The benefits of sequent calculus

Étienne Miquey

Équipe Gallinette, INRIA
LS2N, Université de Nantes

Logic Mentoring Workshop
22/06/2019

Forewords

This talk is about:

sequent calculus / Curry-Howard / operational semantics

But also : *proofs, programs, type systems, safe computation/compilation, ...*

Sequent calculus provides wonderful tools!

Gives **principled answers** to problems such as:

- how to soundly compile XXX?
- how to prove normalization of XXX?
- how should control operators and XXX interact?
- deciding the equivalence of normal forms

Forewords

This talk is about:

sequent calculus / Curry-Howard / operational semantics

But also : *proofs, programs, type systems, safe computation/compilation, ...*

A fairy tale

Sequent calculus provides wonderful tools!

Gives **principled answers** to problems such as:

- how to soundly compile xxx?
- how to prove normalization of xxx?
- how should control operators and xxx interact?
- deciding the equivalence of normal forms

Forewords

This talk is about:

sequent calculus / Curry-Howard / operational semantics

But also : *proofs, programs, type systems, safe computation/compilation, ...*

A fairy tale

Sequent calculus provides wonderful tools!

Gives **principled answers** to problems such as:

- how to soundly compile ×××?
- how to prove normalization of ×××?
- how should control operators and ××× interact?
- deciding the equivalence of normal forms

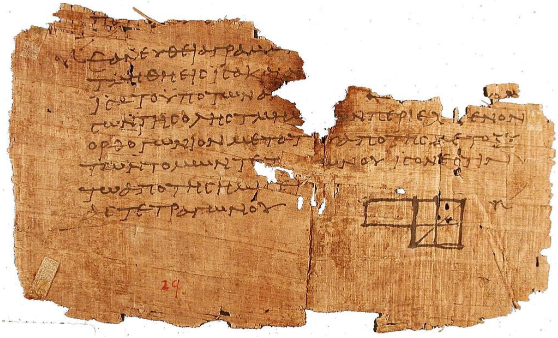
Proofs

A bit of history, fast-tracked

Once upon a time...

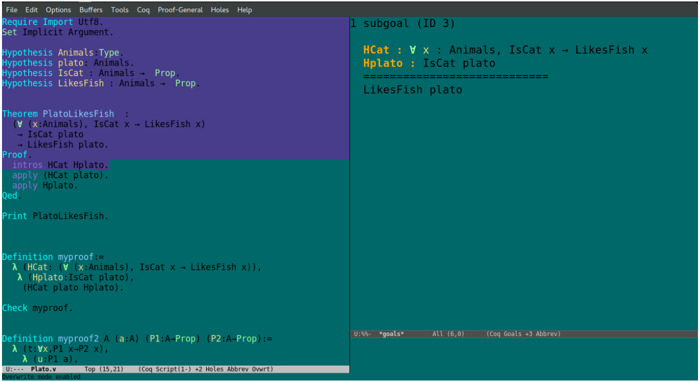
-300

Euclide

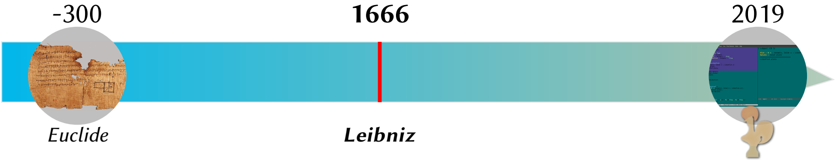


Euclide's Elements

Once upon a time...



Once upon a time...

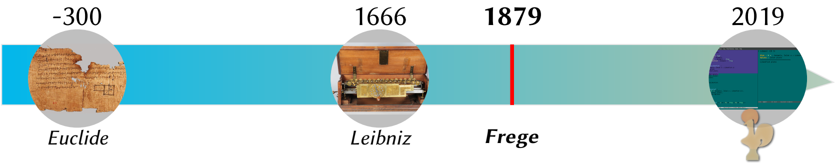


Leibniz's calculus ratiocinator

A crazy dream:

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

Once upon a time...

