Curry-Howard: unveiling the computational content of proofs

Étienne Miquey

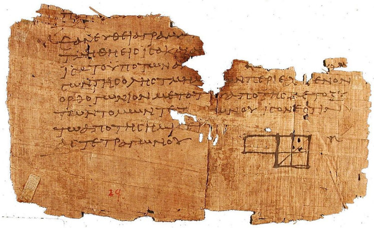
Séminaire CANA, 15/11/2022

I2M, Université Aix-Marseille

Introduction

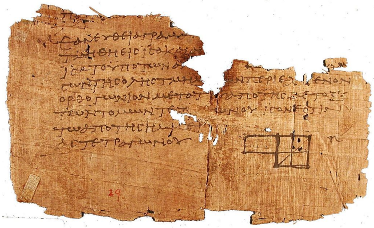
Proofs

A (very) old one:



Proofs

A (very) old one:



An easy one:

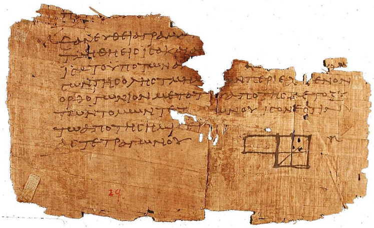
Plato is a cat.

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**

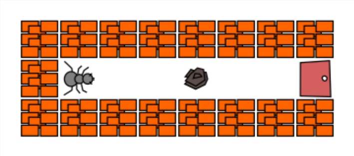
to obtain a **theorem**

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:

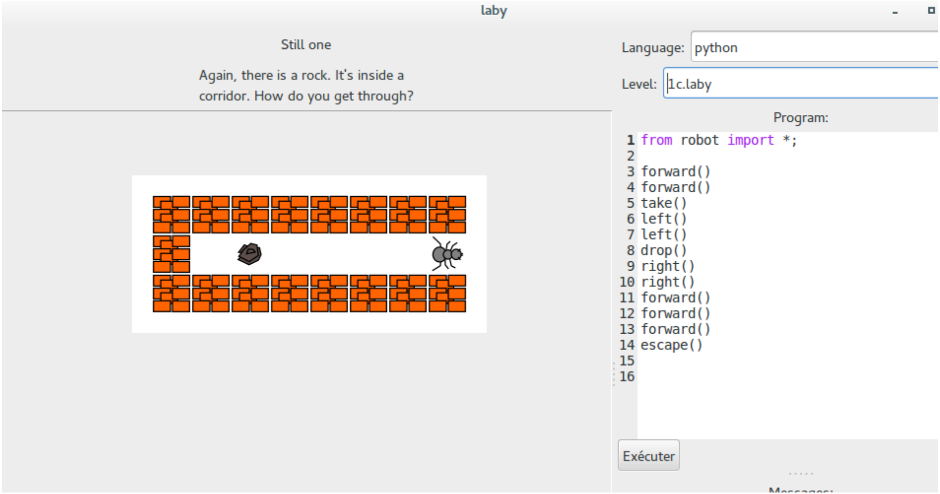
```
from robot import *;

forward()
forward()
take()
left()
left()
drop()
right()
right()
forward()
forward()
forward()
escape()
```

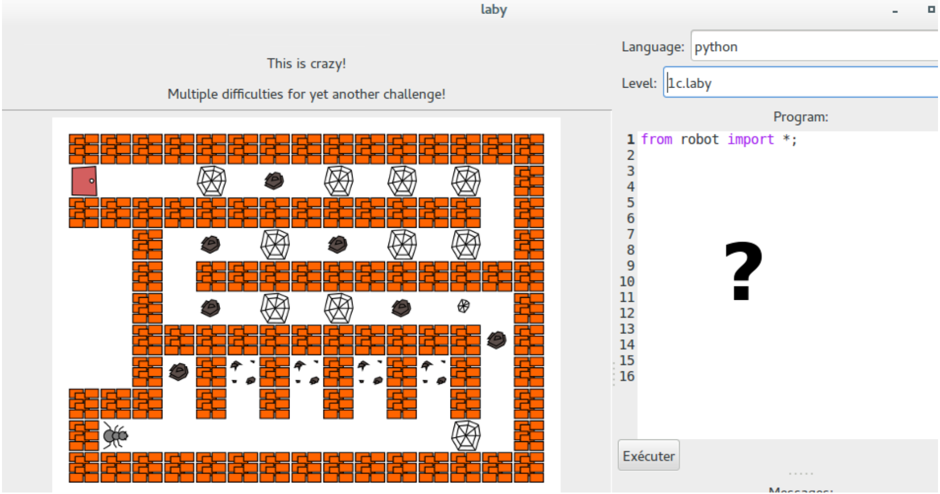
Exécuter

Messages:

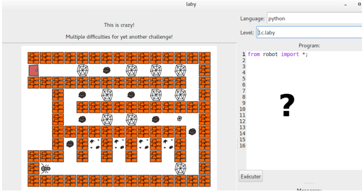
Programs


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

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.

19th century: non-Euclidean geometries

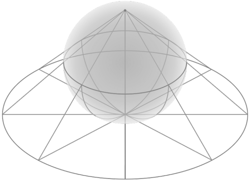
- **Bolyai:** only four postulates
- **Lobachevsky:** four + negation of the fifth
- **Riemann:** four

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

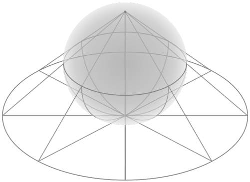
- **Bolyai:** only four postulates
- **Lobachevsky:** four + negation of the fifth
- **Riemann:** four

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

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

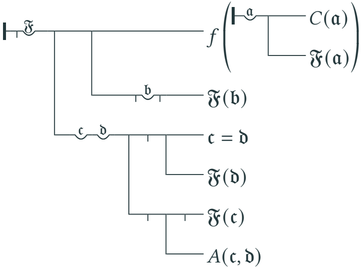
"One cannot serve the truth and the untruth. If Euclidean geometry is true, then non-Euclidean geometry is false."

Begriffsschrift:

- formal notations
- quantifications ∀/∃
- distinction:

  x vs $'x'$

  signified *signifier*



Proof trees (Gentzen)

Sequent:

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

Deduction rules:

$$\frac{A \in \Gamma}{\Gamma \vdash A}(\mathsf{Ax}) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}(\to_I) \qquad \frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}(\to_E)$$

Example:

Proof trees (Gentzen)

Sequent:

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

Deduction rules:

$$\frac{A \in \Gamma}{\Gamma \vdash A}(\mathsf{Ax}) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}(\to_I) \qquad \frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}(\to_E)$$

Example:

Plato is a cat.

If Plato is cat, Plato likes fish.

