C'est maintenant qu'on calcule

It's time to compute

Au cœur de la dualité

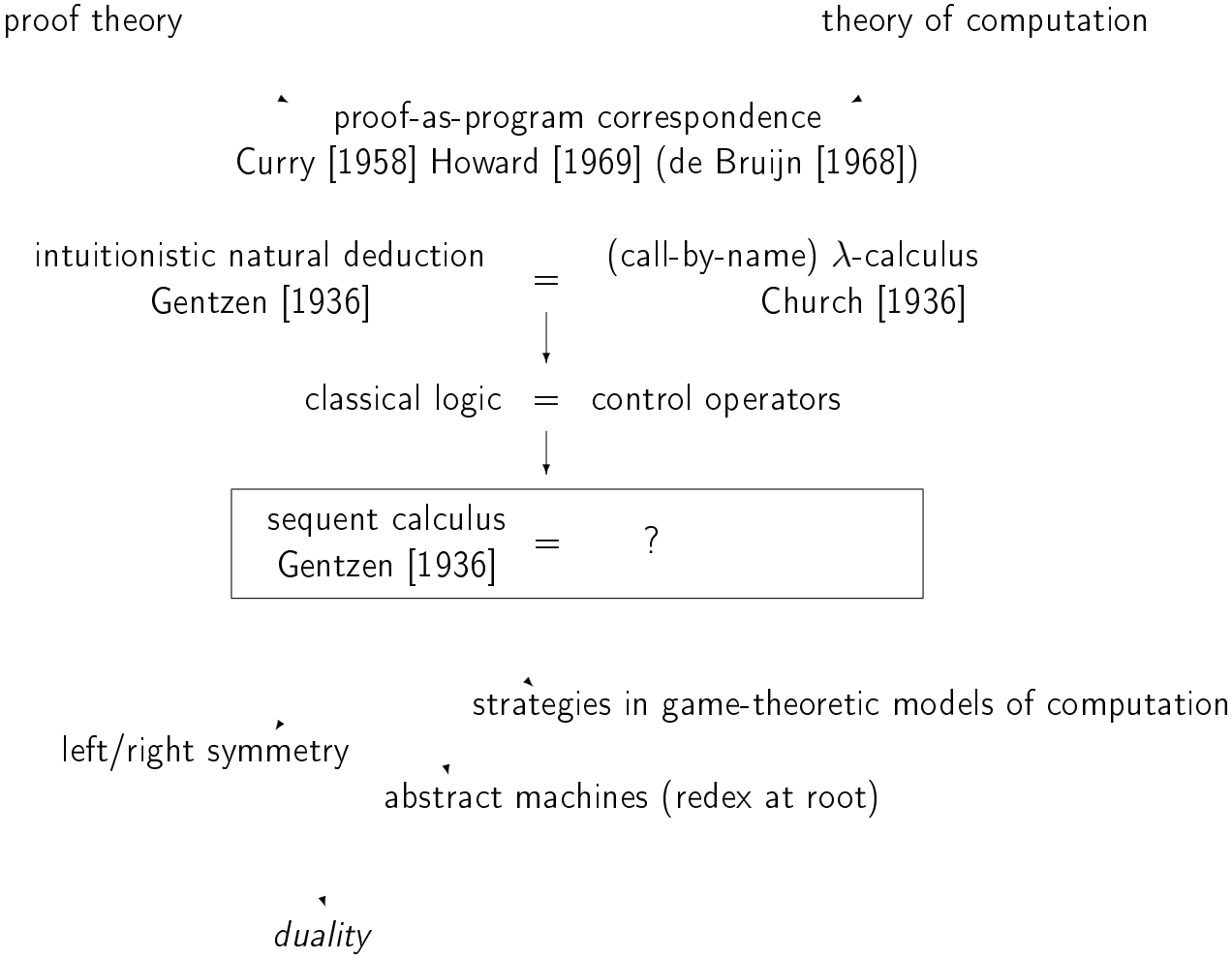
Deep in the duality

Préambule

Je remercie avant tout Jean-Pierre Jouannaud. Sans sa gentille insistance, nous ne serions peut-être pas réuni ensemble ici présent.

J'ai profité de l'occasion de l'habilitation pour approfondir ce qui est sans doute ma contribution scientifique principale, à savoir mon travail avec Pierre-Louis Curien sur la dualité du calcul.

General research framework



The syntactic duality of computation

(The $\mu\tilde{\mu}$-subsystem and its extensions)

The $\mu\tilde{\mu}$-subsystem: the heart of computation

Computational properties

- syntactic emphasis of the call-by-name vs call-by-value duality,
- syntactic emphasis of a duality between term and evaluation contexts that interact to produce results,
- accepts extension made by specific sets of interacting constructors,
- condenses all the computational aspects of its extensions

Syntax

Commands	c	$::=$	$\langle v \| e \rangle$
Terms	v	$::=$	$\mu\alpha.c \mid x$
Evaluation contexts	e	$::=$	$\tilde{\mu}x.c \mid \alpha$

Semantics

$$(\mu)\quad \langle \mu\alpha.c \| e \rangle \rightarrow c[\alpha \leftarrow e]$$
$$(\tilde{\mu})\quad \langle v \| \tilde{\mu}x.c \rangle \rightarrow c[x \leftarrow v]$$

- a good tool to better understand the theory of call-by-value λ-calculus
- a uniform analysis of η-conversion,
- a close connection with abstract environment machines (redexes are at the head of the expressions),

Proof-theoretical properties

- full Curry-Howard correspondence for sequent calculus (left introduction rules build evaluation contexts),
- the "dark side" of sequent calculus is the call-by-value side,
- context-implicit tree-like representation of sequent calculus.

