Curry-Howard: unveiling the computational content of proofs

Étienne Miquey

Équipe Gallinette, INRIA
LS2N, Université de Nantes

23/11/2018

What I am *not* going to tell you

This talk is secretly a personal challenge.

A tricky question

Every Ph.D. student has been asked a thousand times:

"What is the title of your thesis?"

Here is mine:

Classical realizability and side-effects

The next questions:

- classical?
- realizability?
- side-effects?
- *What does it have to do with logic/mathematics/computer science?*

A tricky question

Every Ph.D. student has been asked a thousand times:

"What is the title of your thesis?"

Here is mine:

Classical realizability and side-effects

The next questions:

- classical?
- realizability?
- side-effects?
- *What does it have to do with logic/mathematics/computer science?*

A tricky question

Every Ph.D. student has been asked a thousand times:

"What is the title of your thesis?"

Here is mine:

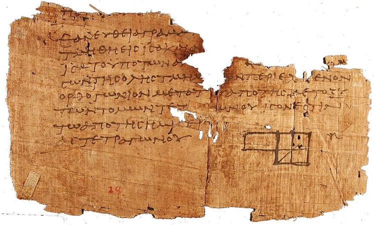
Classical realizability and side-effects

The next questions:

- classical?
- realizability?
- side-effects?
- *What does it have to do with logic/mathematics/computer science?*

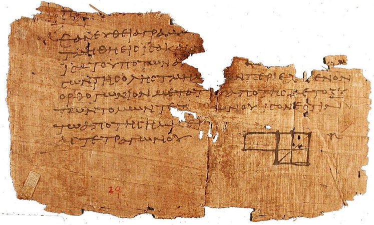
Proofs

A (very) old one:



Proofs

A (very) old one:

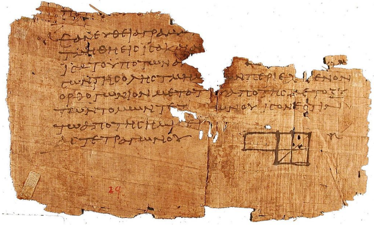


An easy one:

Plato is a cat.
All cats like fish.
Therefore, Plato likes fish .

Proofs

A (very) old one:



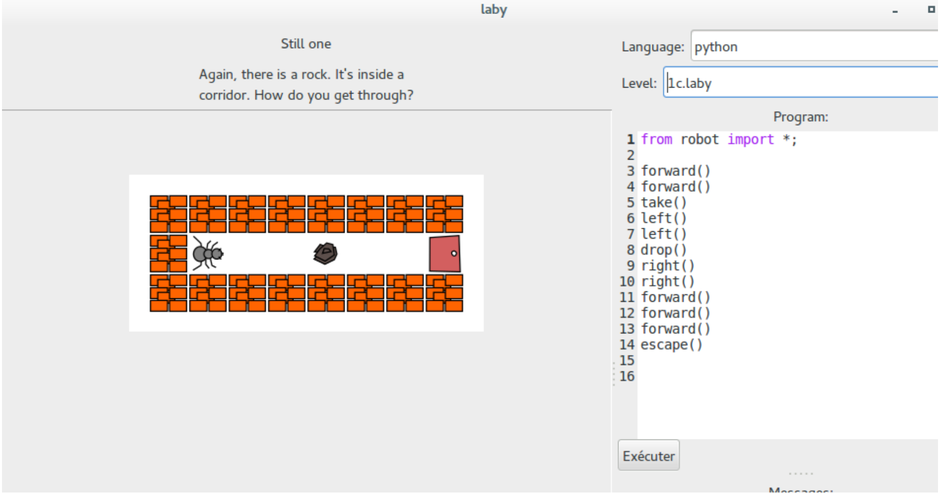
An easy one:

Plato is a cat.
All cats like fish.
Therefore, Plato likes fish .

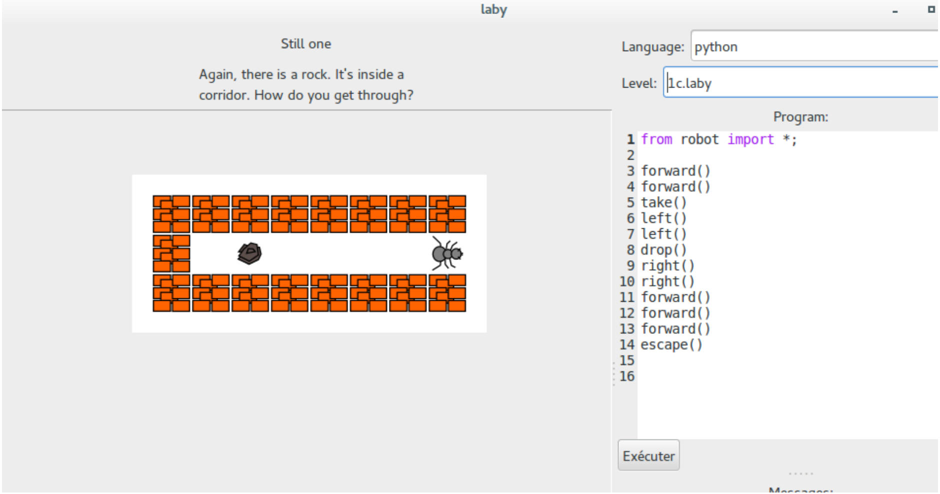
Intuitively:

from a set of **hypotheses**
apply **deduction rules**
to obtain a **theorem**

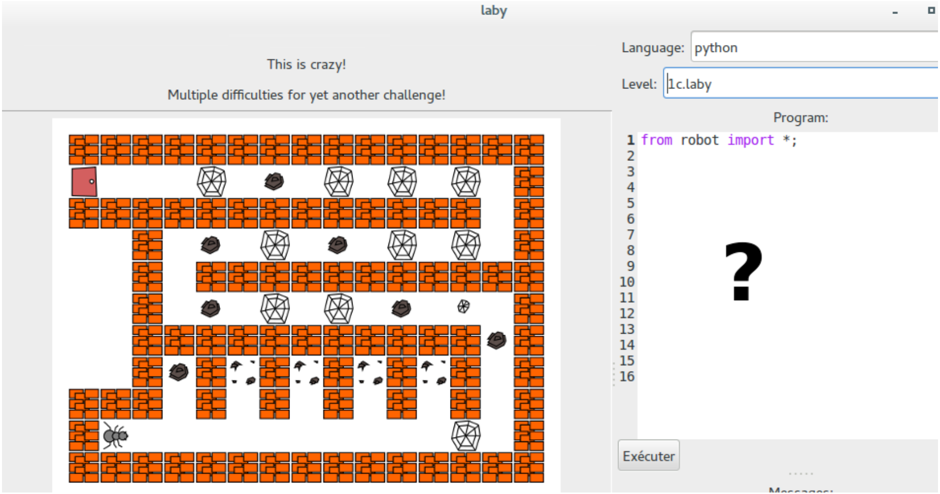
Programs


laby
Still one
Again, there is a rock. It's inside a corridor. How do you get through?
Language: python
Level: 1c.laby
Program:
from robot import *;
forward()
forward()
take()
left()
left()
drop()
right()
right()
forward()
forward()
forward()
escape()
Exécuter

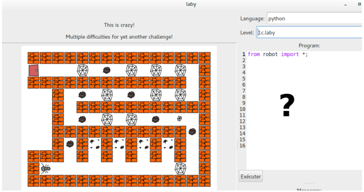
Programs


laby
Still one
Again, there is a rock. It's inside a corridor. How do you get through?
Language: python
Level: 1c.laby
Program:
from robot import *;
forward()
forward()
take()
left()
left()
drop()
right()
right()
forward()
forward()
forward()
escape()
Exécuter

Programs


laby
This is crazy!
Multiple difficulties for yet another challenge!
Language: python
Level: 1c.laby
Program:
from robot import *;
?
Exécuter

Programs



Think of it as a **recipe** (algorithm) to draw a computation forward.