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

$$\dfrac{\dfrac{(A \Rightarrow B) \in \Gamma}{\Gamma \vdash A \Rightarrow B}(\mathsf{Ax}) \quad \dfrac{A \in \Gamma}{\Gamma \vdash A}(\mathsf{Ax})}{\Gamma \vdash B}(\to_E)$$

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

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

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**:

$$\frac{A \in \Gamma}{\Gamma \vdash A}(\mathsf{Ax}) \qquad \frac{}{\Gamma \vdash \top}(\top) \qquad \frac{\Gamma \vdash \bot}{\Gamma \vdash A}(\bot) \qquad \frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}(\rightarrow_I) \qquad \frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}(\rightarrow_E)$$

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

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

- 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)	$\forall x.(s(x) \neq 0)$

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**:

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

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

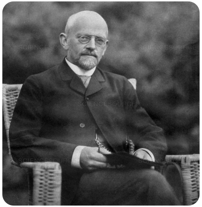
$$\frac{\Gamma \vdash A \vee B \quad \Gamma, A \vdash C \quad \Gamma, B \vdash C}{\Gamma \vdash C}(\vee_E) \qquad \frac{\Gamma \vdash A \quad x \notin FV(\Gamma)}{\Gamma \vdash \forall x.A}(\forall_I) \qquad \frac{\Gamma \vdash \forall x.A}{\Gamma \vdash A[t/x]}(\forall_E)$$

- 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)	$\forall x.(s(x) \neq 0)$

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*

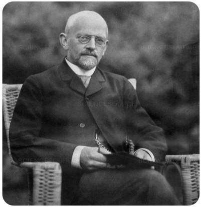
  ↪ Well, you all heard of Gödel, right?

- *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:

- 2^{nd} Hilbert's problem:

Prove the compatibility of the arithmetical axioms.

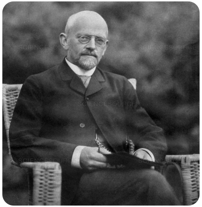
↬Well, you all heard of Gödel, right?

- *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.

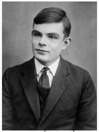
  ↬Well, you all heard of Gödel, right?

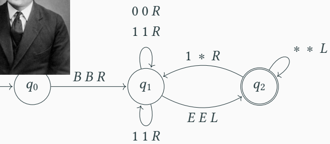
- *Entscheidungsproblem* (with Ackermann):

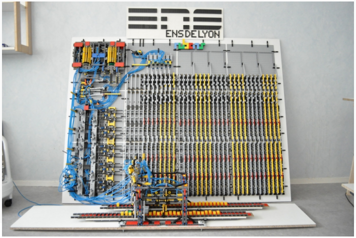
  To decide if a formula of first-order logic is a tautology.

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

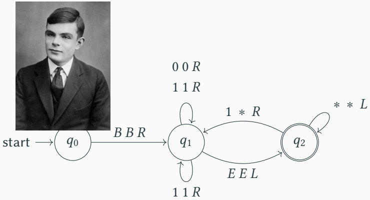
Turing machines

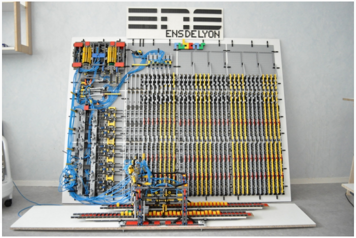






Turing machines





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 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

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:

$$t, u \quad ::= \quad \underset{\text{(variables)}}{x} \quad | \quad \underset{x \mapsto f(x)}{\lambda x.t} \quad | \quad \underset{f\,2}{t\,u}$$

Reduction

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

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

Examples:

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

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

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

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

The λ-calculus (2/2)

Syntax:

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

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

Reduction

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

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

Examples:

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

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

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

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

The λ-calculus (2/2)

Syntax:

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

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

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:

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

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

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

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

The λ-calculus (2/2)

Syntax:

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

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

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:

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

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

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

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

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:

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

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

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

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

The λ-calculus (2/2)

Syntax:

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

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

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:

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

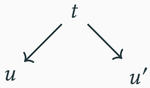
$$(\lambda x.\lambda y.y\,x)\,\bar{2}\,t \longrightarrow_\beta (\lambda y.y\,\bar{2})\,t \longrightarrow_\beta t\,\bar{2}$$

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

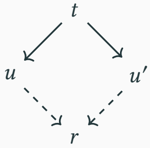
$$(\lambda x.\lambda y.y)\,\omega\,\bar{2} \longrightarrow_\beta \; ?$$

Theoretical questions

Determinism:



Confluence:



Normalization:

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

Types

Goal:

eliminate unwanted behaviour

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:

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

Types

Simple types:

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

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

Sequent:

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

Typing rules:

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

Example:

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

Types

Simple types:

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

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

Sequent:

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

Typing rules:

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

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.

Curry-Howard

A somewhat obvious observation

Deduction rules	Typing rules
$\dfrac{A \in \Gamma}{\Gamma \vdash A}\ (\mathsf{Ax})$	$\dfrac{(x : A) \in \Gamma}{\Gamma \vdash x : A}\ (\mathsf{Ax})$
$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}\ (\to_I)$	$\dfrac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}\ (\to_I)$
$\dfrac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}\ (\to_E)$	$\dfrac{\Gamma \vdash t : A \to B \quad \Gamma \vdash u : A}{\Gamma \vdash t\,u : B}\ (\to_E)$

A somewhat obvious observation

Deduction rules

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

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

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

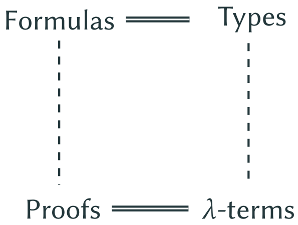
Typing rules

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

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

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

Proofs-as-programs



Proofs-as-programs

The Curry-Howard correspondence

Mathematics	Computer Science
Proofs	Programs
Propositions	Types
Deduction rules	Typing rules
$\dfrac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B}\ (\Rightarrow_E)$	$\dfrac{\Gamma \vdash t : A \rightarrow B \quad \Gamma \vdash u : A}{\Gamma \vdash t\, u : B}\ (\rightarrow_E)$
A implies B	function $A \rightarrow B$
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!*

Cut-elimination (Gentzen's *Hauptsatz*)

Question

How to compare these two proofs?

$$\dfrac{C \qquad \dfrac{A \vee B \quad \begin{matrix}[A] \\ \vdots \\ C \Rightarrow D\end{matrix} \quad \begin{matrix}[B] \\ \vdots \\ C \Rightarrow D\end{matrix}}{C \Rightarrow D}}{D} \qquad\qquad \dfrac{A \vee B \qquad \dfrac{C \quad \begin{matrix}[A] \\ \vdots \\ C \Rightarrow D\end{matrix}}{D} \qquad \dfrac{C \quad \begin{matrix}[B] \\ \vdots \\ C \Rightarrow D\end{matrix}}{D}}{D}$$

Question

How to prove that a proof system is consistent?

Cut-elimination (Gentzen's *Hauptsatz*)

Question

How to compare these two proofs?

$$\dfrac{C \qquad \dfrac{A \lor B \qquad \begin{matrix}[A]\\ \vdots \\ C \Rightarrow D\end{matrix} \qquad \begin{matrix}[B]\\ \vdots \\ C \Rightarrow D\end{matrix}}{C \Rightarrow D}}{D} \qquad\qquad \dfrac{A \lor B \qquad \dfrac{C \qquad \begin{matrix}[A]\\ \vdots \\ C \Rightarrow D\end{matrix}}{D} \qquad \dfrac{C \qquad \begin{matrix}[B]\\ \vdots \\ C \Rightarrow D\end{matrix}}{D}}{D}$$

Question

How to prove that a proof system is consistent?