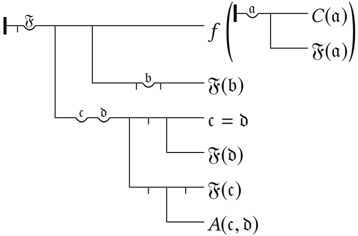


Frege's *Begriffsschrift*:

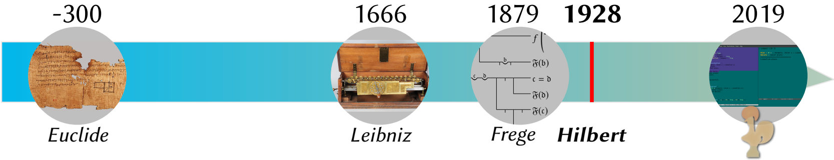
- formal notations
- quantifications ∀/∃
- distinction:

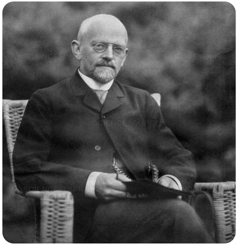
  x vs $'x'$

  signified *signifier*



Once upon a time...





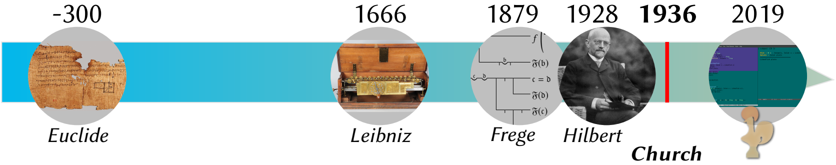
Hilbert

Entscheidungsproblem (with Ackermann):

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

↬ "**to decide**" is meant by means of a procedure

Once upon a time...



Church

λ-calculus - first (negative) answer to the *Entscheidungsproblem* !

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

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

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

A somewhat obvious observation

Deduction rules

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

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

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

Typing rules

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

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

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

Sequent, you said?

Sequent:

Hypotheses $\boxed{A_1, \ldots, A_n \vdash B}$ Conclusion

Remark:

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B}(\Rightarrow_I) \quad \text{is almost} \quad \frac{\begin{array}{c}[A]\\ \vdots\\ B\end{array}}{A \Rightarrow B}(\Rightarrow_I)$$

"*à la Gentzen*" "*à la Prawitz*"

... a.k.a. natural deduction

Sequent, you said?

Sequent:

Hypotheses $\boxed{A_1, \ldots, A_n \vdash B}$ Conclusion

Remark:

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

is almost

$$\frac{\begin{array}{c}[A] \\ \vdots \\ B\end{array}}{A \Rightarrow B}(\Rightarrow_I)$$

"à la Gentzen" *"à la Prawitz"*

… *a.k.a.* **natural deduction**

Gentzen's sequent calculus (1934)

Sequent:

Hypotheses $\boxed{A_1, \ldots, A_n \vdash B_1, \ldots, B_p}$ Conclusion<u>s</u>

Identity rules *connect hypotheses/conclusions*

$$\frac{\Gamma \vdash A, \Delta \qquad \Gamma, A \vdash \Delta}{\Gamma \vdash \Delta}\ (\textsc{Cut}) \qquad\qquad \frac{}{A \vdash A}\ (\textsc{Ax})$$

Structural rules *weaken, contract, permute*

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash A, \Delta}\ (w_r) \qquad \frac{\Gamma \vdash A, A, \Delta}{\Gamma \vdash A, \Delta}\ (c_r) \qquad \frac{\Gamma \vdash \sigma(\Delta)}{\Gamma \vdash \Delta}\ (\sigma_r) \qquad \ldots$$

Logical rules *left/right introduction of connectives*

$$\frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \Rightarrow B, \Delta}\ (\Rightarrow_r) \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \Rightarrow B \vdash \Delta}\ (\Rightarrow_l) \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta}\ (\wedge_r)$$

Gentzen's sequent calculus (1934)

Sequent:

Hypotheses $\boxed{A_1, \ldots, A_n \vdash B_1, \ldots, B_p}$ Conclusions

Identity rules *connect hypotheses/conclusions*

$$\frac{\Gamma \vdash A, \Delta \qquad \Gamma, A \vdash \Delta}{\Gamma \vdash \Delta}\ (\textsc{Cut}) \qquad\qquad \frac{}{A \vdash A}\ (\mathrm{Ax})$$

Structural rules *weaken, contract, permute*

$$\frac{\Gamma \vdash \Delta}{\Gamma \vdash A, \Delta}\ (w_r) \qquad \frac{\Gamma \vdash A, A, \Delta}{\Gamma \vdash A, \Delta}\ (c_r) \qquad \frac{\Gamma \vdash \sigma(\Delta)}{\Gamma \vdash \Delta}\ (\sigma_r) \qquad \cdots$$

Logical rules *left/right introduction of connectives*

$$\frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \Rightarrow B, \Delta}\ (\Rightarrow_r) \qquad \frac{\Gamma \vdash A, \Delta \qquad \Gamma, B \vdash \Delta}{\Gamma, A \Rightarrow B \vdash \Delta}\ (\Rightarrow_l) \qquad \frac{\Gamma \vdash A, \Delta \qquad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta}\ (\wedge_r)$$

Gentzen's sequent calculus (1934)

Sequent:

Hypotheses $\boxed{A_1, \ldots, A_n \vdash B_1, \ldots, B_p}$ Conclusions

Proof-theoretic properties:

- cut elimination
- last rule
- subformula
- classical logic built-in
- ...