Intuitively:

from a set of **inputs**

apply **instructions**

to obtain the **output**

So ?

Proof:

from a set of **hypotheses**

apply **deduction rules**

to obtain a **theorem**

Program:

from a set of **inputs**

apply **instructions**

to obtain the **output**

Curry-Howard

(On well-chosen subsets of mathematics and programs)

That's the same thing!

Proofs

(A bit of history)

Leibniz

A *combinatorial view* of human ideas, thinking that they

> *"can be resolved into a few as their primitives"*

Leibniz

A *combinatorial view* of human ideas, thinking that they

"can be resolved into a few as their primitives"

A crazy dream:

"when there are disputes among persons, we can simply say: Let us calculate, without further ado, to see who is right."

Geometry

Euclid's Elements: the first axiomatic presentation of geometry

- a collection of definitions (line, etc.)
- common notions ("things equal to the same thing are also equal to one another")
- five postulates ("to draw a straight-line from any point to any point")

> If a straight line crossing two straight lines makes the interior angles on the same side less than two right angles, the two straight lines, if extended indefinitely, meet on that side on which are the angles less than the two right angles.

19^{th} **century:** non-Euclidean geometries

- **Bolyai:** only four postulates
- **Lobachevsky:** four + negation of the fifth
- **Riemann:** four

How can it be determined that a theory is not contradictory?

Geometry

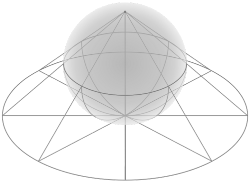
Euclid's Elements: the first axiomatic presentation of geometry

- a collection of definitions (line, etc.)
- common notions ("things equal to the same thing are also equal to one another")
- five postulates ("to draw a straight-line from any point to any point")

If a straight line crossing two straight lines makes the interior angles on the same side less than two right angles, the two straight lines, if extended indefinitely, meet on that side on which are the angles less than the two right angles.

19$^{\text{th}}$ century: non-Euclidean geometries

- **Bolyai:** only four postulates
- **Lobachevsky:** four + negation of the fifth
- **Riemann:** four



How can it be determined that a theory is not contradictory?

Geometry

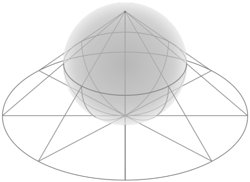
Euclid's Elements: the first axiomatic presentation of geometry

- a collection of definitions (line, etc.)
- common notions ("things equal to the same thing are also equal to one another")
- five postulates ("to draw a straight-line from any point to any point")

If a straight line crossing two straight lines makes the interior angles on the same side less than two right angles, the two straight lines, if extended indefinitely, meet on that side on which are the angles less than the two right angles.

19th century: non-Euclidean geometries

- **Bolyai:** only four postulates
- **Lobachevsky:** four + negation of the fifth
- **Riemann:** four



How can it be determined that a theory is not contradictory?

Frege

> *"One cannot serve the truth and the untruth. If Euclidean geometry is true, then non-Euclidean geometry is false."*

Begriffsschrift:

- formal notations
- quantifications ∀/∃
- distinction:

  x vs $'x'$

  signified *signifier*

Frege

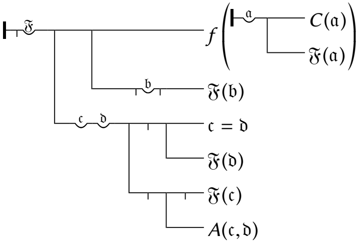
"One cannot serve the truth and the untruth. If Euclidean geometry is true, then non-Euclidean geometry is false."

Begriffsschrift:

- formal notations
- quantifications ∀/∃
- distinction:

  x vs $'x'$

  signified *signifier*



Proof trees (Gentzen)

Sequent:

Hypothesis $\boxed{\Gamma \vdash A}$ Conclusion

Deduction rules:

$$\frac{A \in \Gamma}{\Gamma \vdash A}(\mathrm{Ax}) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}(\Rightarrow_I) \qquad \frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}(\Rightarrow_E)$$

Example:

Proof trees (Gentzen)

Sequent:

Hypothesis $\boxed{\Gamma \vdash A}$ Conclusion

Deduction rules:

$$\frac{A \in \Gamma}{\Gamma \vdash A}(\text{Ax}) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}(\Rightarrow_I) \qquad \frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}(\Rightarrow_E)$$

Example:

Plato is a cat.

If Plato is cat, Plato likes fish.

Therefore, Plato likes fish. (Conclusion)

$$\frac{\dfrac{(A \Rightarrow B) \in \Gamma}{\Gamma \vdash A \Rightarrow B}(\text{Ax}) \qquad \dfrac{A \in \Gamma}{\Gamma \vdash A}(\text{Ax})}{\Gamma \vdash B}(\Rightarrow_E)$$

Proof trees (Gentzen)

Sequent:

Hypothesis $\boxed{\Gamma \vdash A}$ Conclusion

Deduction rules:

$$\frac{A \in \Gamma}{\Gamma \vdash A}(\mathrm{Ax}) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}(\Rightarrow_I) \qquad \frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}(\Rightarrow_E)$$

Example:

Hyp. { *Plato is a cat.*
If Plato is cat, Plato likes fish.

Therefore, $\underbrace{\textit{Plato likes fish}}_{\text{Conclusion}}$.

$$\frac{\dfrac{(A \Rightarrow B) \in \Gamma}{\Gamma \vdash A \Rightarrow B}(\mathrm{Ax}) \qquad \dfrac{A \in \Gamma}{\Gamma \vdash A}(\mathrm{Ax})}{\Gamma \vdash B}(\Rightarrow_E)$$

Theory

A *theory* is the given of:

- a **language**:

Terms	e_1, e_2	$::=$	$x \mid 0 \mid s(e) \mid e_1 + e_2 \mid e_1 \times e_2$
Formulas	A, B	$::=$	$e_1 = e_2 \mid \top \mid \bot \mid \forall x.A \mid \exists x.A \mid A \Rightarrow B \mid A \wedge B \mid A \vee B$

- a **deduction system**:

$$\frac{A \in \Gamma}{\Gamma \vdash A}(\text{Ax}) \qquad \frac{}{\Gamma \vdash \top}(\top) \qquad \frac{\Gamma \vdash \bot}{\Gamma \vdash A}(\bot) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}(\Rightarrow_I) \qquad \frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}(\Rightarrow_E)$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}(\wedge_I) \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}(\wedge_E^1) \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}(\wedge_E^2) \qquad \frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}(\vee_I^1) \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}(\vee_I^2)$$

$$\frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}(\vee_E) \qquad \frac{\Gamma \vdash A \quad x \notin FV(\Gamma)}{\Gamma \vdash \forall x.A}(\forall_I) \qquad \frac{\Gamma \vdash \forall x.A}{\Gamma \vdash A[t/x]}(\forall_E)$$

- a set of **axioms**:

(PA1)	$\forall x.(0 + x = x)$	(PA4)	$\forall x.\forall y.(s(x) \times y = (x \times y) + y)$
(PA2)	$\forall x.\forall y.(s(x) + y = s(x + y))$	(PA5)	$\forall x.\forall y.(s(x) = s(y) \Rightarrow x = y)$
(PA3)	$\forall x.(0 \times x = 0)$	(PA6)	$\forall x.(s(x) \neq 0)$
(E1)	$\forall x.(x = x)$	...	