Cut-elimination (Gentzen's *Hauptsatz*)

Cut admissibility

Every theorem has a cut-free proof.

or

Cut-elimination

Every proof can be *locally* transformed into one of the same theorem without cuts.

$$\dfrac{C \qquad \dfrac{A \vee B \qquad \begin{matrix}[A]\\ \vdots \\ C \Rightarrow D\end{matrix} \qquad \begin{matrix}[B]\\ \vdots \\ C \Rightarrow D\end{matrix}}{C \Rightarrow D}}{D} \quad \longrightarrow \quad \dfrac{A \vee B \qquad \dfrac{C \qquad \begin{matrix}[A]\\ \vdots \\ C \Rightarrow D\end{matrix}}{D} \qquad \dfrac{C \qquad \begin{matrix}[B]\\ \vdots \\ C \Rightarrow D\end{matrix}}{D}}{D}$$

$\looparrowright$ *lots of technicalities, motivation for Gentzen's sequent calculus.*

Cut-elimination (Gentzen's *Hauptsatz*)

Cut admissibility

Every theorem has a cut-free proof.

or

Cut-elimination

Every proof can be *locally* transformed into one of the same theorem without cuts.

$$\dfrac{C \qquad \dfrac{A \vee B \qquad \begin{matrix}[A]\\ \vdots \\ C \Rightarrow D\end{matrix} \qquad \begin{matrix}[B]\\ \vdots \\ C \Rightarrow D\end{matrix}}{C \Rightarrow D}}{D} \quad \longrightarrow \quad \dfrac{A \vee B \qquad \dfrac{C \qquad \begin{matrix}[A]\\ \vdots \\ C \Rightarrow D\end{matrix}}{D} \qquad \dfrac{C \qquad \begin{matrix}[B]\\ \vdots \\ C \Rightarrow D\end{matrix}}{D}}{D}$$

↬ *lots of technicalities, motivation for Gentzen's sequent calculus.*

Cut-elimination (Gentzen's *Hauptsatz*)

Cut admissibility

Every theorem has a cut-free proof.

or

Cut-elimination

Every proof can be *locally* transformed into one of the same theorem without cuts.

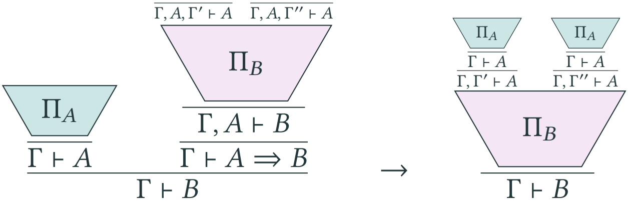
$$\dfrac{C \qquad \dfrac{A \vee B \quad \begin{matrix}[A]\\ \vdots \\ C \Rightarrow D\end{matrix} \quad \begin{matrix}[B]\\ \vdots \\ C \Rightarrow D\end{matrix}}{C \Rightarrow D}}{D} \qquad \rightarrow \qquad \dfrac{A \vee B \qquad \dfrac{C \quad \begin{matrix}[A]\\ \vdots \\ C \Rightarrow D\end{matrix}}{D} \qquad \dfrac{C \quad \begin{matrix}[B]\\ \vdots \\ C \Rightarrow D\end{matrix}}{D}}{D}$$

$\looparrowright$ *lots of technicalities, motivation for Gentzen's sequent calculus.*

Cut-elimination, computationally

Cut-elimination

Every proof can be *locally* transformed into one of the same theorem without cuts.



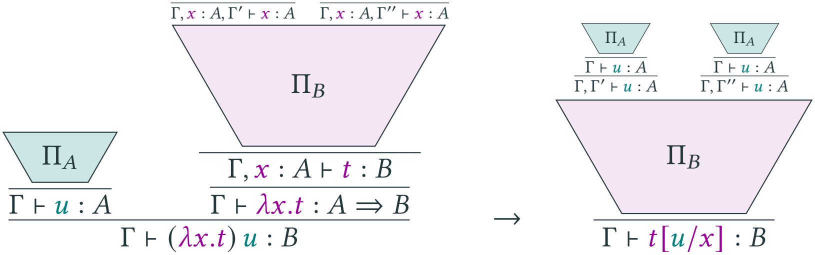
Cut-elimination

Every typed term normalizes.

Cut-elimination, computationally

Cut-elimination

Every proof can be *locally* transformed into one of the same theorem without cuts.



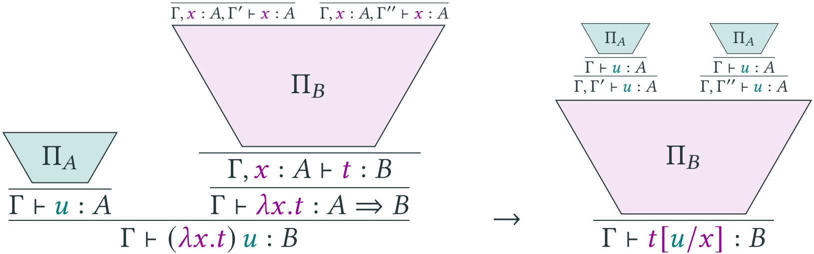
Cut-elimination

Every typed term normalizes.

Cut-elimination, computationally

Cut-elimination

Every proof can be *locally* transformed into one of the same theorem without cuts.



Cut-elimination

Every typed term normalizes.

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

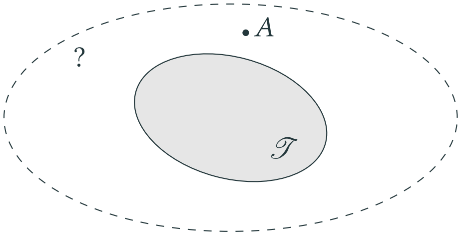
Limitations

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 !*

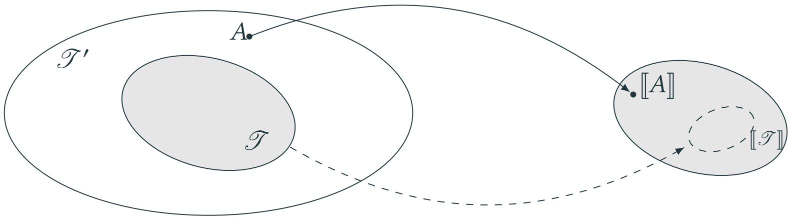
Non-constructive principles	*Side-effects*

