

**Proofs in classical logic as programs:  
a generalization of lambda calculus**

---

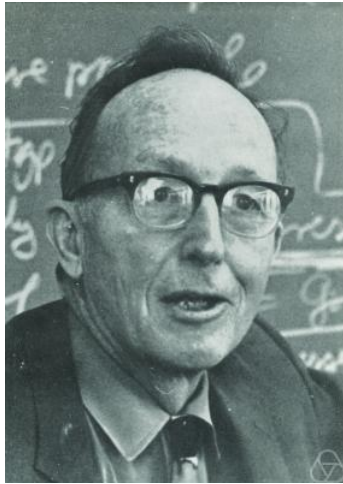
A. Salibra

Università Ca' Foscari Venezia

## Curry–Howard correspondence, in general

---

Direct relationship between systems of logic and computational formalisms



W. A. Howard



H. B. Curry

It's the join of **formal logic** and **computer science**.

## Curry–Howard for constructive logics

---

A *constructive logic* has two main properties:

- the **disjunction property** -  $\Gamma \vdash A \vee B$  iff either  $\Gamma \vdash A$  or  $\Gamma \vdash B$
- the **witness property** -  $\Gamma \vdash \exists x.A(x)$  iff  $\Gamma \vdash A(t)$  for some witness  $t$

**Problem:** in this talk logic means minimal logic (only the connective  $\rightarrow$ ). Then, what does it mean constructive logic?

It is well known that

- Intuitionistic logic (IL) is constructive.
- Classical logic (CL) is not constructive (ex:  $A \vee \neg A$  and  $\exists x(Ax \rightarrow \forall yAy)$ ).

## Curry–Howard for constructive logics

---

*Minimal Logic:*

K:  $A \rightarrow (B \rightarrow A)$  (We will write  $A \rightarrow B \rightarrow A$ )

S:  $(C \rightarrow B \rightarrow A) \rightarrow (C \rightarrow B) \rightarrow C \rightarrow A$

*Minimal Logic with False  $\perp$  ( $\neg A$  is defined as  $A \rightarrow \perp$ ):*

$\perp$ :  $\perp \rightarrow A$

*Minimal Classical Logic = Minimal Logic + Peirce law:*

P:  $((A \rightarrow B) \rightarrow A) \rightarrow A$

*Minimal Classical Logic with False  $\perp$  = Minimal Logic + P +  $\perp$ .*

Why do we need  $\perp$ -axiom? The formula  $\neg\neg A \rightarrow A$  implies the Peirce law, but the opposite direction does not hold (without  $\perp$ -axiom).

## Programs for constructive logics

---

### Minimal Logic

- ML in Hilbert style  $\iff$  Typed Combinatory Logic
- ML in Natural Deduction (ND)  $\iff$  Typed Lambda Calculus
- ML with  $\perp$  in ND  $\iff$  Typed Lambda Calculus + Abort

### Minimal Classical Logic

- MCL in ND  $\iff$  Typed Lambda Calculus + call-cc
- MCL with  $\perp$  in ND  $\iff$  Typed Lambda Calculus + call-cc + Abort

But why?

## Functional interpretation of intuitionistic formulas

---

- The axiom  $A \rightarrow B \rightarrow A$  of minimal logic corresponds from the computational point of view to a function  $K : A \rightarrow (B \rightarrow A)$  (we can give an output of type  $A$  if we have two inputs, the first of type  $A$  and the second of type  $B$ )

$$Kxy = x; \quad \text{recall that } Kxy \equiv (Kx)y$$

so that  $K$  is the  $\lambda$ -term  $K \equiv \lambda xy.x$

- The axiom  $(C \rightarrow B \rightarrow A) \rightarrow (C \rightarrow B) \rightarrow C \rightarrow A$  corresponds to a function  $S : (C \rightarrow B \rightarrow A) \rightarrow (C \rightarrow B) \rightarrow C \rightarrow A$ . How may we give an output of type  $A$  from inputs  $x : C \rightarrow (B \rightarrow A)$ ,  $y : C \rightarrow B$  and  $z : C$ ?

$$Sxyz = (xz)(yz)$$

so that  $S$  is the  $\lambda$ -term  $S \equiv \lambda xyz.(xz)(yz)$ .

- If we modify the Peirce formula  $P$  by substituting the conclusion  $A$  with  $\neg\neg A$  we get a formula  $P' \equiv ((A \rightarrow B) \rightarrow A) \rightarrow \neg\neg A$  (i.e.,  $((A \rightarrow B) \rightarrow A) \rightarrow (A \rightarrow \perp) \rightarrow \perp$ ), which is provable in minimal logic with  $\perp$ .