Theory

A *theory* is the given of:

- a **language**:

Terms	$e_1, e_2 \ ::= \ x \mid 0 \mid s(e) \mid e_1 + e_2 \mid e_1 \times e_2$
Formulas	$A, B \ ::= \ e_1 = e_2 \mid \top \mid \bot \mid \forall x.A \mid \exists x.A \mid A \Rightarrow B \mid A \wedge B \mid A \vee B$

- a **deduction system**:

$$\frac{A \in \Gamma}{\Gamma \vdash A}(\mathrm{Ax}) \qquad \frac{}{\Gamma \vdash \top}(\top) \qquad \frac{\Gamma \vdash \bot}{\Gamma \vdash A}(\bot) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}(\Rightarrow_I) \qquad \frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}(\Rightarrow_E)$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B}(\wedge_I) \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash A}(\wedge_E^1) \qquad \frac{\Gamma \vdash A \wedge B}{\Gamma \vdash B}(\wedge_E^2) \qquad \frac{\Gamma \vdash A}{\Gamma \vdash A \vee B}(\vee_I^1) \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \vee B}(\vee_I^2)$$

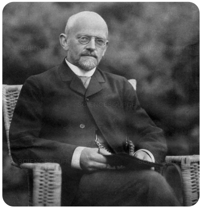
$$\frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}(\vee_E) \qquad \frac{\Gamma \vdash A \quad x \notin FV(\Gamma)}{\Gamma \vdash \forall x.A}(\forall_I) \qquad \frac{\Gamma \vdash \forall x.A}{\Gamma \vdash A[t/x]}(\forall_E)$$

- a set of **axioms**:

(PA1)	$\forall x.(0 + x = x)$	(PA4)	$\forall x.\forall y.(s(x) \times y = (x \times y) + y)$
(PA2)	$\forall x.\forall y.(s(x) + y = s(x + y))$	(PA5)	$\forall x.\forall y.(s(x) = s(y) \Rightarrow x = y)$
(PA3)	$\forall x.(0 \times x = 0)$	(PA6)	$\forall x.(s(x) \neq 0)$
(E1)	$\forall x.(x = x)$	...	

Programs

Hilbert's problems



Radio cast (1930):

> *For us mathematicians, there is no 'ignorabimus' [...] We must know — we shall know!*

Identified important mathematical problems to solve:

- 2nd Hilbert's problem:

  Prove the compatibility of the arithmetical axioms.

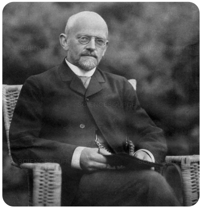
  ↪ Well, you all heard of Gödel, right?

- *Entscheidungsproblem* (with Ackermann):

  To decide if a formula of first-order logic is a tautology.

  ↪ "**to decide**" is meant via an algorithm, by means of a procedure

Hilbert's problems



Radio cast (1930):

> *For us mathematicians, there is no 'ignorabimus' [...] We must know — we shall know!*

Identified important mathematical problems to solve:

- 2nd Hilbert's problem:

> *Prove the compatibility of the arithmetical axioms.*

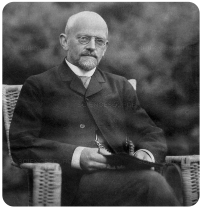
↬Well, you all heard of Gödel, right?

- *Entscheidungsproblem* (with Ackermann):

> *To decide if a formula of first-order logic is a tautology.*

↬ "**to decide**" is meant via an algorithm, by means of a procedure

Hilbert's problems



Radio cast (1930):

> *For us mathematicians, there is no 'ignorabimus' [...] We must know — we shall know!*

Identified important mathematical problems to solve:

- 2nd Hilbert's problem:

> *Prove the compatibility of the arithmetical axioms.*

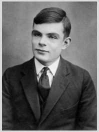
↬Well, you all heard of Gödel, right?

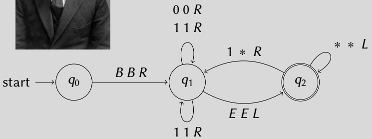
- *Entscheidungsproblem* (with Ackermann):

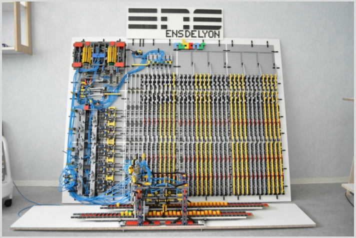
> *To decide if a formula of first-order logic is a tautology.*

↬ "**to decide**" is meant via an algorithm, by means of a procedure

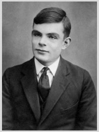
Turing machines

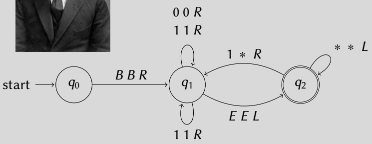


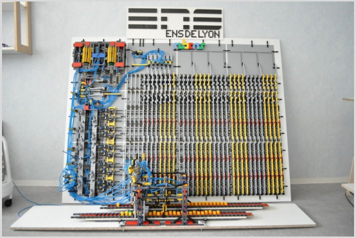




Turing machines







Halting problem: negative answer to the *Entscheidungsproblem*!

248 A. M. TURING [Nov. 12,

We can show further that *there can be no machine* $\mathcal{E}$ *which, when supplied with the S.D of an arbitrary machine* $\mathcal{M}$*, will determine whether* $\mathcal{M}$ *ever prints a given symbol* (0 *say*).

We will first show that, if there is a machine $\mathcal{E}$, then there is a general

The λ-calculus (1/2)

A **model** of computation (a.k.a. a *toy language*) due to **Alonzo Church** (1932)

The λ-calculus (1/2)

A **model** of computation (a.k.a. a *toy language*)
due to **Alonzo Church** (1932)

1936: first (negative) answer to the *Entscheidungsproblem* !

formula **C**, such that **A** conv 1 if and only if **C** has a normal form. From this
the lemma follows.

THEOREM XVIII. *There is no recursive function of a formula* **C**, *whose value is* 2 *or* 1 *according as* **C** *has a normal form or not.*

That is, the property of a well-formed formula, that it has a normal form

The λ-calculus (1/2)

A **model** of computation (a.k.a. a *toy language*) due to **Alonzo Church** (1932)

1936: first (negative) answer to the *Entscheidungsproblem* !

formula **C**, such that **A** conv I if and only if **C** has a normal form. From this the lemma follows.