Extending Curry-Howard



New axiom	~	Programing primitive
⇕		⇕
Logical translation	~	Program translation

Extending Curry-Howard

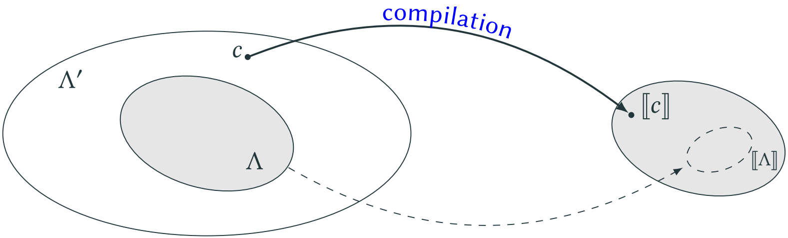


New axiom ~ Programing primitive

⇕ ⇕

Logical translation ~ Program translation

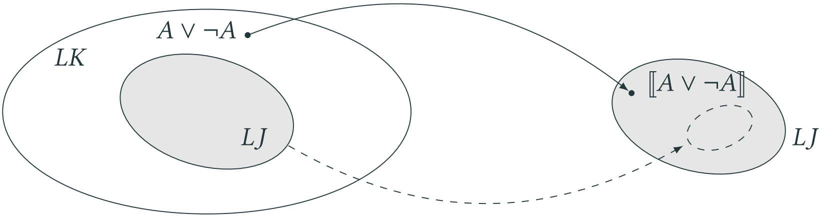
Extending Curry-Howard



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

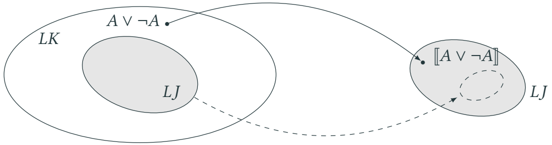
Classical logic

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



Classical logic

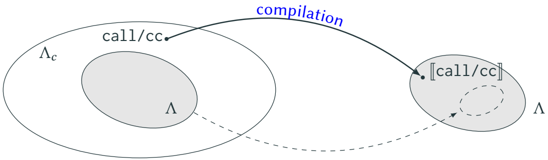
Classical logic = Intuitionistic logic + $A \vee \neg A$



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 + $A \vee \neg A$



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

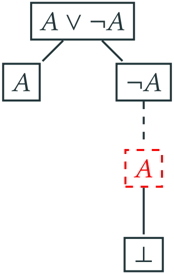
Computational content of classical logic

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

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

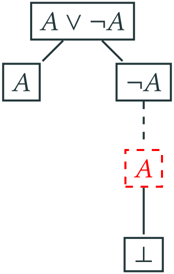
Computational content of classical logic

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

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

Computational content of classical logic

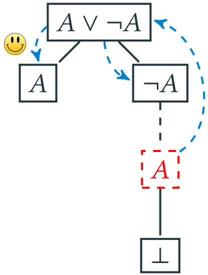
What is a program for $A \lor (A \to \bot)$?

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



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

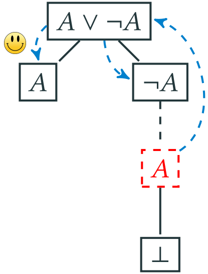
Computational content of classical logic

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

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

Logical content of a memory cell

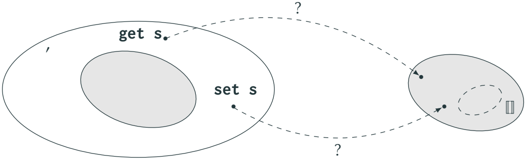
What does a memory cell bring to *the logic*?

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

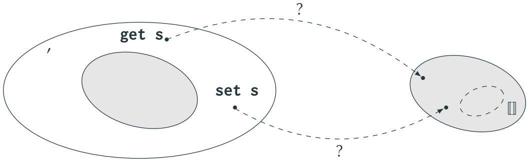
⇕ ⇕

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*:

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

If besides the reference evolves **monotonically**:

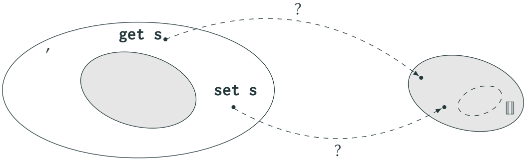
$$[\![A \rightarrow B]\!]_S \quad\triangleq\quad \forall S' \succcurlyeq S.\quad [\![A]\!]_{S'} \quad\supset\quad [\![B]\!]_{S'}$$
$$\omega \Vdash A \Rightarrow B \quad\triangleq\quad \forall \omega' \succcurlyeq \omega.\, \omega' \Vdash A \;\;\Rightarrow\;\; \omega' \Vdash B$$

↬ *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**:

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

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

↬ *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.

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.* $A \vee \neg A$)?

↬ 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.* $A \vee \neg A$)?

↬ 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.* $A \vee \neg A$)?

↬ 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	✓	✗
✓	✓	✓
✗	✓	✗

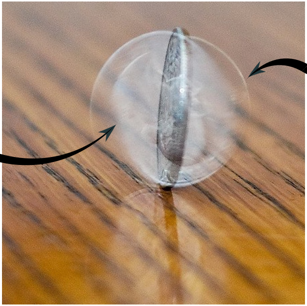
A	$\neg A$	$A \vee \neg A$
✓	✗	✓
✗	✓	✓

Valid formula

Realizability

Realizability

provides **models**
for theories



a **tool** to analyse
programs behavior

Realizability

$$t \quad \Vdash \quad A$$

$$\nearrow \qquad \uparrow \qquad \nwarrow$$

program *"realises"* *formula*

Intuitively

$t \Vdash A$ = "t computes (soundly) w.r.t. A"

The *type soundness* that we really want

Typing

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \to B}$$

(syntax, nothing but the syntax)

Realizability

$$t \Vdash A \to B \triangleq \forall u, (u \Vdash A \Rightarrow tu \vdash B)$$

(computations, nothing but computations)

Adequacy lemma

$$\vdash t : A \quad \Rightarrow \quad t \Vdash A$$

The *type soundness* that we really want

Typing

$$\frac{\Gamma, x : A \vdash t : B}{\Gamma \vdash \lambda x.t : A \longrightarrow B}$$

(syntax, nothing but the syntax)

Realizability

$$t \Vdash A \longrightarrow B \triangleq \forall u, (u \Vdash A \Rightarrow tu \vdash B)$$

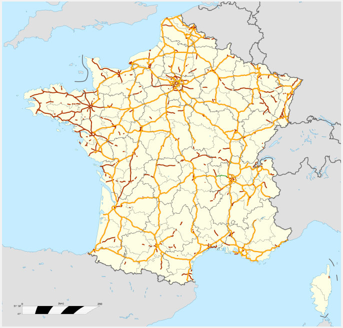
(computations, nothing but computations)

Adequacy lemma

$$\vdash t : A \quad \Rightarrow \quad t \Vdash A$$

The *type soundness* that we really want

Typing



Realizability

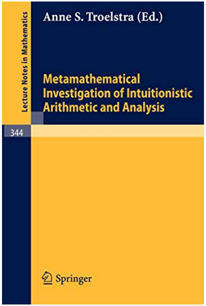
Adequacy lemma

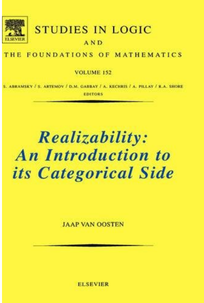
$$\vdash t : A \quad \Rightarrow \quad t \Vdash A$$

A *bouquet garni* of recipes

Programs	Theory
PCA	logique propositionelle
fonctions calculables	HA
Système F	HA2
$\text{CCI}_{\Box}$ (non-typé)	CCI
...	...