Gentzen's sequent calculus (1934)

Sequent:

Hypotheses $\boxed{A_1, \ldots, A_n \vdash B_1, \ldots, B_p}$ Conclusion<u>s</u>

Identity rules *connect hypotheses/conclusions*

Symmetry

Logical rules *left/right introduction of connectives*

$$\frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \Rightarrow B, \Delta}\ (\Rightarrow_r) \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma, B \vdash \Delta}{\Gamma, A \Rightarrow B \vdash \Delta}\ (\Rightarrow_l) \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta}\ (\wedge_r)$$

What about the computational content?

Curien-Herbelin's duality of computation

Griffin (1990): classical logic ≅ control operator

Starting observation:

Computational duality:

$$\begin{tikzcd} \textbf{Terms} \arrow[r, bend left, "\cdot^{\perp}"] & \textbf{Contexts} \arrow[l, bend left, "\cdot^{\perp}"] \end{tikzcd}$$

Sequent calculus ≅ *abstract machine-like calculus*

Curien-Herbelin's duality of computation

Griffin (1990): classical logic ≅ control operator

Starting observation:

calculus and $\lambda\mu$-calculus. Our starting point was the observation that the call-by-value discipline manipulates input much in the same way as (the classical extension of) λ-calculus manipulates output. Computing MN in call-by-

Computational duality:

$$\begin{tikzcd} \textbf{Terms} \arrow[r, bend left, ".^{\perp}"] & \textbf{Contexts} \arrow[l, bend left, ".^{\perp}"] \end{tikzcd}$$

Sequent calculus ≅ *abstract machine-like calculus*

Curien-Herbelin's duality of computation

Griffin (1990): classical logic ≅ control operator

Starting observation:

calculus and $\lambda\mu$-calculus. Our starting point was the observation that the call-by-value discipline manipulates input much in the same way as (the classical extension of) λ-calculus manipulates output. Computing MN in call-by-

Computational duality:

$$
\begin{tikzcd}
\textbf{Terms} \arrow[r, bend left, ".^{\perp}"] & \textbf{Contexts} \arrow[l, bend left, ".^{\perp}"]
\end{tikzcd}
$$

Sequent calculus ≅ *abstract machine-like calculus*

Abstract machine

Reduction

$$\begin{array}{rcl} \langle t\ u \parallel e \rangle & \triangleright_{\text{abs}} & \langle t \parallel u \cdot e \rangle \\ \langle \lambda x.\, t \parallel u \cdot e \rangle & \triangleright_{\text{abs}} & \langle t\,[u/x] \parallel e \rangle \end{array}$$

Syntax

$$c ::= \langle t \parallel e \rangle \qquad \text{commands}$$

terms	$t, u ::=$		$e, f ::=$	contexts
variable	$\mid x, y, z$		$\mid \star$	empty
application	$\mid t\ u$		$\mid t \cdot e$	application stack
λ-abstraction	$\mid \lambda x.\, t$			

Introducing μ

$$\langle t\ u \parallel e \rangle \rhd_{\text{abs}} \langle t \parallel u \cdot e \rangle$$

This reduction **defines** $(t\ u)$:

It is the term that, when put against $\mid e\rangle$*, reduces to* $\langle t \parallel u \cdot e \rangle$*.*

Idea: introduce a more primitive syntax

$$\langle \mu\alpha . c \parallel e \rangle \rhd_{\mu} c\,[e/\alpha]$$

$$t\ u \quad \triangleq \quad \mu\alpha . \langle t \parallel u \cdot \alpha \rangle$$

(actually the intuitionistic version $\mu\star.c$ is enough)

Introducing μ

$$\langle t\ u \parallel e \rangle \rhd_{\text{abs}} \langle t \parallel u \cdot e \rangle$$

This reduction **defines** $(t\ u)$:

It is the term that, when put against $\mid e\rangle$, reduces to $\langle t \parallel u \cdot e \rangle$.