The $\mu\tilde{\mu}$-subsystem

Syntax of $\overline{\lambda}\mu\tilde{\mu}$

$$\begin{array}{lrcl} \text{Commands} & c & ::= & \langle v \| e \rangle \\ \text{Terms} & v & ::= & \mu\alpha.c \mid x \mid \lambda x.v \\ \text{Evaluation contexts} & e & ::= & \tilde{\mu}x.c \mid \alpha \mid v \cdot e \end{array}$$

Semantics

$$\begin{array}{lrcl} (\mu) & \langle \mu\alpha.c \| e \rangle & \to & c[\alpha \leftarrow e] \\ (\tilde{\mu}) & \langle v \| \tilde{\mu}x.c \rangle & \to & c[x \leftarrow v] \\ (\to) & \langle \lambda x.v \| v' \cdot e \rangle & \to & \langle v' \| \tilde{\mu}x.\langle v \| e \rangle \rangle \end{array}$$

Equivalent syntax in $\lambda\mu$-calculus

$$\begin{array}{lrcl} \text{Commands} & c & ::= & e[v] \\ \text{Terms} & v & ::= & \mu\alpha.c \mid x \mid \lambda x.v \\ \text{Evaluation contexts} & e[\,] & ::= & \mathbf{let}\ x = [\,]\ \mathbf{in}\ c \mid [\alpha]([\,]) \mid e[[\,]\, v] \end{array}$$

Rules in the syntax of $\lambda\mu$-calculus

$$\begin{array}{lrcl} (\mu) & e[\mu\alpha.c] & \to & c[[\alpha]v \leftarrow e[v]] \\ (\tilde{\mu}) & \mathbf{let}\ x = v\ \mathbf{in}\ c & \to & c[x \leftarrow v] \\ (\to) & e[(\lambda x.v)v'] & \to & \mathbf{let}\ x = v'\ \mathbf{in}\ e[v] \end{array}$$

Adding constructions: the example of abstraction and application

Syntax of $\overline{\lambda}\mu\tilde{\mu}$

$$
\begin{array}{lrcl}
\text{Commands} & c & ::= & \langle v \| e \rangle \\
\text{Terms} & v & ::= & \mu\alpha.c \mid x \mid \lambda x.v \\
\text{Evaluation contexts} & e & ::= & \tilde{\mu}x.c \mid \alpha \mid v \cdot e
\end{array}
$$

Semantics

$$
\begin{array}{lrcl}
(\mu) & \langle \mu\alpha.c \| e \rangle & \rightarrow & c[\alpha \leftarrow e] \\
(\tilde{\mu}) & \langle v \| \tilde{\mu}x.c \rangle & \rightarrow & c[x \leftarrow v] \\
(\rightarrow) & \langle \lambda x.v \| v' \cdot e \rangle & \rightarrow & \langle v' \| \tilde{\mu}x.\langle v \| e \rangle \rangle
\end{array}
$$

Equivalent syntax in $\lambda\mu$-calculus

$$
\begin{array}{lrcl}
\text{Commands} & c & ::= & e[v] \\
\text{Terms} & v & ::= & \mu\alpha.c \mid x \mid \lambda x.v \\
\text{Evaluation contexts} & e[\,] & ::= & \mathbf{let}\ x = [\,]\ \mathbf{in}\ c \mid [\alpha]([\,]) \mid e[[\,]\, v]
\end{array}
$$

Rules in the syntax of $\lambda\mu$-calculus

$$
\begin{array}{lrcl}
(\mu) & e[\mu\alpha.c] & \rightarrow & c[[\alpha]v \leftarrow e[v]] \\
(\tilde{\mu}) & \mathbf{let}\ x = v\ \mathbf{in}\ c & \rightarrow & c[x \leftarrow v] \\
(\rightarrow) & e[(\lambda x.v)v'] & \rightarrow & \mathbf{let}\ x = v'\ \mathbf{in}\ e[v]
\end{array}
$$

The computational-classical-logic lineage of $\mu\tilde{\mu}^{\rightarrow}$

The programming side

Landin's J [1964]
an operator to translate goto and labels

Reynolds' `escape` [1972]
a syntactical variant of `call-cc`

Scheme's `catch/throw` [1975]
a static variant of Lisp's `catch/throw`

Felleisen *et al*'s C [1986]
abstract study of λ-calculus with control

The proof theory side

Prawitz [1965]
normalisation of natural deduction + $\neg\neg A \to A$
(no Curry-Howard)

↘ ↙

The Curry-Howard connection

Griffin [1990]
typing C of type $\neg\neg A \to A$

Parigot [1992]
a "clean" variant to $\lambda_{\mathcal{C}}$-calculus: $\lambda\mu$-calculus

The $\mu\tilde{\mu}$-subsystem

(the critical dilemma of computation)

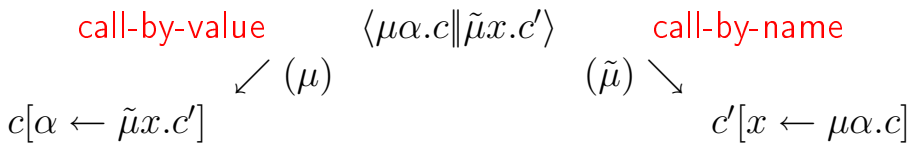
Syntax

Commands	c	$::=$	$\langle v \| e \rangle$
Terms	v	$::=$	$\mu\alpha.c \mid x \mid \ldots$
Evaluation contexts	e	$::=$	$\tilde{\mu}x.c \mid \alpha \mid \ldots$
Linear ev. contexts	E	$::=$	$\alpha \mid \ldots$

