	System primitive
Soundness theorem	Compiler
Completeness theorem	Debugger
Incompleteness theorem	Infinite loop

A paradigmatic shift

Proofs are relevant objects.

"Any list can be quotiented by permutations."

- Quicksort
- Mergesort
- Bogosort

A paradigmatic shift

Proofs are relevant objects.

"Any list can be quotiented by permutations."

- Quicksort
- Mergesort
- Bogosort

Proofs are first-class objects.

$$\forall (A : \mathtt{Set}).\, \forall (x : A).\, \forall (p\, q : x = x).\, p = q \qquad (?)$$

Which logic for which programs?

Standard members of each community will complain.

The mathematician	**The programmer**
"This logic is crappy, it does not feature the following principles I am acquainted with."	"This language is crappy, it does not feature the following structures I am acquainted with."

Either A or not A hold.
Every bounded monotone sequence has a limit.
Two sets with same elements are equal.
Every formula is equivalent to its prenex form.
...
	`printf("Hello world")`
	`x <- 42`
	`while true { … }`
	`goto #considered_harmful`
	`fork()`
	...

Which logic for which programs?

Standard members of each community will complain.

The mathematician	**The programmer**
"This logic is crappy, it does not feature the following principles I am acquainted with."	"This language is crappy, it does not feature the following structures I am acquainted with."

Either A or not A hold.
Every bounded monotone sequence has a limit.
Two sets with same elements are equal.
Every formula is equivalent to its prenex form.
...
	`printf("Hello world")`
	`x <- 42`
	`while true { … }`
	`goto #considered_harmful`
	`fork()`
	...

INTUITIONISTIC LOGIC	**FUNCTIONAL LANGUAGE**

A new hope

What Curry-Howard taketh, it giveth back.

A new hope

What Curry-Howard taketh, it giveth back.

New axioms	$\sim$	Programming primitives
$\Updownarrow$		$\Updownarrow$
Logical encoding	$\sim$	Program translation

The prototypical example: Classical logic

We can implement classical logic through this scheme.

New axiom		Programming primitives
$A \vee \neg A$	$\sim$	`callcc`
Used by your next-door mathematician		From Scheme, though a bit arcane
$\Updownarrow$		$\Updownarrow$
Logical encoding		Program translation
double-negation translation	$\sim$	callback-passing style
Gödel already did that by the 30's		The dullest JS programmer uses this

The prototypical example: Classical logic

We can implement classical logic through this scheme.

New axiom		Programming primitives
$A \vee \neg A$	$\sim$	`callcc`
Used by your next-door mathematician		From Scheme, though a bit arcane
$\Updownarrow$		$\Updownarrow$
Logical encoding		Program translation
double-negation translation	$\sim$	callback-passing style
Gödel already did that by the 30's		The dullest JS programmer uses this

What does it do, computationally?

A little tale

Morale

- Did the Devil lie?

Morale

- Did the Devil lie?
- That's not clear!
- Ensure you agree on the semantics before bargaining with the Devil.
- Hint: that's a parable.

Morale

- Did the Devil lie?
- That's not clear!
- Ensure you agree on the semantics before bargaining with the Devil.
- Hint: that's a parable.

Program translations can change the expected behaviour

Morale

- Did the Devil lie?
- That's not clear!
- Ensure you agree on the semantics before bargaining with the Devil.
- Hint: that's a parable.

Program translations can change the expected behaviour

...and it can unvalidate some other principles

- Axiom of choice is trivial in intuitionistic logic
- Axiom of choice is a monster from outer space in classical logic

An ever-growing field

The back-and-forth journey between world is fruitful.

- Double negation $\sim$ first-class callbacks
- Friedman's translation $\sim$ dynamic exceptions
- Cohen's forcing $\sim$ global variables
- Dialectica translation $\sim$ scoped gotos (*à la* Python's `yield`)

An ever-growing field

The back-and-forth journey between world is fruitful.