Cooking books





A 3-steps recipe

1. formulas (*a.k.a. types*)

   ↬ *simple types, 2^{nd} – order logic , ZF, ...*

2. a computational system (*a.k.a. your favorite calculus*)

   ↬ *some λ – calculus , a combinators algebra, PCF, etc.*

3. formulas interpretation

A 3-steps recipe

1. formulas (*a.k.a. types*)
   ↬ *simple types*, $2^{nd} - order\ logic$, *ZF, ...*

2. a computational system (*a.k.a. your favorite calculus*)
   ↬ *some* $\lambda - calculus$, *a combinators algebra, PCF, etc.*

3. formulas interpretation (*a.k.a. truth values*)
   ↬ $|A| = \{t \in \Lambda : t \Vdash A\}$

Adequacy

If $\mathfrak{p} : (\Gamma \vdash A)$ and $\sigma \Vdash \Gamma$ then $\sigma(\mathfrak{p}^*) \in |A|$.

A 3-steps recipe

1. formulas (*a.k.a. types*)

   ↬ *simple types*, $2^{nd} - order\ logic$, *ZF*, ...

2. a computational system (*a.k.a. your favorite calculus*)

   ↬ *some* $\lambda - calculus$, *a combinators algebra, PCF, etc.*

3. formulas interpretation (*a.k.a. truth values*)

   ↬ $|A| = \{t \in \Lambda : t \Vdash A\}$

Adequacy

If $\Gamma \vdash t : A$ and $\sigma \Vdash \Gamma$ then $\sigma(t) \in |A|$.

A simple realizability interpretation

Types & terms: (excerpt)

1st-order exp.	e	$::=$	$x \mid 0 \mid S(e) \mid f(e_1, \ldots, e_n)$
Formulas	A, B	$::=$	$\mathrm{Nat}(e) \mid X(e_1, \ldots, e_n) \mid A \to B \mid \ldots$
			$\mid \forall x.A \mid \exists x.A \mid \forall X.A \mid \exists X.A$
Terms	t, u	$::=$	$x \mid 0 \mid \mathbf{succ} \mid \mathbf{rec} \mid \lambda x.t \mid t\,u \mid \ldots$

where $f : \mathbb{N}^n \to \mathbb{N}$ is any arithmetical function.

Typing rules:

$$\frac{}{\Gamma \vdash 0 : \mathrm{Nat}(0)} \qquad \frac{}{\Gamma \vdash \mathbf{rec} : \forall Z.Z(0) \to (\forall^{\mathbb{N}} y.(Z(y) \to Z(S(y)))) \to \forall^{\mathbb{N}} x.Z(x)}$$

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

$$\frac{\Gamma \vdash t : A[x := n]}{\Gamma \vdash t : \exists x.A} \qquad \frac{\Gamma \vdash t : A[X(x_1, \ldots, x_n) := B]}{\Gamma \vdash t : \exists X.A}$$

A simple realizability interpretation

Types & terms: (excerpt)

1^{st}-order exp.	e	$::=$	$x \mid 0 \mid S(e) \mid f(e_1, \ldots, e_n)$
Formulas	A, B	$::=$	$\mathrm{Nat}(e) \mid X(e_1, \ldots, e_n) \mid A \to B \mid \ldots$
			$\mid \forall x.A \mid \exists x.A \mid \forall X.A \mid \exists X.A$
Terms	t, u	$::=$	$x \mid 0 \mid \mathbf{succ} \mid \mathbf{rec} \mid \lambda x.t \mid t\, u \mid \ldots$

where $f : \mathbb{N}^n \to \mathbb{N}$ is any arithmetical function.

Typing rules:

$$\ldots$$

Reductions:

$$\frac{}{(\lambda x.t)u \triangleright_\beta t[u/x]} \qquad \frac{}{\mathbf{rec}\, u_0\, u_1\, (\mathbf{succ}\, t) \triangleright_\beta u_1\, t\, (\mathbf{rec}\, u_0\, u_1\, t)} \qquad \cdots$$

A simple realizability interpretation

Realizability interpretation:

$$
\begin{aligned}
|\mathrm{Nat}(e)|_\rho &\triangleq \{t \in \Lambda : t \triangleright^* \mathbf{succ}^n 0, \text{ where } n = [\![e]\!]_\rho\} \\
|X(e_1,\ldots,e_n)|_\rho &\triangleq \rho(X)([\![e_1]\!]_\rho,\ldots,[\![e_n]\!]_\rho) \\
|A \to B|_\rho &\triangleq \{t \in \Lambda : \forall u \in |A|_\rho .(t\,u \in |B|_\rho)\} \\
|\forall x.A|_\rho &\triangleq \textstyle\bigcap_{n\in\mathbb{N}} |A|_{\rho,x\leftarrow n} \\
|\exists x.A|_\rho &\triangleq \textstyle\bigcup_{n\in\mathbb{N}} |A|_{\rho,x\leftarrow n} \\
|\forall X.A|_\rho &\triangleq \textstyle\bigcap_{F:\mathbb{N}^k\to\,\mathbf{SAT}} |A|_{\rho,X\leftarrow F} \\
|\exists X.A|_\rho &\triangleq \textstyle\bigcup_{F:\mathbb{N}^k\to\,\mathbf{SAT}} |A|_{\rho,X\leftarrow F}
\end{aligned}
$$

Adequacy

If $\Gamma \vdash t : A$ and $\sigma \Vdash \Gamma$ then $\sigma(t) \in |A|$.

Key ideas:

- realizers

A simple realizability interpretation

Realizability interpretation:

$$
\begin{aligned}
|\mathrm{Nat}(e)|_\rho &\triangleq \{t \in \Lambda : t \triangleright^* \mathbf{succ}^n 0, \text{ where } n = [\![e]\!]_\rho\} \\
|X(e_1,\dots,e_n)|_\rho &\triangleq \rho(X)([\![e_1]\!]_\rho,\dots,[\![e_n]\!]_\rho) \\
|A \to B|_\rho &\triangleq \{t \in \Lambda : \forall u \in |A|_\rho . (t\,u \in |B|_\rho)\} \\
|\forall x.A|_\rho &\triangleq \textstyle\bigcap_{n\in\mathbb{N}} |A|_{\rho,x\leftarrow n} \\
|\exists x.A|_\rho &\triangleq \textstyle\bigcup_{n\in\mathbb{N}} |A|_{\rho,x\leftarrow n} \\
|\forall X.A|_\rho &\triangleq \textstyle\bigcap_{F:\mathbb{N}^k\to\,\mathbf{SAT}} |A|_{\rho,X\leftarrow F} \\
|\exists X.A|_\rho &\triangleq \textstyle\bigcup_{F:\mathbb{N}^k\to\,\mathbf{SAT}} |A|_{\rho,X\leftarrow F}
\end{aligned}
$$