Idea: introduce a more primitive syntax

$$\langle \mu\alpha.\, c \parallel e \rangle \rhd_{\mu} c\,[e/\alpha]$$

$$t\ u \quad \triangleq \quad \mu\alpha.\, \langle t \parallel u \cdot \alpha \rangle$$

(actually the intuitionistic version $\mu\star.c$ is enough)

Introducing $\tilde{\mu}$

A regular syntax?

$$c ::= \langle t \parallel e \rangle$$

$$t, u ::= \; \mid x, y \; \mid \lambda x.t \; \mid \mu\alpha.c \qquad\qquad e, f ::= \; \mid \alpha, \beta \; \mid t \cdot e \; \mid \; ?$$

Reminder:

calculus and $\lambda\mu$-calculus. Our starting point was the observation that the call-by-value discipline manipulates input much in the same way as (the classical extension of) λ-calculus manipulates output. Computing MN in call-by-

Introducing $\tilde{\mu}$

A regular syntax?

$$c ::= \langle t \parallel e \rangle$$

$$t, u ::= \mid x, y \mid \lambda x.t \mid \mu\alpha.c \qquad e, f ::= \mid \alpha, \beta \mid t \cdot e \mid \boxed{?}$$

Same idea, in the **dual situation**:

$$\langle (\lambda x.t)u \parallel e \rangle \triangleright_{\text{abs}} \langle \text{let } x = t \text{ in } u \parallel e \rangle \quad \triangleright_{\text{abs}} \quad \langle t \parallel \text{“let } x = \square \text{ in } \langle u \parallel e \rangle\text{''} \rangle$$

$$\langle t \parallel \tilde{\mu}x.c \rangle \quad \triangleright_{\tilde{\mu}} \quad c\,[t/x]$$

Introducing $\tilde{\mu}$

A regular syntax

$$c ::= \langle t \parallel e \rangle$$

$$t, u ::= \;\mid x, y \;\mid \lambda x.t \;\mid \mu\alpha.c \qquad e, f ::= \;\mid \alpha, \beta \;\mid t \cdot e \;\mid \tilde{\mu}x.c$$

Same idea, in the **dual situation**:

$$\langle (\lambda x.t)u \parallel e \rangle \rhd_{\text{abs}} \langle \text{let } x = t \text{ in } u \parallel e \rangle \quad \rhd_{\text{abs}} \quad \langle t \parallel \underbrace{\text{“let } x = \Box \text{ in } \langle u \parallel e \rangle\text{''}}_{\tilde{\mu}x.\langle u \parallel e \rangle} \rangle$$

$$\langle t \parallel \tilde{\mu}x.c \rangle \quad \rhd_{\tilde{\mu}} \quad c\,[t/x]$$

Curien-Herbelin's $\lambda\mu\tilde{\mu}$-calculus

Syntax:

$$
\begin{array}{lll}
t, u ::= & c ::= \langle t \parallel e \rangle & e, f ::= \\
\quad \mid x, y & & \quad \mid \alpha, \beta \\
\quad \mid \lambda x.t & & \quad \mid t \cdot e \\
\quad \mid \mu\alpha.\, c & & \quad \mid \tilde{\mu}x.\, c
\end{array}
$$

Reduction:

$$
\begin{array}{rcl}
\langle \lambda x.t \parallel u \cdot e \rangle & \longrightarrow & \big\langle u \,\big\|\, \tilde{\mu}x.\langle t \parallel e \rangle \big\rangle \\
\langle t \parallel \tilde{\mu}x.c \rangle & \longrightarrow & c[t/x] \\
\langle \mu\alpha.c \parallel e \rangle & \longrightarrow & c[e/\alpha]
\end{array}
$$

Curien-Herbelin's $\lambda\mu\tilde{\mu}$-calculus

Syntax:

$$t, u ::= \mid x, y \mid \lambda x.t \mid \mu\alpha.c \qquad c ::= \langle t \parallel e \rangle \qquad e, f ::= \mid \alpha, \beta \mid t \cdot e \mid \tilde{\mu}x.c$$

Reduction:

$$\begin{array}{rcl}
\langle \lambda x.t \parallel u \cdot e \rangle & \longrightarrow & \big\langle u \parallel \tilde{\mu}x.\langle t \parallel e \rangle \big\rangle \\
\langle t \parallel \tilde{\mu}x.c \rangle & \longrightarrow & c[t/x] \\
\langle \mu\alpha.c \parallel e \rangle & \longrightarrow & c[e/\alpha]
\end{array}$$

