Curry-Howard Correspondence for Classical Logic

CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE

Stéphane Graham-Lengrand

CNRS, Laboratoire d'Informatique de l'X

Stephane.Lengrand@Polytechnique.edu

Practicalities

E-mail address: `Stephane.Lengrand@Polytechnique.edu`

Website: `http://www.lix.polytechnique.fr/~lengrand/`

Slides . . .

Schedule: Tuesdays, from 16:15 to 19:15

7th December, 14th December, 3rd January, 10th January (guest lecture by Beniamino Accattoli)

Room : 2035, Sophie Germain building

Lecture I
Classical logic as a typing system

Contents

I. Introduction

Curry-Howard correspondence for Classical Logic

These lectures are part of the course

Logique linéaire et paradigmes logiques du calcul

(mostly)

Although, *polarity* and *focusing* -from linear logic- have played a major part in the understanding of C-H correspondence for Classical Logic.

(see e.g. Olivier Laurent's PhD work Laurent [2003])

Curry-Howard correspondence

One of two sides of the coin

"**computational interpretation** of a logic"

output of a computation = cut-free proof

Side 1 computation as proof search

(starting from a formula to prove)

Logic programming (see e.g. Dale Miller's course)

Side 2 computation as *composition* of proofs / cut-elimination

(starting from a proof with cuts)

Curry-Howard (see e.g. this course)

II. What works and what does not

Where it all works smoothly

Intuitionistic/minimal logic

Logic	*Programming language* λ-calculus	*Categories*
Propositions	Types	Objects
Proofs	Typed programs λ-terms	Morphisms
Cut/Composition	Program composition β-redex	Morphism composition
Where the stuff happens (Cut-elimination)	Execution β-reduction	Equality of morphisms (commuting diagrams)

Original correspondence

For minimal logic:

Curry	Combinators (S,K,I)	$\leftrightarrow$	Hilbert-style system
Howard Howard [1980]	Typed λ-terms	$\leftrightarrow$	Natural Deduction

I : $A \rightarrow A$

K : $A \rightarrow B \rightarrow A$ (provides erasure)

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

$$\mathbf{I}\ M \longrightarrow M$$

$$\mathbf{K}\ M\ N \longrightarrow M$$

$$\mathbf{S}\ M\ N\ P \longrightarrow M\ P\ (N\ P)$$

Original correspondence

For minimal logic:

Curry	Combinators (S,K,I)	$\leftrightarrow$	Hilbert-style system
Howard Howard [1980]	Typed λ-terms	$\leftrightarrow$	Natural Deduction

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

$$\frac{\Gamma, x{:}A \vdash M : B}{\Gamma \vdash \lambda x.M : A \to B} \qquad \frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B}$$

$$(\lambda x.M)\ N \quad \longrightarrow_\beta \quad \{^N\!/_x\}M$$

$$\lambda x.M\ x \quad \longrightarrow_\eta \quad M \qquad \text{if } x \notin \mathsf{FV}(M)$$

$=_{\beta\eta}$ sound and complete for Cartesian Closed Categories (CCC)

Generalising the approach

Decorate proofs with syntactic terms: $\Gamma \vdash A$ becomes $\Gamma \vdash M : A$

Proof transformations

Reductions (execution of) M: $M \longrightarrow_{\mathcal{S}} N$

given by rewrite system $\mathcal{S}$.

Desired properties of the reduction system

- Progress, i.e. any term containing undesirable structures can be reduced.
- Subject reduction property, i.e. preservation of typing:

  If $\Gamma \vdash M : A$ and $M \longrightarrow_{\mathcal{S}} N$ then $\Gamma \vdash N : A$

1. Confluence, programs are deterministic
2. Normalisation (strong), i.e. execution of programs terminates.

Example

From minimal logic to *intuitionistic logic* = *minimal logic* + *"Ex falso Quodlibet"*

add rule:

$$\frac{\Gamma \vdash M : \bot}{\Gamma \vdash \text{abort}(M) : A}$$

Computational behaviour:

$$\text{abort}(M)\ N \longrightarrow\ \text{abort}(M)$$

In category theory: add to a CCC an *initial* object $\bot$

(i.e. for every object A there is a unique morphism $\bot \longrightarrow A$)