THEOREM XVIII. *There is no recursive function of a formula* **C**, *whose value is* 2 *or* 1 *according as* **C** *has a normal form or not.*

That is, the property of a well-formed formula, that it has a normal form

Turing completeness

The λ-calculus and Turing machines are equivalent, *i.e.* they can compute the same partial functions from $\mathbb{N}$ to $\mathbb{N}$.

The λ-calculus (2/2)

Syntax:

$$t, u \quad ::= \quad x \quad | \quad \lambda x.t \quad | \quad t\,u$$

(variables) $\quad x \mapsto f(x) \quad f\,2$

Reduction

$$(\lambda x.t)\,u \longrightarrow_\beta t[u/x]$$

+ contextual closure: $C[t] \longrightarrow_\beta C[t']$ (if $t \longrightarrow_\beta t'$)

Examples:

$$(\lambda x.x)\,t \longrightarrow_\beta t$$

$$(\lambda x.\lambda y.y\,x)\,\overline{2}\,t \longrightarrow_\beta (\lambda y.y\,\overline{2})\,t \longrightarrow_\beta t\,\overline{2}$$

$$\omega = (\lambda x.x\,x)\,(\lambda x.x\,x) \longrightarrow_\beta (\lambda x.x\,x)\,(\lambda x.x\,x) \longrightarrow_\beta \ldots$$

$$(\lambda x.\lambda y.y)\,\omega\,\overline{2} \longrightarrow_\beta\ ?$$

The λ-calculus (2/2)

Syntax:

$$t, u \quad ::= \quad x \quad | \quad \lambda x.t \quad | \quad t\,u$$

(variables) $\quad x \mapsto f(x) \quad f\,2$

Reduction

$$(\lambda x.t)\,u \longrightarrow_\beta t[u/x]$$

+ contextual closure: $\quad C[t] \longrightarrow_\beta C[t'] \quad$ (if $t \longrightarrow_\beta t'$)

Examples:

$$(\lambda x.x)\,t \longrightarrow_\beta t$$

$$(\lambda x.\lambda y.y\,x)\,\bar{2}\,t \longrightarrow_\beta (\lambda y.y\,\bar{2})\,t \longrightarrow_\beta t\,\bar{2}$$

$$\omega = (\lambda x.x\,x)\,(\lambda x.x\,x) \longrightarrow_\beta (\lambda x.x\,x)\,(\lambda x.x\,x) \longrightarrow_\beta \dots$$

$$(\lambda x.\lambda y.y)\,\omega\,\bar{2} \longrightarrow_\beta\ ?$$

The λ-calculus (2/2)

Syntax:

$$t, u \quad ::= \quad x \quad | \quad \lambda x.t \quad | \quad t\,u$$

(variables) $\quad x \mapsto f(x) \quad f\,2$

Reduction

$$(\lambda x.t)\,u \longrightarrow_\beta t[u/x]$$

+ contextual closure: $C[t] \longrightarrow_\beta C[t']$ (if $t \longrightarrow_\beta t'$)

Examples:

$$(\lambda x.x)\,t \longrightarrow_\beta t$$

$$(\lambda x.\lambda y.y\,x)\,\bar{2}\,t \longrightarrow_\beta (\lambda y.y\,\bar{2})\,t \longrightarrow_\beta t\,\bar{2}$$

$$\omega = (\lambda x.x\,x)\,(\lambda x.x\,x) \longrightarrow_\beta (\lambda x.x\,x)\,(\lambda x.x\,x) \longrightarrow_\beta \dots$$

$$(\lambda x.\lambda y.y)\,\omega\,\bar{2} \longrightarrow_\beta\ ?$$

The λ-calculus (2/2)

Syntax:

$$t, u \quad ::= \quad x \quad | \quad \lambda x.t \quad | \quad t\, u$$

(variables) $\quad x \mapsto f(x) \quad f\,2$

Reduction

$$(\lambda x.t)\, u \longrightarrow_\beta t[u/x]$$

+ contextual closure: $\quad C[t] \longrightarrow_\beta C[t'] \quad$ (if $t \longrightarrow_\beta t'$)

Examples:

$$(\lambda x.x)\, t \longrightarrow_\beta t$$

$$(\lambda x.\lambda y.y\, x)\, \overline{2}\, t \longrightarrow_\beta (\lambda y.y\, \overline{2})\, t \longrightarrow_\beta t\, \overline{2}$$

$$\omega = (\lambda x.x\, x)\, (\lambda x.x\, x) \longrightarrow_\beta (\lambda x.x\, x)\, (\lambda x.x\, x) \longrightarrow_\beta \ldots$$

$$(\lambda x.\lambda y.y)\, \omega\, \overline{2} \longrightarrow_\beta \; ?$$

The λ-calculus (2/2)

Syntax:

$$t, u \quad ::= \quad x \quad | \quad \lambda x.t \quad | \quad t\,u$$

(variables) $\qquad x \mapsto f(x) \qquad f\,2$

Reduction

$$(\lambda x.t)\,u \longrightarrow_\beta t[u/x]$$

+ contextual closure: $\quad C[t] \longrightarrow_\beta C[t'] \quad$ (if $t \longrightarrow_\beta t'$)

Examples:

$$(\lambda x.x)\,t \longrightarrow_\beta t$$

$$(\lambda x.\lambda y.y\,x)\,\overline{2}\,t \longrightarrow_\beta (\lambda y.y\,\overline{2})\,t \longrightarrow_\beta t\,\overline{2}$$

$$\omega = (\lambda x.x\,x)\,(\lambda x.x\,x) \longrightarrow_\beta (\lambda x.x\,x)\,(\lambda x.x\,x) \longrightarrow_\beta \dots$$

$$(\lambda x.\lambda y.y)\,\omega\,\overline{2} \longrightarrow_\beta \ ?$$

The λ-calculus (2/2)

Syntax:

$$t, u \quad ::= \quad x \quad | \quad \lambda x.t \quad | \quad t\, u$$

(variables) $\quad x \mapsto f(x) \quad f\, 2$

Reduction

$$(\lambda x.t)\, u \longrightarrow_\beta t[u/x]$$

+ contextual closure: $C[t] \longrightarrow_\beta C[t']$ (if $t \longrightarrow_\beta t'$)

Examples:

$$(\lambda x.x)\, t \longrightarrow_\beta t$$

$$(\lambda x.\lambda y.y\, x)\, \overline{2}\, t \longrightarrow_\beta (\lambda y.y\, \overline{2})\, t \longrightarrow_\beta t\, \overline{2}$$

$$\omega = (\lambda x.x\, x)\, (\lambda x.x\, x) \longrightarrow_\beta (\lambda x.x\, x)\, (\lambda x.x\, x) \longrightarrow_\beta \dots$$

$$(\lambda x.\lambda y.y)\, \omega\, \overline{2} \longrightarrow_\beta ?$$

The λ-calculus (2/2)

Syntax:

$$t, u \quad ::= \quad x \quad | \quad \lambda x.t \quad | \quad t\,u$$

(variables) $\qquad x \mapsto f(x) \qquad f\,2$

Reduction

$$(\lambda x.t)\,u \longrightarrow_\beta t[u/x]$$

+ contextual closure: $\quad C[t] \longrightarrow_\beta C[t'] \quad$ (if $t \longrightarrow_\beta t'$)