Critical pair:

$$\begin{array}{ccc}
 & \langle \mu\alpha.c \parallel \tilde{\mu}x.c' \rangle & \\
\swarrow & & \searrow \\
c[\tilde{\mu}x.c'/\alpha] & & c'[\mu\alpha.c/x]
\end{array}$$

Curien-Herbelin's $\lambda\mu\tilde{\mu}$-calculus

Syntax:

$$
\begin{array}{lll}
t,u ::= & c ::= \langle t \,\|\, e\rangle & e,f ::= \\
\text{Values}\left\{\begin{array}{l} \mid x,y \\ \mid \lambda x.t \end{array}\right. & & \left.\begin{array}{l} \mid \alpha,\beta \\ \mid t\cdot e \end{array}\right\}\text{Co-values} \\
\mid \mu\alpha.\,c & & \mid \tilde{\mu}x.\,c
\end{array}
$$

Reduction:

$$
\begin{array}{rcll}
\langle \lambda x.t \,\|\, u\cdot e\rangle & \to & \langle u \,\|\, \tilde{\mu}x.\langle t \,\|\, e\rangle\rangle & \\
\langle t \,\|\, \tilde{\mu}x.c\rangle & \to & c[t/x] & t\in\mathcal{V} \\
\langle \mu\alpha.c \,\|\, e\rangle & \to & c[e/\alpha] & e\in\mathcal{E}
\end{array}
$$

Critical pair:

$$
c[\tilde{\mu}x.c'/\alpha] \xleftarrow{\text{CbV}} \langle \mu\alpha.c \,\|\, \tilde{\mu}x.c'\rangle \xrightarrow{\text{CbN}} c'[\mu\alpha.c/x]
$$

Curien-Herbelin's $\lambda\mu\tilde{\mu}$-calculus

Syntax:

$$
\begin{array}{lll}
t,u ::= & c ::= \langle t \,\|\, e\rangle & e,f ::= \\
\text{Values}\left\{\begin{array}{l} \mid x,y \\ \mid \lambda x.t \end{array}\right. & & \left.\begin{array}{l} \mid \alpha,\beta \\ \mid t\cdot e \end{array}\right\}\text{Co-values} \\
\quad \mid \mu\alpha.c & & \quad \mid \tilde{\mu}x.c
\end{array}
$$

Reduction:

$$
\begin{array}{rcll}
\langle \lambda x.t \,\|\, u\cdot e\rangle & \to & \langle u \,\|\, \tilde{\mu}x.\langle t \,\|\, e\rangle\rangle & \\
\langle t \,\|\, \tilde{\mu}x.c\rangle & \to & c[t/x] & t \in \mathcal{V} \\
\langle \mu\alpha.c \,\|\, e\rangle & \to & c[e/\alpha] & e \in \mathcal{E}
\end{array}
$$

Critical pair:

$$
c[\tilde{\mu}x.c'/\alpha] \xleftarrow{\text{CbV}} \langle \mu\alpha.c \,\|\, \tilde{\mu}x.c'\rangle \xrightarrow{\text{CbN}} c'[\mu\alpha.c/x]
$$

$\boxed{\text{Duality (again and again)}}$

Curien-Herbelin's $\lambda\mu\tilde{\mu}$-calculus

Syntax:

$$
\begin{array}{llll}
 & t,u ::= & c ::= \langle t \parallel e \rangle & e,f ::= \\
\text{Values} \left\{ \begin{array}{l} \\ \\ \end{array} \right. & \mid x,y & & \mid \alpha,\beta \\
 & \mid \lambda x.t & & \mid t\cdot e \quad \left. \begin{array}{l} \\ \\ \end{array} \right\} \text{Co-values} \\
 & \mid \mu\alpha.c & & \mid \tilde{\mu}x.c
\end{array}
$$

Typing rules:

$$\frac{\Gamma \vdash t : A \mid \Delta \qquad \Gamma \mid e : A \vdash \Delta}{\langle t \parallel e \rangle : (\Gamma \vdash \Delta)}$$

$$\frac{(x : A) \in \Gamma}{\Gamma \vdash x : A \mid \Delta} \qquad \frac{\Gamma, x : A \vdash t : B \mid \Delta}{\Gamma \vdash \lambda x.t : A \to B \mid \Delta} \qquad \frac{c : (\Gamma \vdash \Delta, \alpha : A)}{\Gamma \vdash \mu\alpha.c : A \mid \Delta}$$

$$\frac{(\alpha : A) \in \Delta}{\Gamma \mid \alpha : A \vdash \Delta} \qquad \frac{\Gamma \vdash u : A \mid \Delta \qquad \Gamma \mid e : B \vdash \Delta}{\Gamma \mid u \cdot e : A \to B \vdash \Delta} \qquad \frac{c : (\Gamma, x : A \vdash \Delta)}{\Gamma \mid \tilde{\mu}x.c : A \vdash \Delta}$$