Semantics

$$(\mu)\quad \langle \mu\alpha.c \| e \rangle \;\to\; c[\alpha \leftarrow e]$$
$$(\tilde{\mu})\quad \langle v \| \tilde{\mu}x.c \rangle \;\to\; c[x \leftarrow v]$$

The critical pair



The $\mu\tilde{\mu}$-subsystem

(the critical dilemma of computation)

Syntax

Commands	c	$::=$	$\langle v \| e \rangle$
Terms	v	$::=$	$\mu\alpha.c \mid x \mid \ldots$
Evaluation contexts	e	$::=$	$\tilde{\mu}x.c \mid \alpha \mid \ldots$
Linear ev. contexts	E	$::=$	$\alpha \mid \ldots$

Semantics

$$\begin{array}{lrcl} (\mu) & \langle \mu\alpha.c \| e \rangle & \to & c[\alpha \leftarrow e] \\ (\tilde{\mu}) & \langle v \| \tilde{\mu}x.c \rangle & \to & c[x \leftarrow v] \end{array}$$

The critical pair in the syntax of $\lambda\mu$-calculus

$$\begin{array}{ccc} & \mathbf{let}\ x = \mu\alpha.c\ \mathbf{in}\ c' & \\ \text{call-by-value} \swarrow (\mu) & & (\tilde{\mu}) \searrow \text{call-by-name} \\ c[[\alpha]v \leftarrow \mathbf{let}\ x = v\ \mathbf{in}\ c'] & & c'[x \leftarrow \mu\alpha.c] \end{array}$$

The $\mu\tilde{\mu}$-subsystem

(the critical dilemma of computation)

Syntax

Commands	c	$::=$	$\langle v \| e \rangle$
Terms	v	$::=$	$\mu\alpha.c \mid x \mid \ldots$
Evaluation contexts	e	$::=$	$\tilde{\mu}x.c \mid \alpha \mid \ldots$
Linear ev. contexts	E	$::=$	$\alpha \mid \ldots$

Semantics

$$\begin{array}{lrcl} (\mu) & \langle \mu\alpha.c \| e \rangle & \to & c[\alpha \leftarrow e] \\ (\tilde{\mu}) & \langle v \| \tilde{\mu}x.c \rangle & \to & c[x \leftarrow v] \end{array}$$

The critical pair

$$\begin{array}{ccc} & \langle \mu\alpha.c \| \tilde{\mu}x.c' \rangle & \\ \text{call-by-value} \quad \swarrow (\mu) & & (\tilde{\mu}) \searrow \quad \text{call-by-name} \\ c[\alpha \leftarrow \tilde{\mu}x.c'] & & c'[x \leftarrow \mu\alpha.c] \end{array}$$

The $\mu\tilde{\mu}$-subsystem

(the call-by-name confluent restriction)

Syntax

Commands	c	$::=$	$\langle v \| e \rangle$
Terms	v	$::=$	$\mu\alpha.c \mid x \mid \ldots$
Evaluation contexts	e	$::=$	$\tilde{\mu}x.c \mid E$
Linear ev. contexts	E	$::=$	$\alpha \mid \ldots$

Semantics

$$\begin{array}{lrcl} (\mu_n) & \langle \mu\alpha.c \| E \rangle & \to & c[\alpha \leftarrow E] \\ (\tilde{\mu}) & \langle v \| \tilde{\mu}x.c \rangle & \to & c[x \leftarrow v] \end{array}$$

The solved critical pair

$$\begin{array}{ccc} & \langle \mu\alpha.c \| \tilde{\mu}x.c' \rangle & \\ \text{call-by-value} \quad \swarrow (\mu) & & (\tilde{\mu}) \searrow \quad \text{call-by-name} \\ c[\alpha \leftarrow \tilde{\mu}x.c'] & & c'[x \leftarrow \mu\alpha.c] \end{array}$$

The $\mu\tilde{\mu}$-subsystem

(the call-by-value confluent restriction)

Syntax

Commands	c	$::=$	$\langle v \| e \rangle$
Terms	v	$::=$	$\mu\alpha.c \mid V$
Evaluation contexts	e	$::=$	$\tilde{\mu}x.c \mid \alpha \mid \ldots$
Values	V	$::=$	$x \mid \ldots$

Semantics

$$\begin{array}{lrcl} (\mu) & \langle \mu\alpha.c \| e \rangle & \to & c[\alpha \leftarrow e] \\ (\tilde{\mu}_v) & \langle V \| \tilde{\mu}x.c \rangle & \to & c[x \leftarrow V] \end{array}$$

The solved critical pair

$$\begin{array}{ccc} & \langle \mu\alpha.c \| \tilde{\mu}x.c' \rangle & \\ \text{call-by-value} \quad \swarrow (\mu) & & (\tilde{\mu}) \searrow \quad \text{call-by-name} \\ c[\alpha \leftarrow \tilde{\mu}x.c'] & & c'[x \leftarrow \mu\alpha.c] \end{array}$$

The $\mu\tilde{\mu}$-subsystem

(two confluent symmetric restrictions)

$\mu_n\tilde{\mu}$-subsystem

Commands	c	$::=$	$\langle v \| e \rangle$
Terms	v	$::=$	$\mu\alpha.c \mid x \mid \ldots$
Linear ev. contexts	E	$::=$	$\alpha \mid \ldots$
Evaluation contexts	e	$::=$	$\tilde{\mu}x.c \mid E$

$$(\mu_n)\quad \langle \mu\alpha.c \| E \rangle \rightarrow c[\alpha \leftarrow E]$$
$$(\tilde{\mu})\quad \langle v \| \tilde{\mu}x.c \rangle \rightarrow c[x \leftarrow v]$$