Examples:

$$(\lambda x.x)\,t \longrightarrow_\beta t$$

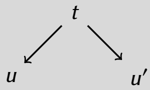
$$(\lambda x.\lambda y.y\,x)\,\overline{2}\,t \longrightarrow_\beta (\lambda y.y\,\overline{2})\,t \longrightarrow_\beta t\,\overline{2}$$

$$\omega = (\lambda x.x\,x)\,(\lambda x.x\,x) \longrightarrow_\beta (\lambda x.x\,x)\,(\lambda x.x\,x) \longrightarrow_\beta \ldots$$

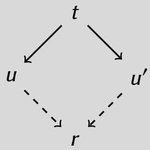
$$(\lambda x.\lambda y.y)\,\omega\,\overline{2} \longrightarrow_\beta \ ?$$

Theoretical questions

Determinism:



Confluence:



Normalization:

$$t \longrightarrow t' \longrightarrow t'' \overset{?}{\dashrightarrow} V \not\rightarrow$$

Types

Goal:

eliminate unwanted behaviour

Simple types: $A, B \ ::= \ X \ | \ A \to B$

$\mathbb{N}$ $\quad$ $\mathbb{R} \to \mathbb{N}$

Sequent:

Hypothesis $\boxed{\Gamma \vdash t : A}$ Conclusion

Typing rules:

$$\frac{(x : A) \in \Gamma}{\Gamma \vdash x : A}\ (\mathrm{Ax}) \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}\ (\to_I) \qquad \frac{\Gamma \vdash t : A \to B \quad \Gamma \vdash u : A}{\Gamma \vdash t\, u : B}\ (\to_E)$$

Types

Simple types: $A, B \ ::= \ X \ \mid \ A \to B$

$\mathbb{N} \qquad \mathbb{R} \to \mathbb{N}$

Sequent:

Hypothesis $\boxed{\Gamma \vdash t : A}$ Conclusion

Typing rules:

$$\frac{(x : A) \in \Gamma}{\Gamma \vdash x : A}\ (\mathrm{Ax}) \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}\ (\to_I) \qquad \frac{\Gamma \vdash t : A \to B \quad \Gamma \vdash u : A}{\Gamma \vdash t\, u : B}\ (\to_E)$$

Example:

$$\vdash ? : (A \to B) \to (B \to C) \to (A \to C)$$

Types

Simple types:

$$A, B \quad ::= \quad \underset{\mathbb{N}}{X} \quad | \quad \underset{\mathbb{R} \to \mathbb{N}}{A \to B}$$

Sequent:

Hypothesis $\boxed{\Gamma \vdash t : A}$ Conclusion

Typing rules:

$$\frac{(x : A) \in \Gamma}{\Gamma \vdash x : A}\,(\text{Ax}) \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}\,(\to_I) \qquad \frac{\Gamma \vdash t : A \to B \quad \Gamma \vdash u : A}{\Gamma \vdash t\,u : B}\,(\to_E)$$

Example:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\cdots, g : (B \to C), \cdots \vdash g : B \to C}\,(\text{Ax}) \quad \dfrac{\dfrac{}{f : A \to B, \cdots \vdash f : A \to B}\,(\text{Ax}) \quad \dfrac{}{\cdots, x : A \vdash x : A}\,(\text{Ax})}{f : A \to B, \cdots, x : A \vdash f\,x : B}\,(\to_E)}{f : A \to B, g : (B \to C), x : A \vdash g\,(f\,x) : C}\,(\to_E)}{f : A \to B, g : (B \to C) \vdash \lambda x.g\,(f\,x) : (A \to C)}\,(\to_I)}{f : A \to B \vdash \lambda g.\lambda x.g\,(f\,x) : (B \to C) \to (A \to C)}\,(\to_I)}{\vdash \lambda f.\lambda g.\lambda x.g\,(f\,x) : (A \to B) \to (B \to C) \to (A \to C)}\,(\to_I)$$

Types

Simple types: $A, B \quad ::= \quad X \quad | \quad A \to B$

$\mathbb{N} \qquad \mathbb{R} \to \mathbb{N}$

Sequent:

Hypothesis $\boxed{\Gamma \vdash t : A}$ Conclusion

Typing rules:

$$\frac{(x : A) \in \Gamma}{\Gamma \vdash x : A}\ (\text{Ax}) \qquad \frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}\ (\to_I) \qquad \frac{\Gamma \vdash t : A \to B \quad \Gamma \vdash u : A}{\Gamma \vdash t\,u : B}\ (\to_E)$$

Properties:

Subject reduction

If $\Gamma \vdash t : A$ and $t \longrightarrow_\beta t'$, then $\Gamma \vdash t' : A$.

Normalization

If $\Gamma \vdash t : A$, then t normalizes.

The Curry-Howard correspondence

A somewhat obvious observation

Deduction rules

$$\frac{A \in \Gamma}{\Gamma \vdash A}\ (\text{Ax})$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\ (\rightarrow_I)$$

$$\frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}\ (\rightarrow_E)$$

Typing rules

$$\frac{(x : A) \in \Gamma}{\Gamma \vdash x : A}\ (\text{Ax})$$

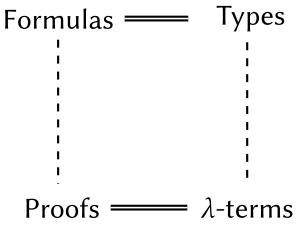
$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \rightarrow B}\ (\rightarrow_I)$$

$$\frac{\Gamma \vdash t : A \rightarrow B \quad \Gamma \vdash u : A}{\Gamma \vdash t\, u : B}\ (\rightarrow_E)$$

A somewhat obvious observation

Deduction rules	**Typing rules**
$\dfrac{A \in \Gamma}{\Gamma \vdash A}\ (\mathrm{Ax})$	$\dfrac{(x : A) \in \Gamma}{\Gamma \vdash x : A}\ (\mathrm{Ax})$
$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\ (\rightarrow_I)$	$\dfrac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \rightarrow B}\ (\rightarrow_I)$
$\dfrac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}\ (\rightarrow_E)$	$\dfrac{\Gamma \vdash t : A \rightarrow B \quad \Gamma \vdash u : A}{\Gamma \vdash t\ u : B}\ (\rightarrow_E)$

Proofs-as-programs



Proofs-as-programs

The Curry-Howard correspondence

Mathematics	**Computer Science**
Proofs	Programs
Propositions	Types
Deduction rules	Typing rules
$\dfrac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}\ (\Rightarrow_E)$	$\dfrac{\Gamma \vdash t : A \to B \quad \Gamma \vdash u : A}{\Gamma \vdash t\,u : B}\ (\to_E)$
A implies B	function $A \to B$
A and B	pair of A and B
$\forall x \in A.B(x)$	dependent product $\Pi x : A.B$

Benefits:

Program your proofs!	*Prove your programs!*

Commercial break

File Edit Options Buffers Tools Coq Proof-General Holes Help

```
Require Import Utf8.
Set Implicit Argument.

Hypothesis Animals:Type.
Hypothesis plato: Animals.
Hypothesis IsCat : Animals →  Prop.
Hypothesis LikesFish : Animals →  Prop.


Theorem PlatoLikesFish  :
  (∀ (x:Animals), IsCat x → LikesFish x)
   → IsCat plato
   → LikesFish plato.
Proof.
  intros HCat Hplato.
  apply (HCat plato).
  apply Hplato.
Qed.

Print PlatoLikesFish.


Definition myproof:=
  λ (HCat: (∀ (x:Animals), IsCat x → LikesFish x)),
   λ (Hplato:IsCat plato),
    (HCat plato Hplato).

Check myproof.


Definition myproof2 A (a:A) (P1:A→Prop) (P2:A→Prop):=
  λ (t:∀x,P1 x→P2 x),
    λ (u:P1 a),
```

```
U:---  Plato.v       Top (15,21)    (Coq Script(1-) +2 Holes Abbrev Ovwrt)
Overwrite mode enabled
```

```
1 subgoal (ID 3)

  HCat : ∀ x : Animals, IsCat x → LikesFish x
  Hplato : IsCat plato
  ============================
  LikesFish plato
```

```
U:%%-  *goals*       All (6,0)      (Coq Goals +3 Abbrev)
```

Commercial break

For programmers:

Say "good bye" to verification, and "hello" to intrinsically correct programs!