Curien-Herbelin's $\lambda\mu\tilde{\mu}$-calculus

Syntax:

$$t, u ::= \underbrace{\mid x, y \mid \lambda x.t}_{\text{Values}} \mid \mu\alpha.c \qquad c ::= \langle t \,\|\, e \rangle \qquad e, f ::= \underbrace{\mid \alpha, \beta \mid t \cdot e}_{\text{Co-values}} \mid \tilde{\mu}x.c$$

Typing rules:

$$\frac{\Gamma \vdash A \mid \Delta \qquad \Gamma \mid A \vdash \Delta}{(\Gamma \vdash \Delta)}$$

$$\frac{A \in \Gamma}{\Gamma \vdash A \mid \Delta} \qquad \frac{\Gamma, A \vdash B \mid \Delta}{\Gamma \vdash A \to B \mid \Delta} \qquad \frac{\Gamma \vdash \Delta, A}{\Gamma \vdash A \mid \Delta}$$

$$\frac{A \in \Delta}{\Gamma \mid A \vdash \Delta} \qquad \frac{\Gamma \vdash A \mid \Delta \qquad \Gamma \mid B \vdash \Delta}{\Gamma \mid A \to B \vdash \Delta} \qquad \frac{\Gamma, A \vdash \Delta}{\Gamma \mid A \vdash \Delta}$$

Curien-Herbelin's $\lambda\mu\tilde{\mu}$-calculus

Syntax:

$$
\begin{array}{lll}
t,u ::= & c ::= \langle t \parallel e \rangle & e,f ::= \\
\quad \mid x,y & & \quad \mid \alpha,\beta \\
\quad \mid \lambda x.t & & \quad \mid t \cdot e \\
\quad \mid \mu\alpha.c & & \quad \mid \tilde{\mu}x.c
\end{array}
$$

Values: x,y and $\lambda x.t$. Co-values: α,β and $t \cdot e$.

Typing rules:

$$\frac{\Gamma \vdash t : A \mid \Delta \qquad \Gamma \mid e : A \vdash \Delta}{\langle t \parallel e \rangle : (\Gamma \vdash \Delta)}$$

$$\frac{(x : A) \in \Gamma}{\Gamma \vdash x : A \mid \Delta} \qquad \frac{\Gamma, x : A \vdash t : B \mid \Delta}{\Gamma \vdash \lambda x.t : A \to B \mid \Delta} \qquad \frac{c : (\Gamma \vdash \Delta, \alpha : A)}{\Gamma \vdash \mu\alpha.c : A \mid \Delta}$$

$$\frac{(\alpha : A) \in \Delta}{\Gamma \mid \alpha : A \vdash \Delta} \qquad \frac{\Gamma \vdash u : A \mid \Delta \qquad \Gamma \mid e : B \vdash \Delta}{\Gamma \mid u \cdot e : A \to B \vdash \Delta} \qquad \frac{c : (\Gamma, x : A \vdash \Delta)}{\Gamma \mid \tilde{\mu}x.c : A \vdash \Delta}$$

"Why should I care?"

"Why should I care?"

Because sequent calculus is well-behaved! 🙂

Sequent calculus as IR

You just defined a wonderful calculus, and you are wondering:

Problem

How to define a continuation-passing style translation?

CPS translation:
$[\![\cdot]\!] : \mathit{source} \longrightarrow \lambda^{\text{something}}$

- preserving reduction
- preserving typing
- the type $[\![\bot]\!]$ is not inhabited

Typically: $[\![V]\!] \triangleq \lambda k.k\,V$
$[\![t]\!] \triangleq \lambda k.?$

Benefits:
If $\lambda^{\text{something}}$ is sound and normalizing:

1. If $[\![t]\!]$ normalizes, then t normalizes
2. If t is typed, then t normalizes
3. There is no term $\vdash t : \bot$

Sequent calculus as IR

You just defined a wonderful calculus, and you are wondering:

Problem

How to define a continuation-passing style translation?

Solution

Use sequent calculus!

Slogan:

A sequent calculus is a defunctionalization of CPS representations.

↬ as such it defines a good intermediate representation for compilation

Method: Danvy's semantics artifacts

Sequent calculus as IR

You just defined a wonderful calculus, and you are wondering:

Problem

How to define a continuation-passing style translation?

Solution

Use sequent calculus!