Now, remember that $\neg A$ is $A \rightarrow \bot$

How to get classical logic

Either add axiom schemes

$$(\neg\neg A) \Rightarrow A \qquad \text{(Elimination of double negation)}$$

$$((A \Rightarrow B) \Rightarrow A) \Rightarrow A \qquad \text{(Peirce's law)}$$

$$A \vee \neg A \qquad \text{(Law of excluded middle)}$$

In presence of "Ex falso Quodlibet" ($\bot \Rightarrow A$): *All equivalent*

Without "Ex falso Quodlibet": *only EDN⇒PL⇒LEM*

...Or with inference rules, for instance:

$$\frac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash A} \text{ for EDN}$$

or by the structure of formalism cf. classical sequent calculus and right-contraction

The bad news

Take a CCC with initial object $\bot$

Require that for all object A, A is naturally isomorphic to $(A \Rightarrow \bot) \Rightarrow \bot$

The category collapses to a boolean algebra:

at most 1 morphism between any 2 objects

= cannot distinguish 2 proofs of the same theorem

Quite useless for a theory of proofs, or for the proofs-as-program paradigm

Has classical logic a computational content?

Girard, Lafont, Taylor *Proofs and types*: No Girard et al. [1989]

III. A bit of history

The notion of continuation

Program execution flow:

$\downarrow$	code P that has been executed, producing data v
v	its output
$\downarrow$	code E that remains to be executed, consuming data v

Continuation

= programming environment/context within which some code is executed

The notion of continuation

. . . is also useful for compiling recursive calls

```
myfunction(a1,...,an){
 some code;
 x = myfunction(a1',...,an');
 some code possibly using x;
}
```

When executing recursive call, whole environment must be saved to resume computation (code that remains to be executed + state of memory).

Not needed if `some code possibly using x` is empty (tail recursion).

Trick = pass it to the recursive call as a “continuation” `c'` :

```
myfunction(a1,...,an,c){
 some code;
 return myfunction(a1',...,an',c');
}
```

The notion of continuation in λ-calculus

Program execution flow:

$\downarrow$ code P that has been executed, producing data v

v its output

$\downarrow$ code E that remains to be executed, consuming data v

can be seen in

- P is a λ-term that is reduced
- E is the context, in the syntactic sense (a term with a hole $E[\ ]$)

$E[\ ]$ can also be seen as a function $\lambda x.E[x]$

The notion of control

In pure λ-calculus, P has no knowledge of $E[\,]$ while being evaluated.

Control =

letting a program know and manipulate its environment/continuation

getting "unpure features", modelling `goto` instructions

- Reynolds Reynolds [1972], Strachey-Wadsworth Strachey and Wadsworth [2000] (re-edition of 74)

  on continuations and *call-with-current-continuation (call-cc): cc*

  Added to programming langage `Scheme`

- Felleisen's PhD work Felleisen [1987] on Syntactic Theory of Control: the $\mathcal{C}$ *operator*

Connection with Logic (89-90)

The general idea:

$$
\begin{array}{lcl}
E[\text{abort}(M)] & \longrightarrow & M \\
E[\text{cc } M] & \longrightarrow & E[M\ (\lambda x.E[x])] \\
E[\mathcal{C}\ M] & \longrightarrow & M\ (\lambda x.E[x])
\end{array}
$$

In presence of abort( ), cc and $\mathcal{C}$ are interdefinable:

$$
\begin{array}{lcll}
\mathcal{C}\ M & := & \text{cc}\ (\lambda k.\text{abort}(M\ k)) & k \notin \text{FV}(M) \\
\text{cc}\ M & := & \mathcal{C}\ (\lambda k.k\ (M\ k)) & k \notin \text{FV}(M)
\end{array}
$$

Griffin Griffin [1990]:

cc can be typed by $((A \to B) \to A) \to A$

$\mathcal{C}$ can be typed by $(\neg\neg A) \to A$

Central question about control

$$E[\textsf{abort}(M)] \longrightarrow M$$
$$E[\textsf{cc}\ M] \longrightarrow E[M\ (\lambda x.E[x])]$$
$$E[\mathcal{C}\ M] \longrightarrow M\ (\lambda x.E[x])$$

Above rules are not “standard” rewrite rules. . .