For mathematicians:

Write true proofs of real maths!

(e.g. Feit-Thompson theorem)

For everybody:

Discover new ways of thinking of proofs!

Commercial break

For programmers:

Say "good bye" to verification, and "hello" to intrinsically correct programs!

For mathematicians:

Write true proofs of real maths!

(e.g. Feit-Thompson theorem)

For everybody:

Discover new ways of thinking of proofs!

Commercial break

For programmers:

Say "good bye" to verification, and "hello" to intrinsically correct programs!

For mathematicians:

Write true proofs of real maths!

(e.g. Feit-Thompson theorem)

For everybody:

Discover new ways of thinking of proofs!

Bad news

Yet a lot of things are missing

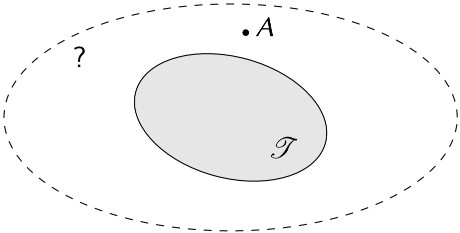
Limitations

Mathematics	Computer Science
$A \vee \neg A$	`try... catch ...`
$\neg\neg A \Rightarrow A$	`x := 42`
All sets can be well-ordered	`random()`
Sets that have the same elements are equal	`stop`
	`goto`

↬ *We want more !*

Non-constructive principles	*Side-effects*

Extending Curry-Howard

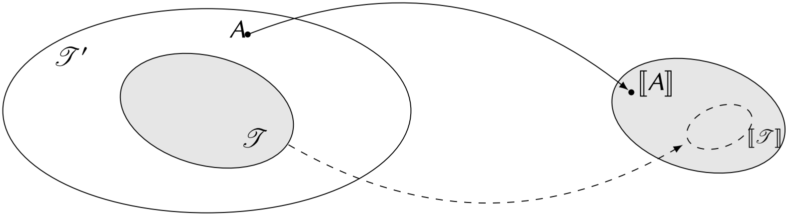


New axiom ~ Programing primitive

⇕ ⇕

Logical translation ~ Program translation

Extending Curry-Howard

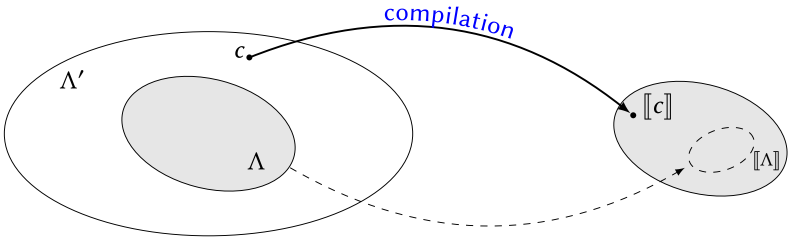


New axiom ~ Programing primitive

⇕ ⇕

Logical translation ~ Program translation

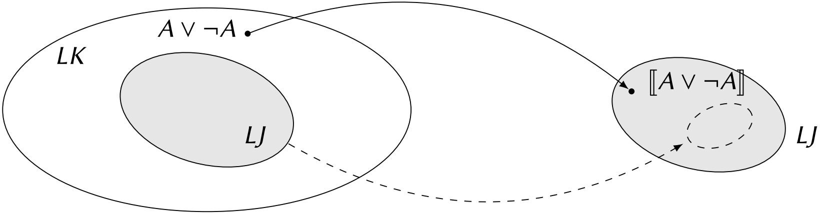
Extending Curry-Howard



New axiom	$\sim$	Programing primitive
$\Updownarrow$		$\Updownarrow$
Logical translation	$\sim$	Program translation

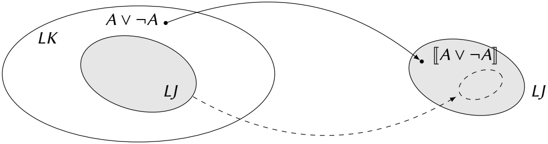
Classical logic

Classical logic = Intuitionistic logic + $A \lor \neg A$



Classical logic

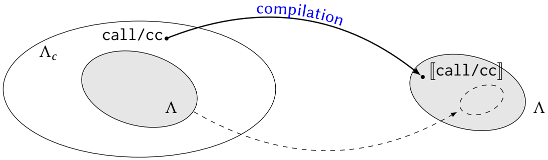
Classical logic = Intuitionistic logic + $A \vee \neg A$



New axiom		Programing primitive
$A \vee \neg A$	$\sim$	call/cc
Who doesn't use it?		Backtracking operator
$\Updownarrow$		$\Updownarrow$
Logical translation		Program translation
$A \mapsto \neg\neg A$	$\sim$	$2 \mapsto \lambda k.k\,2$
Gödel's negative translation		Continuation-passing style translation

Classical logic

Classical logic = Intuitionistic logic + $A \lor \neg A$



New axiom		Programing primitive
$A \lor \neg A$	$\sim$	call/cc
Who doesn't use it?		Backtracking operator
$\Updownarrow$		$\Updownarrow$
Logical translation		Program translation
$A \mapsto \neg\neg A$	$\sim$	$2 \mapsto \lambda k.k\,2$
Gödel's negative translation		Continuation-passing style translation

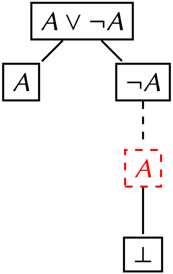
Computational content of classical logic

What is a program for $A \vee (A \to \bot)$?

In the pure λ-calculus:

- $A \vee B \rightsquigarrow$ *choose* one side and give a proof
- $A \to B \rightsquigarrow$ given a proof of A, computes a proof of B

Which side to choose?



Extension: call/cc allows us to *backtrack*!

1. Create a backtrack point
2. Play right: $A \to \bot$
3. Given a proof t of A, go back to 1
4. Play left: A
5. Give t

$$\mathtt{em} \triangleq \mathtt{call/cc}\,(\lambda k.\mathtt{inr}(\lambda t.k\,\mathtt{inl}(t)))$$

Computational content of classical logic

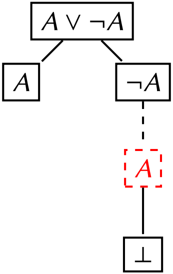
What is a program for $A \vee (A \to \bot)$?