Slogan:

A sequent calculus is a defunctionalization of CPS representations.

↬ as such it defines a good intermediate representation for compilation

Method: Danvy's semantics artifacts

Semantics artifacts in action

Call-by-name $\lambda\mu\tilde{\mu}$-calculus:

$$\begin{array}{llll} \text{Terms} & t ::= V \mid \mu\alpha.c & \text{Contexts} & e ::= E \mid \tilde{\mu}x.c \\ \text{Values} & V ::= x \mid \lambda x.t & \text{Co-values} & E ::= \alpha \mid t \cdot e \end{array}$$

$$\text{Commands} \quad c ::= \langle t \parallel e \rangle$$

Reduction rules:

$$\begin{array}{ccc} \langle t \parallel \tilde{\mu}x.c \rangle & \rightarrow & c[t/x] \\ \langle \mu\alpha.c \parallel E \rangle & \rightarrow & c[E/\alpha] \\ \langle \lambda x.t \parallel u \cdot e \rangle & \rightarrow & \big\langle u \,\big\|\, \tilde{\mu}x.\langle t \parallel e \rangle \big\rangle \end{array}$$

Semantics artifacts in action

$$
\begin{array}{llll}
\text{Terms} & t ::= V \mid \mu\alpha.c & \text{Contexts} & e ::= E \mid \tilde{\mu}x.c \\
\text{Values} & V ::= x \mid \lambda x.t & \text{Co-values} & E ::= \alpha \mid t \cdot e
\end{array}
$$

$$\text{Commands} \quad c ::= \langle t \,\|\, e \rangle$$

Small steps

$$
\begin{array}{lrcl}
 & \langle t \,\|\, \tilde{\mu}x.c \rangle_e & \rightsquigarrow & c_e[t/x] \\
e & \langle t \,\|\, E \rangle_e & \rightsquigarrow & \langle t \,\|\, E \rangle_t \\
 & \langle \mu\alpha.c \,\|\, E \rangle_t & \rightsquigarrow & c_e[E/\alpha] \\
t & \langle V \,\|\, E \rangle_t & \rightsquigarrow & \langle V \,\|\, E \rangle_E \\
E & \langle V \,\|\, u \cdot e \rangle_E & \rightsquigarrow & \langle V \,\|\, u \cdot e \rangle_V \\
V & \langle \lambda x.t \,\|\, u \cdot e \rangle_V & \rightsquigarrow & \big\langle u \,\big\|\, \tilde{\mu}x.\langle t \,\|\, e \rangle \big\rangle_e
\end{array}
$$

Semantics artifacts in action

Terms	$t ::= V \mid \mu\alpha.c$	Contexts	$e ::= E \mid \tilde{\mu}x.c$
Values	$V ::= x \mid \lambda x.t$	Co-values	$E ::= \alpha \mid t \cdot e$

Commands $c ::= \langle t \parallel e \rangle$

Small steps **CPS**

	Small steps			CPS
e	$\langle t \parallel \tilde{\mu}x.c\rangle_e$	$\rightsquigarrow$	$c_e[t/x]$	$[\![\tilde{\mu}x.c]\!]_e\, t \triangleq (\lambda x.[\![c]\!]_c)\, t$
	$\langle t \parallel E\rangle_e$	$\rightsquigarrow$	$\langle t \parallel E\rangle_t$	$[\![E]\!]_e\, t \triangleq t\, [\![E]\!]_E$
t	$\langle \mu\alpha.c \parallel E\rangle_t$	$\rightsquigarrow$	$c_e[E/\alpha]$	$[\![\mu\alpha.c]\!]_t\, E \triangleq (\lambda\alpha.[\![c]\!]_c)\, E$
	$\langle V \parallel E\rangle_t$	$\rightsquigarrow$	$\langle V \parallel E\rangle_E$	$[\![V]\!]_t\, E \triangleq E\, [\![V]\!]_V$
E	$\langle V \parallel u \cdot e\rangle_E$	$\rightsquigarrow$	$\langle V \parallel u \cdot e\rangle_V$	$[\![u \cdot e]\!]_E\, V \triangleq V\, [\![u]\!]_t\, [\![e]\!]_e$
V	$\langle \lambda x.t \parallel u \cdot e\rangle_V$	$\rightsquigarrow$	$\big\langle u \,\big\|\, \tilde{\mu}x.\langle t \parallel e\rangle\big\rangle_e$	$[\![\lambda x.t]\!]_V\, u\, e \triangleq (\lambda x.e\, [\![t]\!]_t)\, u$