We would like to get an output of type  $\perp$  starting from two inputs  $x : (A \rightarrow B) \rightarrow A$  and  $y : A \rightarrow \perp$ . Let  $z : A$ . Then  $yz : \perp$ . If we have a function  $\mathcal{A} : \perp \rightarrow B$ , called “Abort”, then  $\mathcal{A}(yz) : B$ , then  $\lambda z.\mathcal{A}(yz) : A \rightarrow B$ . In conclusion

$$P'xy = y(x(\lambda z.\mathcal{A}(yz))),$$

so that  $P' \equiv \lambda xy.y(x(\lambda z.\mathcal{A}(yz)))$ .

**Warning:** if we try with Peirce formula  $((A \rightarrow B) \rightarrow A) \rightarrow A$ , we fail.

## typed $\lambda$ -calculus - minimal logic in ND

---

$$\frac{}{\Delta, x : A \vdash x : A} [\text{ax}] \quad \frac{\Delta \vdash M : A \rightarrow B \quad \Delta \vdash N : A}{\Delta \vdash MN : B} [\rightarrow_e]$$

$$\frac{\Delta, x : A \vdash M : B}{\Delta \vdash \lambda x.M : A \rightarrow B} [\rightarrow_i] \quad \frac{\Delta \vdash M : \perp}{\Delta \vdash \mathcal{A}(M) : B} [F]$$

Some Programs:

$$K = \lambda xy.x; \quad S = \lambda xyz.(xz)(yz); \quad P' = \lambda xy.y(x(\lambda z.\mathcal{A}(yz)))$$

## Some Proofs in ND

---

We show that the  $\lambda$ -term  $\lambda xy.x$  is a proof of the axiom  $A \rightarrow (B \rightarrow A)$  of minimal logic.

$$\frac{}{x : A, y : B \vdash x : A} \text{[ax]}$$
$$\frac{}{x : A \vdash \lambda y.x : B \rightarrow A} \text{[}\rightarrow_i\text{]}$$
$$\frac{}{\vdash \lambda xy.x : A \rightarrow (B \rightarrow A)} \text{[}\rightarrow_i\text{]}$$

We show that  $\lambda$ -term  $\lambda xyz.(xz)(yz)$  is a proof of the axiom :  $(C \rightarrow (B \rightarrow A)) \rightarrow (C \rightarrow B) \rightarrow C \rightarrow A$  of minimal logic.

Let  $\Delta = x : C \rightarrow (B \rightarrow A), y : C \rightarrow B, z : C$  and  $\phi = C \rightarrow (B \rightarrow A)$ .

$$\frac{\frac{\Delta \vdash y : C \rightarrow B \quad \Delta \vdash z : C}{\Delta \vdash yz : B} [\rightarrow_e]}{\Delta \vdash yz : B} [\text{ax}]$$

$$\frac{\frac{\Delta \vdash x : \phi \quad \Delta \vdash z : C}{\Delta \vdash xz : B \rightarrow A} [\rightarrow_e]}{\Delta \vdash xz : B \rightarrow A} [\text{ax}]$$

$$\frac{x : C \rightarrow (B \rightarrow A), y : C \rightarrow B, z : C \vdash (xz)(yz) : A}{\vdash \lambda xyz.(xz)(yz) : (C \rightarrow (B \rightarrow A)) \rightarrow (C \rightarrow B) \rightarrow C \rightarrow A} [\rightarrow_i]$$

## Proof simplification

---

*Reduction:*

$$(\lambda x.M)N \rightarrow M[x := N]; \quad \mathcal{A}(M)N \rightarrow \mathcal{A}(M)$$

$$\Delta, x : A \vdash M : B$$

-----

$$\Delta \vdash \lambda x.M : A \rightarrow B$$

$$\Delta \vdash N : A$$

-----

$$\Delta \vdash (\lambda x.M)N : B$$

Both the programs  $(\lambda x.M)N$  and  $M[x := N]$  correspond to a proof of the formula  $B$  by the assumption in  $\Delta$ . However, the proof  $(\lambda x.M)N$  does unuseful work because of an arrow introduction followed by an arrow elimination. Then the program  $M[x := N]$  simplifies the proof  $(\lambda x.M)N$  of  $B$  as follows: we take the proof  $M$  of  $B$  from the assumptions in  $\Delta, x : A$  and we substitute every occurrence of the assumption  $x : A$  by the proof  $N : A$ .

## Proof simplification

---

For example,

$$y : B, x : A \rightarrow A \vdash y : B$$
$$y : B, z : A \vdash z : A$$
$$y : B \vdash \lambda x. y : (A \rightarrow A) \rightarrow B$$
$$y : B \vdash \lambda z. z : A \rightarrow A$$
$$y : B \vdash (\lambda x. y)(\lambda z. z) : B$$

becomes the trivial proof

$$y : B \vdash y : B$$

## Proof simplification

---

We have the following further rule :  $\mathcal{A}(M)N \rightarrow \mathcal{A}(M)$ . Why? Here is an example.