In the pure λ-calculus:

- $A \vee B$ $\rightsquigarrow$ *choose* one side and give a proof
- $A \to B$ $\rightsquigarrow$ given a proof of A, computes a proof of B

Extension: `call/cc` allows us to *backtrack*!

1. Create a backtrack point
2. Play right: $A \to \bot$
3. Given a proof t of A, go back to 1
4. Play left: A
5. Give t



$$\mathtt{em} \triangleq \mathtt{call/cc}\,(\lambda k.\mathtt{inr}(\lambda t.k\,\mathtt{inl}(t)))$$

Computational content of classical logic

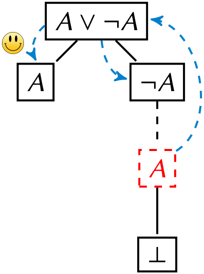
What is a program for $A \vee (A \to \bot)$?

In the pure λ-calculus:

- $A \vee B \rightsquigarrow$ *choose* one side and give a proof
- $A \to B \rightsquigarrow$ given a proof of A, computes a proof of B

Extension: `call/cc` allows us to *backtrack*!

1. Create a backtrack point
2. Play right: $A \to \bot$
3. Given a proof t of A, go back to 1
4. Play left: A
5. Give t



$\mathrm{em} \triangleq \texttt{call/cc}\,(\lambda k.\texttt{inr}(\lambda t.k\,\texttt{inl}(t)))$

Computational content of classical logic

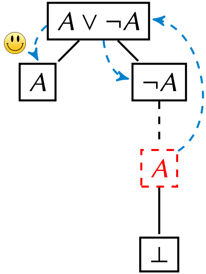
What is a program for $A \lor (A \to \bot)$?

In the pure λ-calculus:

- $A \lor B \rightsquigarrow$ *choose* one side and give a proof
- $A \to B \rightsquigarrow$ given a proof of A, computes a proof of B

Extension: `call/cc` allows us to *backtrack*!

1. Create a backtrack point
2. Play right: $A \to \bot$
3. Given a proof t of A, go back to 1
4. Play left: A
5. Give t



$$\text{em} \triangleq \texttt{call/cc}\,(\lambda k.\texttt{inr}(\lambda t.k\,\texttt{inl}(t)))$$

Logical content of a memory cell

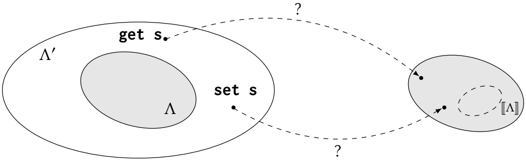
What does a memory cell bring to *the logic*?

Any idea?

Logical content of a memory cell

What does a memory cell bring to *the logic*?

Examine the compilation process !

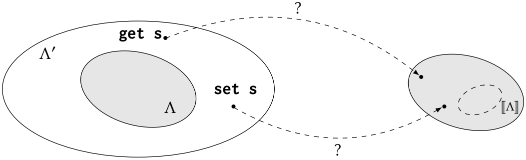


New axiom ?	∼	Programing primitive
⇕		⇕
Logical translation ?	∼	Program translation ?

Logical content of a memory cell

What does a memory cell bring to *the logic*?

Examine the compilation process !



First approximation, *state monad*:

$$[\![A \to B]\!] \triangleq \mathcal{S} \times [\![A]\!] \to \mathcal{S} \times [\![B]\!]$$

If besides the reference evolves **monotonically**:

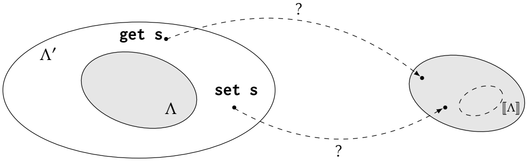
$$[\![A \to B]\!]_S \triangleq \forall S' \succcurlyeq S.\ [\![A]\!]_{S'} \to [\![B]\!]_{S'}$$
$$\omega \Vdash A \Rightarrow B \triangleq \forall \omega' \succcurlyeq \omega.\ \omega' \Vdash A \Rightarrow \omega' \Vdash B$$

↬ *forcing translation!*

Logical content of a memory cell

What does a memory cell bring to *the logic*?

Examine the compilation process !



First approximation, *state monad*:

$$[\![A \to B]\!] \triangleq \mathcal{S} \times [\![A]\!] \to \mathcal{S} \times [\![B]\!]$$

If besides the reference evolves **monotonically**:

$$[\![A \to B]\!]_S \triangleq \forall S' \succcurlyeq S.\ [\![A]\!]_{S'} \to [\![B]\!]_{S'}$$

$$\omega \Vdash A \Rightarrow B \triangleq \forall \omega' \succcurlyeq \omega.\ \omega' \Vdash A \Rightarrow \omega' \Vdash B$$

↬ *forcing translation!*

A new way of life

The motto

With side-effects come new reasoning principles.

In my thesis, I used several **computational features**:

- dependent types
- streams
- lazy evaluation
- shared memory

to get a **proof** for the axioms of **dependent and countable choice** that is compatible with **classical logic**.

Key idea

Memoization of choice functions through the stream of their values.

Classical realizability

Theory vs Model

What is the status of axioms (*e.g.* $A \vee \neg A$)?

↬ neither true nor false in the ambient theory

(here, *true* means *provable*)

There is another point of view:

- **Theory**: *provability* in an axiomatic representation (syntax)
- **Model**: *validity* in a particular structure (semantic)

Example:

$A \wedge B$		
A \ B	✓	✗
✓	✓	✗
✗	✗	✗

$A \vee B$		
A \ B	✓	✗
✓	✓	✓
✗	✓	✗

A	$\neg A$	$A \vee \neg A$
✓	✗	✓
✗	✓	✓

Theory vs Model

What is the status of axioms (*e.g.* $A \lor \neg A$)?

↬ neither true nor false in the ambient theory
(here, *true* means *provable*)

There is another point of view:

- **Theory**: *provability* in an axiomatic representation (syntax)
- **Model**: *validity* in a particular structure (semantic)

Example:

$A \land B$		
A \ B	✓	✗
✓	✓	✗
✗	✗	✗

$A \lor B$		
A \ B	✓	✗
✓	✓	✓
✗	✓	✗

A	$\neg A$	$A \lor \neg A$
✓	✗	✓
✗	✓	✓

Theory vs Model

What is the status of axioms (*e.g.* $A \vee \neg A$)?

↬ neither true nor false in the ambient theory
(here, *true* means *provable*)

There is another point of view:

- **Theory**: *provability* in an axiomatic representation (syntax)
- **Model**: *validity* in a particular structure (semantic)

Example:

$A \wedge B$		
A \ B	✓	✗
✓	✓	✗
✗	✗	✗

$A \vee B$		
A \ B	✓	✗
✓	✓	✓
✗	✓	✗

A	$\neg A$	$A \vee \neg A$
✓	✗	✓
✗	✓	✓

Theory vs Model

What is the status of axioms (*e.g.* $A \lor \neg A$)?