$\mu\tilde{\mu}_v$-subsystem

Commands	c	$::=$	$\langle v \| e \rangle$
Linear terms (= values)	V	$::=$	$x \mid \ldots$
Terms	v	$::=$	$\mu\alpha.c \mid V$
Evaluation contexts	e	$::=$	$\tilde{\mu}x.c \mid \alpha \mid \ldots$

$$(\mu)\quad \langle \mu\alpha.c \| e \rangle \rightarrow c[\alpha \leftarrow e]$$
$$(\tilde{\mu}_v)\quad \langle V \| \tilde{\mu}x.c \rangle \rightarrow c[x \leftarrow V]$$

The principle of duality

Static duality

term	/	evaluation context
value	/	linear ev. context
term variable	/	ev. context variable
μ-binder	/	$\tilde{\mu}$-binder

Semantical duality

call-by-value	/	call-by-name
$(\tilde{\mu}_v)$	/	(μ_n)
(μ)	/	$(\tilde{\mu})$

Connected works

Computational symmetry

- Barbanera-Berardi's symmetric λ-calculus [1996]
- Filinski's symmetric λ-calculus syntax [1988]

Call-by-name/call-by-value duality

- Filinski's dual cont.-passing-style call-by-name and call-by-value semantics of his symmetric λ-calculus [1988]
- Selinger's dual control and co-control categories modelling call-by-name and call-by-value $\lambda\mu$-calculi [2000]

The intuitionistic constraint

evaluation context variables are bound linearly

In extensions that don't bind ev. context variables (e.g. the constructors of implication), this implies one can take a unique ev. context variable, say $\star$.

Example:

Syntax of intuitionistic $\mu\tilde{\mu}^{\rightarrow}$

Commands	c	$::=$	$\langle v \| e \rangle \mid \langle v \| \star \rangle$
Terms	v	$::=$	$\mu v \star e.c \mid x \mid \lambda x.v$
Evaluation contexts	e	$::=$	$\tilde{\mu} x.c v \mid v \cdot \star \mid v \cdot e$

In the intuitionistic case, $=_v$ is a strict subset of $=_n$.

The $\mu\tilde{\mu}$-subsystem

(η-conversions)

$$\begin{array}{lrcll} (\eta_\mu) & \mu\alpha.\langle V \| \alpha \rangle & = & V & \alpha \text{ not free in } V \\ (\eta_{\tilde{\mu}}) & \tilde{\mu}x.\langle x \| E \rangle & = & E & x \text{ not free in } E \end{array}$$

Each extension comes with its own η-conversions. E.g., for the constructors of implication, we have

$$(\eta_\to) \quad \lambda x.\mu\alpha.\langle y \| x \cdot \alpha \rangle \;=\; y$$

from which we derive

$$\begin{array}{lrcll} (\eta_{\to n}) & \lambda x.\mu\alpha.\langle v \| x \cdot \alpha \rangle & = & v & x \text{ and } \alpha \text{ not free in } v \\ (\eta_{\to v}) & \lambda x.\mu\alpha.\langle V \| x \cdot \alpha \rangle & = & V & x \text{ and } \alpha \text{ not free in } V \end{array}$$

Focus on $\overline{\lambda}\mu_n$-calculus and $\overline{\lambda}\tilde{\mu}_v$-calculus

$\overline{\lambda}\mu_n$-calculus

$$\begin{array}{lcl} c & ::= & \langle v \| E \rangle \\ v & ::= & \mu\alpha.c \mid x \mid \lambda x.v \\ E & ::= & \alpha \mid v \cdot E \end{array}$$

Reduction

$$\begin{array}{lll} (\mu_n) & \langle \mu\alpha.c \| E \rangle & \to \; c[\alpha \leftarrow E] \\ (\to_n^\beta) & \langle \lambda x.v \| v' \cdot E \rangle & \to \; \langle v[x \leftarrow v'] \| E \rangle \end{array}$$

η-reduction (with usual constraints)

$$\begin{array}{lll} (\eta_\mu) & \mu\alpha.\langle v \| \alpha \rangle & \to \; v \\ (\eta^R_{\to n}) & v & \to \; \lambda x.\mu\alpha.\langle v \| x \cdot \alpha \rangle \end{array}$$

$\overline{\lambda}\tilde{\mu}_v$-calculus

$$\begin{array}{lcl} c & ::= & \langle V \| e \rangle \\ V & ::= & x \mid \lambda x.\mu\alpha.c \\ e & ::= & \alpha \mid V \cdot e \mid \tilde{\mu}x.c \end{array}$$

Reduction

$$\begin{array}{lll} (\tilde{\mu}_v) & \langle V \| \tilde{\mu}x.c \rangle & \to \; c[x \leftarrow V] \\ (\to_v^\beta) & \langle \lambda x.\mu\alpha.c \| V \cdot e \rangle & \to \; c[x \leftarrow V][\alpha \leftarrow e] \end{array}$$

η-reduction (with usual constraints)

$$\begin{array}{lll} (\eta_{\tilde{\mu}}) & \tilde{\mu}x.\langle x \| e \rangle & \to \; e \\ (\eta^R_{\to v}) & \lambda x.\mu\alpha.\langle V \| x \cdot \alpha \rangle & \to \; V \end{array}$$

Typing the $\mu\tilde{\mu}$-subsystem

(a sequent calculus structure)