$\Delta \vdash M : \perp$

----- ( $\perp$ )

$\Delta \vdash \mathcal{A}(M) : A \rightarrow B$

$\Delta \vdash N : A$

----- ( $\rightarrow_e$ )

$\Delta \vdash \mathcal{A}(M)N : B$

reduces to the simple proof

$\Delta \vdash M : \perp$

----- ( $\perp$ )

$\Delta \vdash \mathcal{A}(M) : B$

## Peirce law: the backtrack interpretation

---

Recall that until 1990 it was common belief that classical logic didn't admit a Curry–Howard correspondence. We cannot give a functional interpretation to the Peirce law. Why?

A tentative proof of the Peirce law in ND (to be read bottom -up): we get an output of type  $A$  from an input of type  $(A \rightarrow B) \rightarrow A$  only if we have a proof of  $A \vdash B$  (i.e., from an input of type  $A$  we can construct an output of type  $B$ ). But this is not possible in intuitionistic logic because  $B$  is arbitrary.

$$\begin{array}{c}
 A \vdash B \\
 \text{-----}(\rightarrow_i) \\
 (A \rightarrow B) \rightarrow A \vdash (A \rightarrow B) \rightarrow A \quad \vdash A \rightarrow B \\
 \text{-----}(\rightarrow_e) \\
 (A \rightarrow B) \rightarrow A \vdash A \\
 \text{-----}(\rightarrow_i) \\
 \vdash ((A \rightarrow B) \rightarrow A) \rightarrow A
 \end{array}$$

To prove  $A \vdash B$  we need that  $B \equiv B_1 \rightarrow B_2 \rightarrow \dots \rightarrow B_n \rightarrow C$  and that we can construct functionally an output of type  $C$  from  $n + 1$  inputs of type  $A, B_1, \dots, B_n$  respectively.

## Peirce law: the backtrack interpretation

---

Let  $P$  be the Peirce formula.

Remark: We have that  $C \vdash A$  iff  $C \vdash C \rightarrow A$  in intuitionistic logic. We prove this by using the functional interpretation. Let  $x, y : C$ . If  $h : C \rightarrow (C \rightarrow A)$  then  $hxy : A$ . It follows that  $hxx : A$  and  $\lambda x.hxx : C \rightarrow A$ . Vice versa, if  $g : C \rightarrow A$  then  $gx : A$ , so that  $\lambda xy.gx : C \rightarrow (C \rightarrow A)$ . Then we can change the deduction rule  $\rightarrow_i$

$$A \vdash B$$
$$\text{---}(\rightarrow_i)$$
$$\vdash A \rightarrow B$$

as follows (to be read bottom-up):

$$A \vdash A \rightarrow B$$
$$\text{---}(\rightarrow_i^*)$$
$$\vdash A \rightarrow B$$



## Peirce law: the backtrack interpretation

---

The complete proof:

$$\begin{array}{c}
 A \vdash A \rightarrow B, P \\
 \text{-----}(\rightarrow_i^*) \\
 \vdash (A \rightarrow B) \rightarrow P \qquad \vdash A \rightarrow B, P \\
 \text{-----}(\rightarrow_e) \\
 \vdash P
 \end{array}$$

The proof is complete because  $A \vdash A \rightarrow B, P$  is provable:  $A$  is in the conclusion of  $P$ . Note that the left occurrence (in red) of  $A$  in  $A \vdash A \rightarrow B, P$  comes from  $A \rightarrow B$  by  $(\rightarrow_i^*)$ , while the other occurrence of  $A$  (making the proof possible) is in blue within  $P$ :  $A \vdash A \rightarrow B, ((A \rightarrow B) \rightarrow A) \rightarrow A$ .

## Griffin's call/cc and Parigot's $\lambda\mu$

---



M. Parigot



T. G. Griffin

Leggere i seguenti due articoli per capire come estendere il lambda calcolo tipato per ottenere le prove della logica classica:

Griffin, 1990 "A formulae-as-types notion of control", POPL'90. [Single-conclusion Classical Natural Deduction with Elimination of Double Negation corresponding to explicit access to the current continuation \(call/cc\).](#)

Parigot, 1992 " $\lambda\mu$ -calculus: an algorithmic interpretation of Classical Natural Deduction", LPAR'92. [Multi-conclusion Natural Deduction. The Call/cc operator can be encoded in an extension of  \$\lambda\$ -calculus.](#)