What exactly does $E[\,]$ stand for / range over?

More fundamentally:

What kind of continuation can be captured by a control operator?

Is the capture delimited? undelimited? etc

One proposed formalisation: Parigot's $\lambda\mu$-calculus Parigot [1992]

$$\begin{array}{lrcl} \text{Terms} & M, N, P \ldots & ::= & x \mid \lambda x.M \mid M\ N \mid \mu\alpha.c \\ \text{Commands} & c & ::= & [\alpha]M \end{array}$$

$$\frac{}{\Gamma, x : A \vdash x : A; \Delta}$$

$$\frac{\Gamma, x : A \vdash M : B; \Delta}{\Gamma \vdash \lambda x.M : A \to B; \Delta} \qquad \frac{\Gamma \vdash M : A \to B; \Delta \quad \Gamma \vdash N : A; \Delta}{\Gamma \vdash M\ N : B; \Delta}$$

$$\frac{c : (\Gamma \vdash\ ; \alpha : A, \Delta)}{\Gamma \vdash \mu\alpha.c : A; \Delta} \qquad \frac{\Gamma \vdash M : A; \alpha : A, \Delta}{[\alpha]M : (\Gamma \vdash\ ; \alpha : A, \Delta)}$$

Parigot's $\lambda\mu$-calculus Parigot [1992]

Extra reduction rules:

$$
\begin{array}{lcl}
(\mu\alpha.c)\ N & \longrightarrow & \mu\beta.\{{}^{[\beta]M\ N}/_{[\alpha]M}\}\,c \\
[\beta]\mu\alpha.c & \longrightarrow & \{{}^{\beta}/_{\alpha}\}\,c
\end{array}
$$

Integrates Peirce's law: cc $:= \ \lambda x.\mu\alpha.[\alpha](x\ \lambda y.\mu\beta.[\alpha]y)$ $\quad : ((A\to B)\to A)\to A$

Consider that contexts are of the form $E[\,] = [\gamma]([\,]\ N_1 \ldots N_n)$

If given a top-level continuation variable top $:\bot$ (Ariola-Herbelin Ariola and Herbelin [2003]),

- integrates "Ex falso quod libet" $\quad \lambda x.\mu\alpha.[\text{top}]x \quad : \bot\to A$
- integrates DNE $\quad \mathcal{C} := \ \lambda x.\mu\alpha.[\text{top}](x\ \lambda y.\mu\beta.[\alpha]y) \quad : (\neg\neg A)\to A$

So far, so good

Symmetry

There's something symmetric about classical logic:

- Boolean algebras
- De Morgan duality:

$$\neg(A \wedge B) = \neg A \vee \neg B$$

$$\neg(A \vee B) = \neg A \wedge \neg B$$

- Classical Sequent calculus LK

  left-contraction symmetric to right-contraction (≠ intuitionistic logic)

So far, not explicit in our proof theory + term calculi

Filinski Filinski [1989]: first formalisation of a duality between

- functions as values
- functions as continuations

Symmetric λ-calculus, with explicit conversions from one view to the other

No explicit connection with logic. Is there one?

Barbanera - Berardi and after

Yes, there's one.