- two axioms
- no contraction: simulated by cuts with the axioms
- three kinds of sequents
  - terms: distinguished formula on the right
  - ev. contexts: distinguished formula on the left
  - commands: no distinguished formula

$$\frac{}{\Gamma, x : A \vdash x : A \mid \Delta}\, Ax_R \qquad \frac{}{\Gamma \mid \alpha : A \vdash \alpha : A, \Delta}\, Ax_L$$

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

$$\frac{\Gamma \vdash v : A \mid \Delta \quad \Gamma \mid e : A \vdash \Delta}{\langle v \| e \rangle : (\Gamma \vdash \Delta)}\, Cut$$

Typing extensions (e.g. implication connective)

$$\frac{\Gamma, x : A \vdash v : B \mid \Delta}{\Gamma \vdash \lambda x.v : A \to B \,;\, \Delta} \qquad \frac{\Gamma \vdash v : A \mid \Delta \qquad \Gamma \mid e : B \vdash \Delta}{\Gamma \,;\, v \cdot e : A \to B \vdash \Delta}$$

Typing the $\mu\tilde{\mu}$-subsystem

(sequent calculus in context-free form)

Thanks to the absence of contraction, sequent calculus proofs can be represented *à la* natural deduction

$$\dfrac{\begin{array}{c}[A\vdash]\\ \vdots\\ \vdash\end{array}}{\vdash A}\,\mu \qquad\qquad \dfrac{\begin{array}{c}[\vdash A]\\ \vdots\\ \vdash\end{array}}{A\vdash}\,\tilde{\mu}$$

$$\dfrac{\vdash A \qquad A\vdash}{\vdash}\,Cut$$

$$\dfrac{\begin{array}{c}[\vdash A]\\ \vdots\\ \vdash B\end{array}}{\vdash A\to B}\,{\to_R} \qquad\qquad \dfrac{\vdash A \qquad B\vdash}{A\to B\vdash}\,{\to_L}$$

The computational-content-of-sequent-calculus lineage of $\mu\tilde{\mu}^{\rightarrow}$

- Gentzen's LK sequent calculus [1935]:
  various (implicitly weak) cut-elimination strategies of LK
- Dragalin [1979]: strong cut-elimination of LK

  Griffin's stimulus: classical logic computes in real life

- Girard's LC [1991]: associativity-and-commutativity-preserving $\neg\neg$-translation of LK
- Danos-Joinet-Schellinx's LKT/LKQ fragments of LK [1994]:
  intuition of a connection to call-by-name and call-by-value
- Barbanera-Berardi's λ_{sym}-calculus [1996]: non-deterministic computational content (implicitly?) of sequent calculus
- Herbelin's $\overline{\lambda}$-calculus [1994]: LJT and LKT as a λ-calculus

$$\frac{\begin{array}{c}[\alpha : A \vdash] \\ \vdots \\ c\end{array}}{\vdash \mu\alpha.c : A}\,\mu \qquad \frac{\begin{array}{c}[\vdash x : A] \\ \vdots \\ c\end{array}}{\tilde{\mu}x.c : A \vdash}\,\tilde{\mu}$$

$$\frac{\vdash v : A \qquad e : A \vdash}{\langle v \| e \rangle}\,Cut$$

$$\frac{\begin{array}{c}[\vdash x : A] \\ \vdots \\ \vdash v : B\end{array}}{\vdash \lambda x.v : A \rightarrow B}\,{\rightarrow_R} \qquad \frac{\vdash v : A \qquad e : B \vdash}{v \cdot e : A \rightarrow B \vdash}\,{\rightarrow_L}$$

- Danos-Joinet-Schellinx's analysis of the LK to LL embeddings [1995,1997]
- Urban-Bierman's extension of Barbanera-Berardi's strong normalisation method to an LK syntax [2000]
- Curien-Herbelin $\overline{\lambda}\mu\tilde{\mu}$-calculus [2000]
- Lengrand's λ_{ξ}-calculus [2003]:
  application of the duality-of-computation paradigm to Urban-Bierman's LK syntax
- Wadler's variant of $\overline{\lambda}\mu\tilde{\mu}$-calculus based on disjunctions and conjunctions [2003]

Consequences for call-by-value λ-calculus

Call-by-value interprets the "dark side" of sequent calculus.

Conversely, sequent calculus gives to call-by-value its "nobility".

Call-by-value λ-calculus (natural deduction style) is complex to study.

Can the duality foster further theoretical research on call-by-value?
Böhm theorem, standardisation, complete confluent systems, ...

A complete reduction system

cbv λ-calculus operational rule (Plotkin [1975])

$$(\beta_v) \quad (\lambda x.v)\, V \quad \to \quad v[x \leftarrow V]$$

cbv λ-calculus sub-operational rule (Moggi [1988])

$$(let_{lift}) \quad F[(\lambda x.v)\, v'] \quad \to \quad (\lambda x.F[v])\, v'$$

cbv λ-calculus observational rules (Moggi [1988])

$$(\eta_v) \quad \lambda x.(V\, x) \quad \to \quad V \qquad x \text{ not free in } V$$
$$(\eta_{let}) \quad (\lambda x.E[x])\, v \quad \to \quad E[v] \qquad x \text{ not free in } E$$

cbv $\lambda\mu$-calculus extra operational rules $(*)$

$$(\mu_v) \quad F[\mu\alpha.c] \quad \to \quad \mu\alpha.c[\alpha \leftarrow [\alpha]F]$$
$$(\mu_{var}) \quad [\alpha]\mu\beta.v \quad \to \quad v[\beta \leftarrow [\alpha][\ ]]$$

cbv $\lambda\mu$-calculus extra sub-operational rule $(*)$

$$(\mu_{let}) \quad (\lambda x.\mu\alpha.[\beta]v)\, v' \quad \to \quad \mu\alpha.[\beta]((\lambda x.v)\, v')$$

cbv $\lambda\mu$-calculus extra observational rule $(*)$

$$(\eta_\mu) \quad \mu\alpha.[\alpha]v \quad \to \quad v \qquad \alpha \text{ not free in } V$$