Key ideas:

- realizers *compute*
- realizers *defend the validity* of their formula
- truth values are *saturated*: $t \triangleright^* t' \ \wedge\ t' \in |A| \implies t \in |A|$

A simple realizability interpretation

Realizability interpretation:

$$
\begin{aligned}
|\mathrm{Nat}(e)|_\rho &\triangleq \{t \in \Lambda : t \triangleright^* \mathbf{succ}^n 0, \text{ where } n = [\![e]\!]_\rho\} \\
|X(e_1, \ldots, e_n)|_\rho &\triangleq \rho(X)([\![e_1]\!]_\rho, \ldots, [\![e_n]\!]_\rho) \\
|A \to B|_\rho &\triangleq \{t \in \Lambda : \forall u \in |A|_\rho.(t\,u \in |B|_\rho)\} \\
|\forall x.A|_\rho &\triangleq \bigcap_{n\in\mathbb{N}} |A|_{\rho,x\leftarrow n} \\
|\exists x.A|_\rho &\triangleq \bigcup_{n\in\mathbb{N}} |A|_{\rho,x\leftarrow n} \\
|\forall X.A|_\rho &\triangleq \bigcap_{F:\mathbb{N}^k\to\,\mathbf{SAT}} |A|_{\rho,X\leftarrow F} \\
|\exists X.A|_\rho &\triangleq \bigcup_{F:\mathbb{N}^k\to\,\mathbf{SAT}} |A|_{\rho,X\leftarrow F}
\end{aligned}
$$

Key ideas:

- realizers *compute*
- realizers *defend the validity* of their formula
- truth values are *saturated*: $t \triangleright^* t' \;\wedge\; t' \in |A| \;\Rightarrow\; t \in |A|$

A simple realizability interpretation

Realizability interpretation:

$$
\begin{aligned}
|\mathrm{Nat}(e)|_\rho &\triangleq \{t \in \Lambda : t \triangleright^* \mathbf{succ}^n 0, \text{ where } n = [\![e]\!]_\rho\} \\
|X(e_1, \ldots, e_n)|_\rho &\triangleq \rho(X)([\![e_1]\!]_\rho, \ldots, [\![e_n]\!]_\rho) \\
|A \to B|_\rho &\triangleq \{t \in \Lambda : \forall u \in |A|_\rho .(t\, u \in |B|_\rho)\} \\
|\forall x.A|_\rho &\triangleq \bigcap_{n \in \mathbb{N}} |A|_{\rho, x \leftarrow n} \\
|\exists x.A|_\rho &\triangleq \bigcup_{n \in \mathbb{N}} |A|_{\rho, x \leftarrow n} \\
|\forall X.A|_\rho &\triangleq \bigcap_{F : \mathbb{N}^k \to \mathbf{SAT}} |A|_{\rho, X \leftarrow F} \\
|\exists X.A|_\rho &\triangleq \bigcup_{F : \mathbb{N}^k \to \mathbf{SAT}} |A|_{\rho, X \leftarrow F}
\end{aligned}
$$

Key ideas:

- realizers *compute*
- realizers *defend the validity* of their formula
- truth values are *saturated*: $t \triangleright^* t' \wedge t' \in |A| \implies t \in |A|$

A simple realizability interpretation

Realizability interpretation:

$$
\begin{array}{rcl}
|\mathrm{Nat}(e)|_\rho & \triangleq & \{t \in \Lambda : t \triangleright^* \mathbf{succ}^n 0, \text{ where } n = [\![e]\!]_\rho\} \\
|X(e_1, \ldots, e_n)|_\rho & \triangleq & \rho(X)([\![e_1]\!]_\rho, \ldots, [\![e_n]\!]_\rho) \\
|A \to B|_\rho & \triangleq & \{t \in \Lambda : \forall u \in |A|_\rho .(t\, u \in |B|_\rho)\} \\
|\forall x .A|_\rho & \triangleq & \bigcap_{n \in \mathbb{N}} |A|_{\rho, x \leftarrow n} \\
|\exists x .A|_\rho & \triangleq & \bigcup_{n \in \mathbb{N}} |A|_{\rho, x \leftarrow n} \\
|\forall X.A|_\rho & \triangleq & \bigcap_{F:\mathbb{N}^k \to \mathbf{SAT}} |A|_{\rho, X \leftarrow F} \\
|\exists X.A|_\rho & \triangleq & \bigcup_{F:\mathbb{N}^k \to \mathbf{SAT}} |A|_{\rho, X \leftarrow F}
\end{array}
$$

Key ideas:

- realizers *compute*
- realizers *defend the validity* of their formula
- truth values are *saturated*: $t \triangleright^* t' \wedge t' \in |A| \Rightarrow t \in |A|$

Evidenced Frame: the common denominator

$$(\ \Phi \ , \ E \ , \ \phi_1 \xrightarrow{e} \phi_2 \ )$$

propositions — evidences — e realizes $\phi_1 \to \phi_2$

Intuitively: a "specification" of the minimal structure

Evidenced Frame: the common denominator

$$\left(\Phi \; , \; E \; , \; \phi_1 \xrightarrow{e} \phi_2 \right)$$

propositions — evidences — e realizes $\phi_1 \rightarrow \phi_2$

Intuitively: a "specification" of the minimal structure

Evidenced Frame $(\Phi, E, \cdot \xrightarrow{\cdot} \cdot)$

Reflexivity. $e_{id} \in E$ s.t.:

- $\phi \xrightarrow{e_{id}} \phi$

Transitivity. $; \in E \times E \to E$ s.t.:

- $\phi_1 \xrightarrow{e} \phi_2 \wedge \phi_2 \xrightarrow{e'} \phi_3 \implies \phi_1 \xrightarrow{e\,;\,e'} \phi_3$

Top. $\top \in \Phi$ and $e_\top \in E$ s.t.:

- $\phi \xrightarrow{e_\top} \top$

Conjunction. $\wedge \in \Phi \times \Phi \to \Phi$, $\langle\!|\cdot,\cdot|\!\rangle \in E \times E \to E$, and $e_{fst}, e_{snd} \in E$ s.t.:

- $\phi_1 \wedge \phi_2 \xrightarrow{e_{fst}} \phi_1$
- $\phi_1 \wedge \phi_2 \xrightarrow{e_{snd}} \phi_2$
- $\phi \xrightarrow{e_1} \phi_1 \wedge \phi \xrightarrow{e_2} \phi_2 \Longrightarrow \phi \xrightarrow{\langle\!| e_1, e_2 |\!\rangle} \phi_1 \wedge \phi_2$

$\phi_1 \wedge \phi_2 \xrightarrow{e_{snd}} \phi_2 7$

Universal implication. $\supset \in \Phi \times \mathcal{P}(\Phi) \to \Phi$, $\lambda \in E \to E$, and $e_{eval} \in E$:

- $(\forall \phi \in \vec{\phi}.\ \phi_1 \wedge \phi_2 \xrightarrow{e} \phi) \implies \phi_1 \xrightarrow{\lambda e} \phi_2 \supset \vec{\phi}$
- $\forall \phi \in \vec{\phi}.[(\phi_1 \supset \vec{\phi}) \wedge \phi_1 \xrightarrow{e_{eval}} \phi]$

Evidenced Frame $(\Phi, E, \cdot \xrightarrow{\cdot} \cdot)$

Reflexivity. $e_{\text{id}} \in E$ s.t.:

- $\phi \xrightarrow{e_{\text{id}}} \phi$

Transitivity. $; \in E \times E \to E$ s.t.:

- $\phi_1 \xrightarrow{e} \phi_2 \wedge \phi_2 \xrightarrow{e'} \phi_3 \implies \phi_1 \xrightarrow{e\,;\,e'} \phi_3$

Top. $\top \in \Phi$ and $e_{\top} \in E$ s.t.:

- $\phi \xrightarrow{e_{\top}} \top$

Conjunction. $\wedge \in \Phi \times \Phi \to \Phi$, $\langle\!|\cdot,\cdot|\!\rangle \in E \times E \to E$, and $e_{\text{fst}}, e_{\text{snd}} \in E$ s.t.:

- $\phi_1 \wedge \phi_2 \xrightarrow{e_{\text{fst}}} \phi_1$
- $\phi_1 \wedge \phi_2 \xrightarrow{e_{\text{snd}}} \phi_2$
- $\phi \xrightarrow{e_1} \phi_1 \wedge \phi \xrightarrow{e_2} \phi_2 \implies \phi \xrightarrow{\langle\!| e_1, e_2 |\!\rangle} \phi_1 \wedge \phi_2$

Universal implication. $\supset \in \Phi \times \mathcal{P}(\Phi) \to \Phi$, $\lambda \in E \to E$, and $e_{\text{eval}} \in E$:

- $(\forall \phi \in \vec{\phi}.\ \phi_1 \wedge \phi_2 \xrightarrow{e} \phi) \implies \phi_1 \xrightarrow{\lambda e} \phi_2 \supset \vec{\phi}$
- $\forall \phi \in \vec{\phi}.[(\phi_1 \supset \vec{\phi}) \wedge \phi_1 \xrightarrow{e_{\text{eval}}} \phi]$

Models

Realizability model

$$\mathcal{M} \vDash A \quad \Leftrightarrow \quad \exists t . t \Vdash A$$

Categorically speaking

a *topos*

Construction

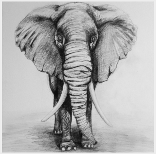
PCA / λ-calculus / … → Evidenced frame realizability → Topos model

Models

Realizability model

$$\mathcal{M} \vDash A \quad \Leftrightarrow \quad \exists t. t \Vdash A$$

Categorically speaking



a *topos*

Construction

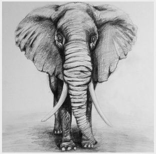
PCA / λ-calculus / … → Evidenced frame realizability → Topos model

Models

Realizability model

$$\mathcal{M} \vDash A \quad \Leftrightarrow \quad \exists t. t \Vdash A$$

Categorically speaking



a *topos*

Construction

PCA / λ-calculus / … → Evidenced frame (realizability) → Topos (model)

Why do we care?

One key lemma:

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

Plenty of consequences:

Normalization

Typed terms normalize.

Soundness

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

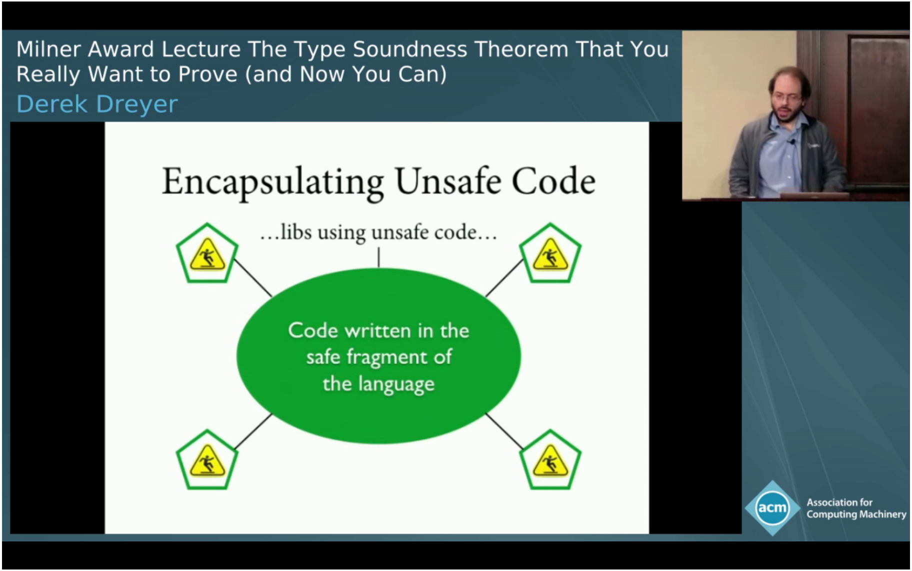
Witness extraction

If $\vdash t : \exists x^{\mathbb{N}}.f(x) = 0$ then we can compute n out of t s.t. $f(n) = 0$.

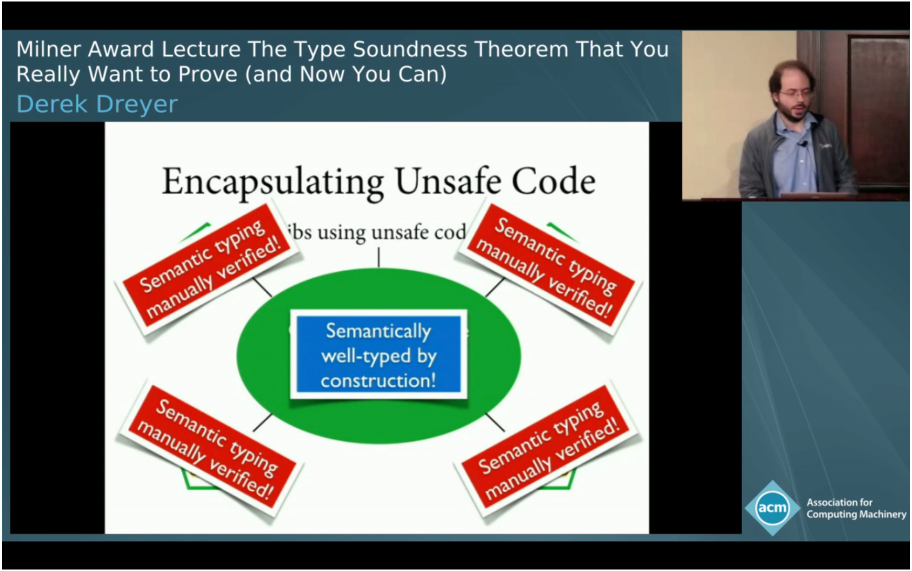
Specification

t realizes $\forall X.X \to X$ iff $C[t\,u] \to_\beta^* C[u]$

Modularity put in practice: Rust



Modularity put in practice: Rust



Models

Tarski

$$A \mapsto |A| \in \mathbb{B}$$

Boolean algebra

(intuition: level of truthness)

Realizability

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

(intuition: programs whose computational behavior is guided by A)

Krivine realizability: crazy new models

$$\mathcal{M}_{\perp\!\!\!\perp} \vDash A \qquad \Leftrightarrow \qquad \exists t, t \in |A|$$

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: $\mathcal{M} \vDash \neg AC$ and $\mathcal{M} \vDash \neg CH$

Algebraic structure of realizability models

There is always a lattice somewhere.