See BB's symmetric λ-calculus Barbanera and Berardi [1996]: Natural Deduction with continuations

It is more like a one-sided sequent calculus

Symmetry of the calculus corresponds to symmetry/duality of LK

Other calculi for (bi-sided) versions of LK, with cut-elimination as computation:

- Urban's calculus Urban [2000],
- Curien-Herbelin's $\overline{\lambda}\mu\widetilde{\mu}$ Curien and Herbelin [2000] for $\Rightarrow$ (easier in bi-sided sequent calculus),

  later extended by Wadler Wadler [2003] for $\wedge$ and $\vee$ (connecting to De Morgan)

Two independent works

Curien-Herbelin's aim:

Express duality of computation syntactically (with a Filinski-like calculus)

Semantics, no proof of SN.

Urban's aim:

Have a typing system as close as possible to LK, have a reduction system as close as possible to basic cut-elimination procedures

SN, but no semantics.

Then: a broad literature on comparing such calculi.

Curien-Herbelin-Wadler - syntax

terms	t	$::= x \mid \mu\beta.c \mid \lambda x.t \mid \langle t_1, t_2 \rangle \mid \mathsf{inj}_i(t)$
continuations	e	$::= \alpha \mid \mu x.c \mid t :: e \mid \langle e_1, e_2 \rangle \mid \mathsf{inj}_i(e)$
commands	c	$::= \langle t \bullet e \rangle$

Intuition:

$x, y, \ldots$:	inputs (variables standing for terms)
$\alpha, \beta, \ldots$:	outputs (variables standing for continuations)
terms =	some inputs (free term variables)
	+ one main output
	+ alternative outputs (free continuation variables)
continuations =	one main input
	+ additional inputs (free term variables)
	+ some outputs (free continuation variables)
commands =	a term facing a continuation (this interaction creates computation)

Curien-Herbelin-Wadler - typing

$$\frac{}{\Gamma, x : A \vdash x : A \;;\; \Delta} \qquad \frac{}{\Gamma \;;\; \alpha : A \vdash \alpha : A, \Delta}$$

$$\frac{\Gamma, x : A \vdash t : B \;;\; \Delta}{\Gamma \vdash \lambda x.t : A \to B \;;\; \Delta} \qquad \frac{\Gamma \vdash t : A \;;\; \Delta \quad \Gamma \;;\; e : B \vdash \Delta}{\Gamma \;;\; t :: e : A \to B \vdash \Delta}$$

$$\frac{\Gamma \vdash t_1 : A_1 \;;\; \Delta \quad \Gamma \vdash t_2 : A_2 \;;\; \Delta}{\Gamma \vdash \langle t_1, t_2 \rangle : A_1 \wedge A_2 \;;\; \Delta} \qquad \frac{\Gamma \;;\; e : A_i \vdash \Delta}{\Gamma \;;\; \mathsf{inj}_i(e) : A_1 \wedge A_2 \vdash \Delta}$$

$$\frac{\Gamma \vdash t : A_i \;;\; \Delta}{\Gamma \vdash \mathsf{inj}_i(t) : A_1 \vee A_2 \;;\; \Delta} \qquad \frac{\Gamma \;;\; e_1 : A_1 \vdash \Delta \quad \Gamma \;;\; e_2 : A_2 \vdash \Delta}{\Gamma \;;\; \langle e_1, e_2 \rangle : A_1 \vee A_2 \vdash \Delta}$$

$$\frac{c : (\Gamma \vdash \alpha : A, \Delta)}{\Gamma \vdash \mu\alpha.c : A \;;\; \Delta} \qquad \frac{c : (\Gamma, x : A \vdash \Delta)}{\Gamma \;;\; \mu x.c : A \vdash \Delta}$$

$$\frac{\Gamma \vdash t : A \;;\; \Delta \quad \Gamma \;;\; e : A \vdash \Delta}{\langle t \bullet e \rangle : (\Gamma \vdash \Delta)}$$

Example: Law of Excluded Middle

A story: The devil, the fool, and the $1000000. (borrowed from Phil Wadler)

- I have an offer for you! My promise is:

Either I offer you $1000000

or, if you give me $1000000

then I will grant you any wish

I choose to offer you the latter.