Extra rule for confluence

$$(\mu_v^\eta) \quad v\, \mu\alpha.c \quad \to \quad (\lambda x.\mu\alpha.c[\alpha \leftarrow [\alpha](x\,[\ ])])\, v$$

$(*)$ inspired by Sabry-Felleisen [1993], Hofmann [1995] and Selinger [2000] complete axiomatics

A dual to call-by-need ?

lazy call-by-value
(call-by-need)

Commands	c	$::=$	$\langle v \| e \rangle$
Linear terms (= values)	V	$::=$	$x \mid \ldots$
Terms	v	$::=$	$\mu\alpha.c \mid V$
Linear ev. contexts	E	$::=$	$\alpha \mid \ldots$
Evaluation contexts	e	$::=$	$\tilde{\mu}x.c \mid E$

(μ_n)	$\langle \mu\alpha.c \| E \rangle$	$\rightarrow_{lv}$	$c[\alpha \leftarrow E]$	
$(\tilde{\mu}_v)$	$\langle V \| \tilde{\mu}x.c \rangle$	$\rightarrow_{lv}$	$c[x \leftarrow V]$	
(μ_{lv})	$\langle \mu\alpha.c \| \tilde{\mu}x.c' \rangle$	$\rightarrow_{lv}$	$c[\alpha \leftarrow \tilde{\mu}x.c']$	$(*)$
$(\tilde{\mu}_{lv})$	$\langle \mu\alpha.c \| \tilde{\mu}x.c' \rangle$	$\rightarrow_{lv}$	c'	$(**)$

$(*)$ if x "needed" in c'
$(**)$ if $x \notin FV(c')$

intuitionistic restriction
observationally collapses to call-by-name

lazy call-by-name

Commands	c	$::=$	$\langle v \| e \rangle$
Linear terms (= values)	V	$::=$	$x \mid \ldots$
Terms	v	$::=$	$\mu\alpha.c \mid V$
Linear ev. contexts	E	$::=$	$\alpha \mid \ldots$
Evaluation contexts	e	$::=$	$\tilde{\mu}x.c \mid E$

?

(μ_n)	$\langle \mu\alpha.c \| E \rangle$	$\rightarrow_{ln}$	$c[\alpha \leftarrow E]$	
$(\tilde{\mu}_v)$	$\langle V \| \tilde{\mu}x.c \rangle$	$\rightarrow_{ln}$	$c[x \leftarrow V]$	
(μ_{ln})	$\langle \mu\alpha.c \| \tilde{\mu}x.c' \rangle$	$\rightarrow_{ln}$	c	$(**)$
$(\tilde{\mu}_{ln})$	$\langle \mu\alpha.c \| \tilde{\mu}x.c' \rangle$	$\rightarrow_{ln}$	$c'[x \leftarrow \mu\alpha.c]$	$(*)$

$(*)$ if α "needed" in c
$(**)$ if $\alpha \notin FV(c)$

intuitionistic restriction
operationally collapses to call-by-name

with control, all four reduction systems are observationally different

Coinductive and inductive types

(Paulin-Mohring's fixpoint-based decomposition)

Coinductive type
= term constructors
+ ev. ctx. constructors
+ term recursion guarded by a term constructor

Inductive type
= term constructors
+ ev. ctx. constructors
+ ev. ctx. recursion guarded by an ev. ctx. constructor

E x a m p l e s

A coinductive type (possibly infinite lists)

$$\begin{array}{rcl} V & ::= & \ldots \mid \mathtt{nil} \mid \mathtt{cons}(v, v) \mid \nu_x.V_c \\ E & ::= & \ldots \mid [\mathtt{nil}.c, \mathtt{cons}(x_a, x_l).c] \end{array}$$

$$\begin{array}{lrcl} (\mathtt{nil}) & \langle \mathtt{nil} \| [\mathtt{nil}.c, \mathtt{cons}(x_a, x_l).c'] \rangle & \to & c \\ (\mathtt{cons}) & \langle \mathtt{cons}(v_a, v_l) \| [\mathtt{nil}.c, \mathtt{cons}(x_a, x_l).c'] \rangle & \to & \langle v_a \| \tilde{\mu} x_a. \langle v_l \| \tilde{\mu} x_l. c' \rangle \rangle \end{array}$$

$$(\nu) \quad \langle \nu_x.V_c \| E_c \rangle \to \langle V_c[x \leftarrow \nu_x.V_c] \| E_c \rangle$$

An inductive type (lists)

$$\begin{array}{rcl} V & ::= & \ldots \mid \mathtt{nil} \mid \mathtt{cons}(v, v) \\ E & ::= & \ldots \mid [\mathtt{nil}.c, \mathtt{cons}(x_a, x_l).c] \mid \tilde{\nu}_\alpha.E_c \end{array}$$

$$(\tilde{\nu}) \quad \langle V_c \| \tilde{\nu}_\alpha.E_c \rangle \to \langle V_c \| E_c[\alpha \leftarrow \tilde{\nu}_\alpha.E_c] \rangle$$

V_c means constructed V
E_c means constructed E

The proof-as-strategy approach

strategies in game model of computations are polarised normal proofs in some sequent calculus