Algebraic structure of realizability models

Subtyping:

$$\frac{\Gamma \vdash p : T \quad T <: U}{\Gamma \vdash p : U}\ (\textsc{Sub}) \qquad\qquad \frac{U_1 <: T_1 \quad T_2 <: U_2}{T_1 \to T_2 <: U_1 \to U_2}\ (\textsc{S-Arr})$$

Semantically:

$$A <: B \quad \Rightarrow \quad |A| \subseteq |B|$$

↪ *this induces a structure of complete lattice with* $\curlywedge = \bigcap$

$$\|\forall x.A\| \triangleq \bigcup_{n\in\mathbb{N}} \|A\{x := n\}\| = \curlywedge\{\|A\{x := n\}\| : n \in \mathbb{N}\}$$

Réalisabilité : $\forall = \curlywedge$ $\quad \wedge = \times$ $\quad \exists = \curlyvee$ $\quad \vee = +$

Forcing : $\forall = \wedge = \curlywedge$ $\quad \exists = \vee = \curlyvee$

Algebraic structure of realizability models

Subtyping:

$$\frac{\Gamma \vdash p : T \qquad T <: U}{\Gamma \vdash p : U}\ (\textsc{Sub}) \qquad\qquad \frac{U_1 <: T_1 \qquad T_2 <: U_2}{T_1 \to T_2 <: U_1 \to U_2}\ (\textsc{S-Arr})$$

Semantically:

$$A \leq_{\perp\!\!\!\perp} B \triangleq |A| \subseteq |B|$$

↬ *this induces a structure of complete lattice with* $\curlywedge = \bigcap$

$$\|\forall x.A\| \triangleq \bigcup_{n\in\mathbb{N}} \|A\{x := n\}\| = \curlywedge \{\|A\{x := n\}\| : n \in \mathbb{N}\}$$

Réalisabilité : $\forall = \curlywedge \qquad \wedge = \times \qquad \exists = \curlyvee \qquad \vee = +$

Forcing : $\forall = \wedge = \curlywedge \qquad \exists = \vee = \curlyvee$

Algebraic structure of realizability models

Subtyping:

$$\frac{\Gamma \vdash p : T \quad T <: U}{\Gamma \vdash p : U}\ (\textsc{Sub}) \qquad\qquad \frac{U_1 <: T_1 \quad T_2 <: U_2}{T_1 \to T_2 <: U_1 \to U_2}\ (\textsc{S-Arr})$$

Semantically:

$$A \leq_{\perp\!\!\!\perp} B \triangleq |A| \subseteq |B|$$

↬ *this induces a structure of complete lattice with* $\curlywedge = \bigcap$

$$\|\forall x.A\| \triangleq \bigcup_{n \in \mathbb{N}} \|A\{x := n\}\| = \curlywedge \{\|A\{x := n\}\| : n \in \mathbb{N}\}$$

Réalisabilité : $\forall = \curlywedge$ $\quad \wedge = \times$ $\quad \exists = \curlyvee$ $\quad \vee = +$

Forcing : $\forall = \wedge = \curlywedge$ $\quad \exists = \vee = \curlyvee$

Implicative algebra

Implicative algebra

complete lattice	$(\mathcal{A}, \preccurlyeq, \curlywedge)$	+	$\cdot \to \cdot \in \mathcal{A}^{\mathcal{A}\times\mathcal{A}}$	*"implication"*
		+	$\mathcal{S} \subseteq \mathcal{A}$	*separator*

Application $\quad a@b \triangleq \curlywedge\{c \in \mathcal{A} : a \preccurlyeq b \to c\}$

Abstraction $\quad \lambda f \triangleq \curlywedge_{a\in\mathcal{A}}(a \to f(a))$

Implicative algebra

Implicative algebra

complete lattice	$(\mathcal{A}, \preccurlyeq, \curlywedge)$	$+$	$\cdot \to \cdot \in \mathcal{A}^{\mathcal{A}\times\mathcal{A}}$	*"implication"*
		$+$	$\mathcal{S} \subseteq \mathcal{A}$	*separator*

Application $\quad a@b \triangleq \curlywedge\{c \in \mathcal{A} : a \preccurlyeq b \to c\}$

Abstraction $\quad \lambda f \triangleq \curlywedge_{a\in\mathcal{A}}(a \to f(a))$

Implicative algebra

Implicative algebra

complete lattice $(\mathcal{A}, \preccurlyeq, \curlywedge)$ $+$ $\cdot \to \cdot \in \mathcal{A}^{\mathcal{A}\times\mathcal{A}}$ *"implication"*

$+$ $\mathcal{S} \subseteq \mathcal{A}$ *separator*

Types ═ Formulas

‖ ‖

Programs ═ Proofs

Order relation $\cdot \preccurlyeq \cdot$:

- $A \preccurlyeq B$ — A subtype of B
- $t \preccurlyeq A$ — t realizes A
- $t \preccurlyeq u$ — t is more defined than u

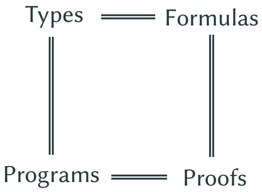
Soundness

1. If $\vdash t : A$ then $t^{\mathcal{A}} \preccurlyeq A^{\mathcal{A}}$ *(w.r.t. typing)*
2. If $t \to_\beta u$ then $t^{\mathcal{A}} \preccurlyeq u^{\mathcal{A}}$. *(w.r.t. computation)*

Implicative algebra

Implicative algebra

complete lattice $(\mathcal{A}, \preccurlyeq, \curlywedge)$	+	$\cdot \to \cdot \in \mathcal{A}^{\mathcal{A}\times\mathcal{A}}$	*"implication"*
	+	$\mathcal{S} \subseteq \mathcal{A}$	*separator*



Order relation $\cdot \preccurlyeq \cdot$:

- $A \preccurlyeq B$ — A subtype of B
- $t \preccurlyeq A$ — t realizes A
- $t \preccurlyeq u$ — t is more defined than u

Soundness

1. If $\vdash t : A$ then $t^{\mathcal{A}} \preccurlyeq A^{\mathcal{A}}$ *(w.r.t. typing)*
2. If $t \to_\beta u$ then $t^{\mathcal{A}} \preccurlyeq u^{\mathcal{A}}$. *(w.r.t. computation)*

The end

Thank you for your attention!