↬ neither true nor false in the ambient theory

(here, *true* means *provable*)

There is another point of view:

- **Theory**: *provability* in an axiomatic representation (syntax)
- **Model**: *validity* in a particular structure (semantic)

Example:

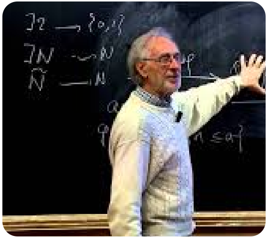
$A \land B$		
A \ B	✓	✗
✓	✓	✗
✗	✗	✗

$A \lor B$		
A \ B	✓	✗
✓	✓	✓
✗	✓	✗

A	$\neg A$	$A \lor \neg A$
✓	✗	✓
✗	✓	✓

Valid formula

Krivine classical realizability



Classical realizability:

$$A \mapsto \{t : t \Vdash A\}$$

(intuition: programs that share a common computational behavior given by A)

Great news

Classical realizability semantics gives surprisingly new models!

Realizability *à la* Krivine

Intuition

- falsity value $\|A\|$: contexts, opponent to A
- truth value $|A|$: proofs, player of A
- pole $\perp\!\!\!\perp$: commands, referee

$$\langle p \,\|\, e\rangle \succ c_0 \succ \cdots \succ c_n \in \perp\!\!\!\perp ?$$

$\rightsquigarrow \perp\!\!\!\perp \subset \Lambda \star \Pi$ closed by anti-reduction

Truth value defined by **orthogonality** :

$$|A| = \|A\|^{\perp\!\!\!\perp} = \{p \in \Lambda : \forall e \in \|A\|, \langle p \,\|\, e\rangle \in \perp\!\!\!\perp\}$$

Realizability *à la* Krivine

Intuition

- falsity value $\|A\|$: contexts, opponent to A
- truth value $|A|$: proofs, player of A
- pole $\perp\!\!\!\perp$: commands, referee

$$\langle p \| e \rangle > c_0 > \cdots > c_n \in \perp\!\!\!\perp ?$$

$\rightsquigarrow \perp\!\!\!\perp \subset \Lambda \star \Pi$ closed by anti-reduction

Truth value defined by **orthogonality** :

$$|A| = \|A\|^{\perp\!\!\!\perp} = \{p \in \Lambda : \forall e \in \|A\|, \langle p \| e \rangle \in \perp\!\!\!\perp\}$$

Realizability *à la* Krivine

Intuition

- falsity value $\|A\|$: contexts, opponent to A
- truth value $|A|$: proofs, player of A
- pole $\perp\!\!\!\perp$: commands, referee

$$\langle p \parallel e \rangle \succ c_0 \succ \cdots \succ c_n \in \perp\!\!\!\perp ?$$

$\rightsquigarrow \perp\!\!\!\perp \subset \Lambda \star \Pi$ closed by anti-reduction

Truth value defined by **orthogonality** :

$$|A| = \|A\|^{\perp\!\!\!\perp} = \{p \in \Lambda : \forall e \in \|A\|, \langle p \parallel e \rangle \in \perp\!\!\!\perp\}$$

Realizability *à la* Krivine

Intuition

- falsity value $\|A\|$: contexts, opponent to A
- truth value $|A|$: proofs, player of A
- pole $\perp\!\!\!\perp$: commands, referee

$$\langle p \,\|\, e\rangle \succ c_0 \succ \cdots \succ c_n \in \perp\!\!\!\perp ?$$

$\rightsquigarrow \perp\!\!\!\perp \subset \Lambda \star \Pi$ closed by anti-reduction

Truth value defined by **orthogonality** :

$$|A| = \|A\|^{\perp\!\!\!\perp} = \{p \in \Lambda : \forall e \in \|A\|, \langle p \,\|\, e\rangle \in \perp\!\!\!\perp\}$$

Realizability *à la* Krivine

Intuition

- falsity value $\|A\|$: contexts, opponent to A
- truth value $|A|$: proofs, player of A
- pole $⫫$: commands, referee

$$\langle p \parallel e \rangle \succ c_0 \succ \cdots \succ c_n \in ⫫ ?$$

$\rightsquigarrow ⫫ \subset \Lambda \star \Pi$ closed by anti-reduction

Truth value defined by **orthogonality** :

$$|A| = \|A\|^{⫫} = \{p \in \Lambda : \forall e \in \|A\|, \langle p \parallel e \rangle \in ⫫\}$$

Realizability *à la* Krivine

Intuition

- falsity value $\|A\|$: contexts, opponent to A
- truth value $|A|$: proofs, player of A
- pole $\perp\!\!\!\perp$: commands, referee

$$\langle p \,\|\, e \rangle \succ c_0 \succ \cdots \succ c_n \in \perp\!\!\!\perp ?$$

$\rightsquigarrow \perp\!\!\!\perp \subset \Lambda \star \Pi$ closed by anti-reduction

Truth value defined by **orthogonality** :

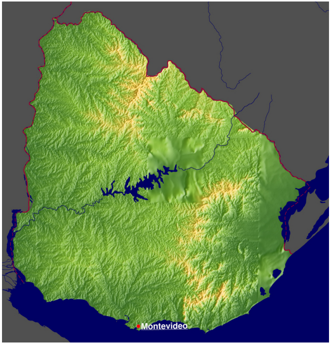
$$|A| = \|A\|^{\perp\!\!\!\perp} = \{p \in \Lambda : \forall e \in \|A\|, \langle p \,\|\, e \rangle \in \perp\!\!\!\perp\}$$

Results

One key lemma:

Adequacy

$$\text{If } \Gamma \vdash t : A \text{ then } t \in |A|$$



Typing **Realizability**

Results

One key lemma:

Adequacy

If $\Gamma \vdash t : A$ then $t \in |A|$

Plenty of consequences:

Normalization

Typed terms normalize.

Proof. $\perp\!\!\!\perp_{\Downarrow} \triangleq \{c : c \text{ normalizes}\}$ *defines a valid pole.* □

Soundness

There is no proof p such that $\vdash p : \bot$.

Proof. Otherwise, $p \in |\bot| = \Pi^{\perp\!\!\!\perp}$ *for any pole, absurd* ($\perp\!\!\!\perp \triangleq \emptyset$). □

Model theory

$$\mathcal{M}_{⫫} \vDash A \quad \Leftrightarrow \quad \exists t, t \in |A|$$

First feature:

Classical realizability can simulate any forcing construction!

A puzzling fact:

$\forall x.\mathrm{Nat}(x)$ is not realized in general

There exists a model where $\nabla_n \triangleq \{x : x < n\}$ verifies:

1. ∇_2 is not well-ordered
2. there is an injection from ∇_n to ∇_{n+1}
3. there is no surjection from ∇_n to ∇_{n+1}
4. $\nabla_m \times \nabla_n \simeq \nabla_{mn}$

In particular: $\vDash \neg AC$ and $\vDash \neg CH$

Implicative algebras

Great news, again

The algebraic analysis of the models that classical realizability induces can be done within simple structures.

Implicative structures

Complete meet-semilattice $(\mathcal{A}, \preccurlyeq, \rightarrow)$ s.t.:

- if $a_0 \preccurlyeq a$ and $b \preccurlyeq b_0$ then $(a \rightarrow b) \preccurlyeq (a_0 \rightarrow b_0)$ *(Variance)*
- $\curlywedge_{b \in B}(a \rightarrow b) = a \rightarrow \curlywedge_{b \in B} b$ *(Distributivity)*

- Generalize Heyting/Boolean algebras
- Generalize combinatory algebras
- Sound encoding of λ-terms
- Give rise to realizability triposes

Implicative algebras

Great news, again

The algebraic analysis of the models that classical realizability induces can be done within simple structures.