Lorenz-Lorenzen's game semantics of provability
strategies = cut-free proofs in LJQ/LKQ (Herbelin's PhD [1995])

different possible polarisations of $\mu\tilde{\mu}$-systems normal proofs

decompose normal proofs along commands: yields (abstract) Böhm trees (Curien-Herbelin [1998])

$\langle x \| E \rangle$ = "(" (question) with possible reactions determined by E

$\langle V \| \alpha \rangle$ = "]" (answer) with possible reactions determined by V

game interaction = head reduction in an abstract machine (Danos-Herbelin-Regnier [1996])
= head reduction in $\mu\tilde{\mu}$

intuitionistic restriction = well-bracketed parentheses
(Lorenzen's school [see Felscher 1986])

Simply-typed $\mu_n\tilde{\mu}^{\to\mathbb{N}}$-system ($\mu$PCF)
(Herbelin [1997], Laird [1997])

$N \to N$ interpreted as $?N^{\perp} \parr N$
$\mathbb{N}$ interpreted as $? \oplus_n 1$
maximal $\eta_{\to n}$-expansion
maximal η_{μ}-expansion of atoms

$$c \quad ::= \quad \langle x_i^j \| v_1 \cdot \ldots \cdot v_p \cdot [\mathtt{n} \mapsto c_{\mathtt{n}}] \rangle \mid \langle \mathtt{n} \| \alpha_i \rangle$$

where $v ::= \lambda x_1 \ldots x_n . \mu\alpha . c$

$$c \quad ::= \quad \overset{(_i^j}{\underset{[_0^1 \quad \cdots \quad [_0^p \quad)_0^{\mathtt{n}}}{\swarrow \quad \searrow \quad \searrow}} \quad \mid \quad]_i^{\mathtt{n}}$$

Initial state

$$\langle x \| \overbrace{v_1 \cdot \ldots \cdot v_p \cdot [\mathtt{n} \mapsto c_{\mathtt{n}}]}^{Opponent} \rangle \quad [x \leftarrow \overbrace{v}^{Player}]$$

Interaction rules

$$\begin{array}{llllll} (\to \mu_n) & \langle x_i^j \| \vec{v} \cdot [\mathtt{n} \mapsto c_{\mathtt{n}}] \rangle & [\sigma] & \to & c & [\vec{x} \leftarrow \vec{v}; \alpha \leftarrow [\mathtt{n} \mapsto c_{\mathtt{n}}]; \sigma'] \\ (\mathbb{N}) & \langle \mathtt{n} \| \alpha_i \rangle & [\sigma] & \to & c_{\mathtt{n}} & [\sigma'] \end{array}$$

$$\sigma(x_i^j) = (\lambda \vec{x} . \mu\alpha . c)[\sigma'] \quad \sigma(\alpha_i) = ([\mathtt{n} \mapsto c_{\mathtt{n}}])[\sigma']$$

Simply-typed $\mu\tilde{\mu}_v^{\to\mathbb{N}}$-system ($\mu$PCF$_v$)
(Abramsky-McCusker [1997], Honda-Yoshida [1997], Laird [1998])

$P \to P$ interpreted as $P^{\perp} \parr !P$
$\mathbb{N}$ interpreted as $? \oplus_n 1$
maximal $\eta_{\to v}$-expansion and $\eta_{\tilde{\mu}}$-expansion
needs new constructions $\lambda\!\!\lambda \mathtt{n} . v_{\mathtt{n}}$ and $\mathtt{n} \cdot e$

$$c \quad ::= \quad \langle x_i \| V_\lambda \cdot e \rangle \mid \langle x_i \| \mathtt{n} \cdot e \rangle \mid \langle V_\lambda \| \alpha_i \rangle \mid \langle \mathtt{n} \| \alpha_i \rangle$$

where
$$\begin{array}{l} V_\lambda ::= \lambda x . \mu\alpha . c \mid \lambda\!\!\lambda \mathtt{n} . \mu\alpha . c_{\mathtt{n}} \\ e \ ::= \tilde{\mu} x . c \mid [\mathtt{n} \mapsto c_{\mathtt{n}}] \end{array}$$

$$c \quad ::= \quad \overset{(_i^{\lambda}}{\underset{[_0^V \qquad)_0^V}{\swarrow \quad \searrow}} \quad \mid \quad \overset{(_i^{\mathtt{n}}}{\underset{)_0^V}{\downarrow}} \quad \mid \quad]_i^{\lambda} \quad \mid \quad]_i^{\mathtt{n}}$$

Initial states

$$\begin{array}{ll} \langle \overbrace{[\ \mathtt{n}\ ]}^{Player} \| \alpha \rangle & [\alpha \leftarrow \overbrace{[\mathtt{n} \mapsto c_{\mathtt{n}}]}^{Opponent}] \\ \langle \lambda x . \mu\alpha . c \ \| \alpha \rangle & [\alpha \leftarrow V_\lambda \cdot e] \\ \langle \lambda\!\!\lambda \mathtt{n} . \mu\alpha . c_{\mathtt{n}} \| \alpha \rangle & [\alpha \leftarrow \mathtt{n} \cdot e] \end{array}$$