- Here's $1000000! I want immortality.

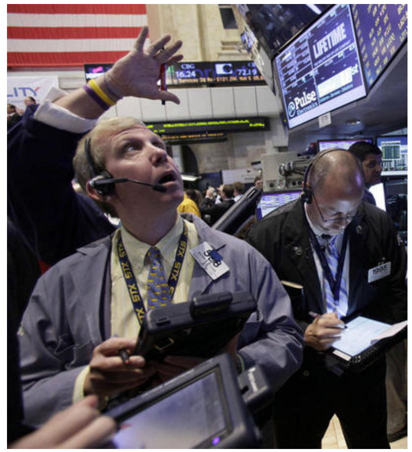
- Well done and thank you!

Now, I've changed my mind.

I've now decided to fulfill my promise

by offering you $1000000.

Here is your money back!



Questions?

References

Z. M. Ariola and H. Herbelin. Minimal classical logic and control operators. In J. C. M. Baeten, J. K. Lenstra, J. Parrow, and G. J. Woeginger, editors, *Proc. of the 30th Intern. Col. on Automata, Languages and Programming (ICALP)*, volume 2719 of *LNCS*, pages 871–885. Springer-Verlag, July 2003.

F. Barbanera and S. Berardi. A symmetric lambda-calculus for classical program extraction. *Inform. and Comput.*, 125(2):103–117, 1996.

P.-L. Curien and H. Herbelin. The duality of computation. In *Proc. of the* 5^{th} *ACM SIGPLAN Int. Conf. on Functional Programming (ICFP'00)*, pages 233–243. ACM Press, 2000.

M. Felleisen. *The Calculi of* λ*-v-CS Conversion: A Syntactic Theory of Control and State in Imperative Higher-Order Programming Languages*. PhD thesis, Department of Computer Science, Indiana University, Bloomington, Indiana, Aug. 1987.

A. Filinski. Declarative continuations and categorical duality. Master's thesis, DIKU, Computer Science Department, University of Copenhagen, Aug. 1989. DIKU Rapport 89/11.

J.-Y. Girard, P. Taylor, and Y. Lafont. *Proofs and Types*, volume 7 of *Cambridge Tracts in Theoret. Comput. Sci.* Cambridge University Press, 1989.

T. G. Griffin. A formulae-as-type notion of control. In P. Hudak, editor, *Proc. of the 17th Annual ACM Symp. on Principles of Programming Languages (POPL'90)*, pages 47–58. ACM Press, Jan. 1990. doi: 10.1145/96709.96714.

W. A. Howard. The formulae-as-types notion of construction. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism*, pages 479–490. Academic Press, London, 1980. Reprint of a manuscript written 1969.

O. Laurent. Polarized proof-nets and lambda-mu calculus. *Theoret. Comput. Sci.*, 1(290): 161–188, 2003.

M. Parigot. $\lambda\mu$-calculus: An algorithmic interpretation of classical natural deduction. In A. Voronkov, editor, *Proc. of the Int. Conf. on Logic Programming and Automated Reasoning (LPAR'92)*, volume 624 of *LNCS*, pages 190–201. Springer-Verlag, July 1992.

J. C. Reynolds. Definitional interpreters for higher-order programming languages. In *Proc. of the ACM annual Conf.* , pages 717–740, 1972.

C. Strachey and C. P. Wadsworth. Continuations: A mathematical semantics for handling fulljumps. *Higher-Order and Symbolic Computation*, 13:135–152, April 2000. ISSN 1388-3690. doi: 10.1023/A:1010026413531. URL http://dl.acm.org/citation.cfm?id=609150.609224.

C. Urban. *Classical Logic and Computation*. PhD thesis, University of Cambridge, 2000.

P. Wadler. Call-by-value is dual to call-by-name. In *Proc. of the 8th ACM SIGPLAN Int. Conf. on Functional programming (ICFP'03)*, volume 38, pages 189–201. ACM Press, Sept. 2003. ISBN 1581137567.