- Double negation $\sim$ first-class callbacks
- Friedman's translation $\sim$ dynamic exceptions
- Cohen's forcing $\sim$ global variables
- Dialectica translation $\sim$ scoped gotos (*à la* Python's `yield`)

...and there is more

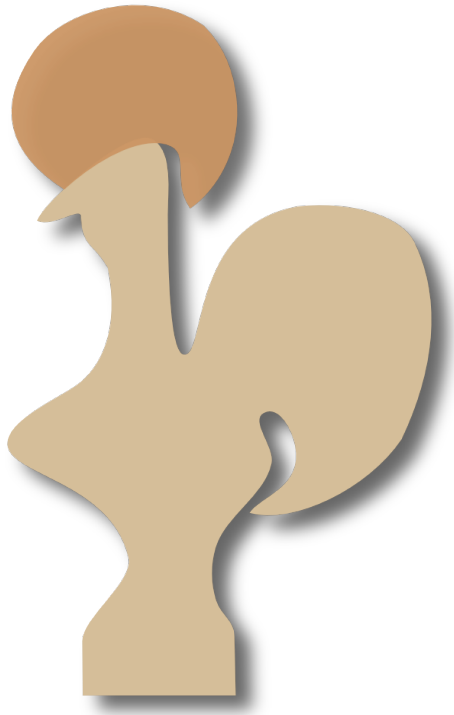
- *Trending topic*: HoTT
- Proofs as paths, types as homotopy spaces
- Discovered by Voedvodsky (2002 Fields medal)

WARNING

In the following next slides, you are going to be exposed to blatant advertising material. You have been advised.

Coq

The *fine fleur* of the Curry-Howard isomorphism.



- **Your 2-in-1 proof assistant!**
  - A programming language
  - A mathematical system
- Developped by πr^2 team
- Est. 1984 (Coquand, Huet, Paulin...)
- 2013 ACM Software System Award

  Previously: LLVM, Unix, TCP/IP, TeX, Apache, Java...
- Version 8.5 beta just released.

A success story

Coq is increasingly used in **The Real World®**.

Maths	**CS**
• Four-Colour Theorem	• CompCert (Gallium)
• Feit-Thomson Theorem	• Bedrock
• HoTT/Coq	• There is even a blog engine written in Coq!

A success story

Coq is increasingly used in **The Real World®**.

Maths	**CS**
• Four-Colour Theorem	• CompCert (Gallium)
• Feit-Thomson Theorem	• Bedrock
• HoTT/Coq	• There is even a blog engine written in Coq!

Program your proofs | Prove your programs

A success story

Coq is increasingly used in **The Real World®**.

Maths	**CS**
• Four-Colour Theorem	• CompCert (Gallium)
• Feit-Thomson Theorem	• Bedrock
• HoTT/Coq	• There is even a blog engine written in Coq!

Program your proofs | Prove your programs

Try it out today!

The Curry-Howard isomorphism: It's a Revolution™

- A hindsightful way to look at logic
  - Logic and computation unified
  - A theory of Everything

The Curry-Howard isomorphism: It's a Revolution™

- A hindsightful way to look at logic
  - Logic and computation unified
  - A theory of Everything
- Mathematics are entering a new era
  - "The Golden Digital Era of Mathematics"
  - Leveraging the power of computers
  - Towards a wikipedia of formalized mathematics?

The Curry-Howard isomorphism: It's a Revolution™

- A hindsightful way to look at logic
  - Logic and computation unified
  - A theory of Everything
- Mathematics are entering a new era
  - "The Golden Digital Era of Mathematics"
  - Leveraging the power of computers
  - Towards a wikipedia of formalized mathematics?
- Food for thought
  - Why aren't mathematics closed-source?
  - What is a mathematical bug?
  - Is a iTheorem really cooler than a WinTheorem32?
  - Do Python coders truly believe that $2 + 2 = 5$?

Scribitur ad narrandum, non ad probandum

Any question?