Interaction rules

$$\begin{array}{llllll} (\to \mu) & \langle x_i \| V_\lambda \cdot e \rangle & [\sigma] & \to & c & [x \leftarrow V_\lambda; \alpha \leftarrow e; \sigma'] \\ (\to^{\mathbb{N}} \mu) & \langle x_i' \| \mathtt{n} \cdot e \rangle & [\sigma] & \to & c_{\mathtt{n}} & [\alpha \leftarrow e; \sigma'] \\ (\tilde{\mu}_v) & \langle V_\lambda \| \alpha_i \rangle & [\sigma] & \to & c & [x \leftarrow V_\lambda; \sigma'] \\ (\mathbb{N}) & \langle \mathtt{n} \| \alpha_i' \rangle & [\sigma] & \to & c_{\mathtt{n}} & [\sigma'] \end{array}$$

$$\begin{array}{ll} \sigma(x_i) = (\lambda x . \mu\alpha . c)[\sigma'] & \sigma(\alpha_i) = (\tilde{\mu} x . c)[\sigma'] \\ \sigma(x_i') = (\lambda\!\!\lambda \mathtt{n} . \mu\alpha . c_{\mathtt{n}})[\sigma'] & \sigma(\alpha_i') = ([\mathtt{n} \mapsto c_{\mathtt{n}}])[\sigma'] \end{array}$$

Control delimiters

Felleisen [1988], Danvy-Filinski [1989]
control delimiters

Sitaram-Felleisen [1990]
PCF with control and delimiters is complete wrt domains (up to parallel-if)

Filinski [1994]
control and delimiters have the expressiveness of monads

Ariola-Herbelin-Sabry [2004]
$\lambda\mu$-calculus + control delimiters
delimiter = dynamic binding of the "toplevel continuation"

Would the addition of delimiters and (intuitionistic restrictions of) control be the right way to provide direct-style monadic extensions in intuitionistic type theory?

David-Py [2001]
call-by-name $\lambda\mu$-calculus with η is not complete (separation contexts missing)

Saurin [2005]
call-by-name $\lambda\mu$-calculus extended with operators "similar" to delimiters is complete

Are delimiters the ingredients for a Böhm theorem for call-by-name and call-by-value λ-calculus with control?

Abstract Computing Devices

(collecting the ingredients)

Hardin-Maranget-Pagano [1996]
relevance of explicit substitution to represent environments in abstract devices

Abstract machines characterised by reducing at the root of the computation

The $\mu\tilde{\mu}$-system:

- has a primitive notion of evaluation contexts ("stacks")
- has a primitive notion of "states" (the commands)
- redex of non head normal states are at the root

Add an (ad hoc) variable numbering schemes

All ingredients are here to simulate abstract machines

The $\mu\tilde{\mu}$-subsystem

(non-deterministic union)

$$
\begin{array}{rcl}
c & ::= & \langle v \| e \rangle \mid \{c, c\} \\
V & ::= & x \mid \ldots \\
v & ::= & \mu\alpha.c \mid V \\
E & ::= & \alpha \mid \ldots \\
e & ::= & \tilde{\mu}x.c \mid E
\end{array}
$$

$$
\begin{array}{lrcl}
(\cup) & \langle \mu\alpha.c \| \tilde{\mu}x.c' \rangle & \rightarrow & \{c[\alpha \leftarrow \tilde{\mu}x.c'], c'[x \leftarrow \mu\alpha.c]\} \\
(\mu_n) & \langle \mu\alpha.c \| E \rangle & \rightarrow & c[\alpha \leftarrow E] \\
(\tilde{\mu}_v) & \langle V \| \tilde{\mu}x.c \rangle & \rightarrow & c[x \leftarrow V]
\end{array}
$$

Open questions

- Which valid equations about $\{c, c'\}$ (associativity, commutativity, distributivity)?
- Relation with Kosta Došen and Zoran Petrić's categories of distributive lattices?

Beyond substitution-based computation: cross-cuts

Cross-cuts have been introduced by Gentzen [1938].

They provide some form of computational optimisation.

They are not reducible to substitution.

Can they smoothly be taken into account in the system $\mu\tilde{\mu}$?

(see also Coquand-Herbelin's unpublished analysis of symmetric !/! critical pair [1994])

Duality and dependent types

(when left-right symmetry meets left-right dependency)

	call-by-name	call-by-value
explicit dependent product (type-theoretic)	OK (but bypass $\tilde{\mu}$)	OK (but applied to value only)
explicit dependent sum (type-theoretic)	classical case degenerated (Herbelin [2005])	

Summary

The $\mu\tilde{\mu}$-subsystem is an elegant tool to investigate the duality properties of the computation, and to revisit the foundations of λ-calculus.

The duality finds limits in dependent type theory.

Et maintenant...

(and now...)

Arrêté du 23 novembre 1988 relatif à l'habilitation à diriger des recherches (article 7, extraits)

« La présentation des travaux est publique. »

"The presentation of works is public."

« Le candidat fait devant le jury un exposé sur l'ensemble de ses travaux (...). Cet exposé donne lieu à une discussion avec le jury. »

"The candidate gives a talk on his whole works (...). The talk is followed by a discussion with the the committee."

« Le jury procède à un examen de la valeur de candidat, évalue sa capacité à *concevoir*, *diriger*, *animer* et *coordonner* des activités de recherche et de *valorisation* et statue sur la délivrance de l'habilitation. »

"The committee conducts an examination of the value of the candidate, evaluates its ability to conceive, head, animate *and* arrange *research and* valorisation *activities and gives a decision on the delivery of the habilitation"*

What about Coq?

(Mister Hyde)

Strong involvement in the Coq development projects (1999-2004)

- Coq version 7 (enforcement of de Bruijn principle, re-appropriation of Coq code)
- Coq version 8 (uniform and scalable revision of the concrete syntax)

A few visible features (induction tactics, let-in construct, nested pattern-matching, mathematical notations) and more discrete ones (general maintenance, bug fixes).

Many projects for improvements now stem from outside the LogiCal team.

How to make the Logical contribution to the Coq project evolves?