Preservation

$$c \overset{1}{\rightsquigarrow} c' \quad \Rightarrow \quad [\![c]\!]_c \xrightarrow{+}_\beta [\![c']\!]_c$$

Semantics artifacts in action

Terms $t ::= V \mid \mu\alpha.c$ Contexts $e ::= E \mid \tilde{\mu}x.c$

Values $V ::= x \mid \lambda x.t$ Co-values $E ::= \alpha \mid t \cdot e$

Commands $c ::= \langle t \| e \rangle$

CPS

e:

$$[\![\tilde{\mu}x.c]\!]_e\, t \triangleq (\lambda x.[\![c]\!]_c)\, t$$

$$[\![E]\!]_e\, t \triangleq t\, [\![E]\!]_E$$

t:

$$[\![\mu\alpha.c]\!]_t\, E \triangleq (\lambda\alpha.[\![c]\!]_c)\, E$$

$$[\![V]\!]_t\, E \triangleq E\, [\![V]\!]_V$$

E:

$$[\![u \cdot e]\!]_E\, V \triangleq V\, [\![u]\!]_t\, [\![e]\!]_e$$

V:

$$[\![\lambda x.t]\!]_V\, u\, e \triangleq (\lambda x.e\, [\![t]\!]_t)\, u$$

Types translation

$$[\![A]\!]_e \triangleq [\![A]\!]_t \to \bot$$

$$[\![A]\!]_t \triangleq [\![A]\!]_E \to \bot$$

$$[\![A]\!]_E \triangleq [\![A]\!]_V \to \bot$$

$$[\![A \to B]\!]_V \triangleq [\![A]\!]_t \to [\![B]\!]_e \to \bot$$

Preservation

$$\Gamma \vdash t : A \mid \Delta \qquad \Rightarrow \qquad [\![\Gamma]\!]_t, [\![\Delta]\!]_E \vdash [\![t]\!]_t : [\![A]\!]_t$$

Semantics artifacts in action

Normalization

Typed commands of the call-by-name $\lambda\mu\tilde{\mu}$-calculus normalize.

Inhabitation

There is no simply-typed λ-term t such that $\vdash t : [\![\bot]\!]_t$.

Soundness

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

Normalization proofs

You just defined a wonderful calculus, but the CPS method is too complex:

Problem

How can I prove normalization?

Solution

Use sequent calculus + Krivine realizability!

Slogan:

A sequent calculus specifies the interactions of terms and contexts.

↬ as you will see, this helps a lot the definition of a realizability interpretation

Method: Danvy's semantics artifacts, again

Normalization proofs

You just defined a wonderful calculus, but the CPS method is too complex:

Problem

How can I prove normalization?

Solution

Use sequent calculus + Krivine realizability!

Slogan:

A sequent calculus specifies the interactions of terms and contexts.

$\looparrowright$ as you will see, this helps a lot the definition of a realizability interpretation

Method: Danvy's semantics artifacts, again

Normalization proofs

You just defined a wonderful calculus, but the CPS method is too complex:

Problem

How can I prove normalization?

Solution

Use sequent calculus + Krivine realizability!

Slogan:

A sequent calculus specifies the interactions of terms and contexts.

$\looparrowright$ as you will see, this helps a lot the definition of a realizability interpretation

Method: Danvy's semantics artifacts, again

Realizability *à la* Krivine

(think of some kind of unary logical relation)

Intuition

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

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

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

Truth value defined by **orthogonality** :

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

Realizability *à la* Krivine

(think of some kind of unary logical relation)

Intuition

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

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

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

Truth value defined by **orthogonality** :

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

Realizability *à la* Krivine

(think of some kind of unary logical relation)

Intuition

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

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

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

Truth value defined by **orthogonality** :

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

Semantic artifacts, bis

Terms	$t ::= V \mid \mu\alpha.c$	Contexts	$e ::= E \mid \tilde{\mu}x.c$
Values	$V ::= x \mid \lambda x.t$	Co-values	$E ::= \alpha \mid t \cdot e$

Commands $c ::= \langle t \,\|\, e \rangle$

Small steps

$$
\begin{array}{lrcl}
e & \langle t \,\|\, \tilde{\mu}x.c \rangle_e & \rightsquigarrow & c_e[t/x] \\
  & \langle t \,\|\, E \rangle_e & \rightsquigarrow & \langle t \,\|\, E \rangle_t \\
t & \langle \mu\alpha.c \,\|\, E \rangle_t & \rightsquigarrow & c_e[E/\alpha] \\
  & \langle V \,\|\, E \rangle_t & \rightsquigarrow & \langle V \,\|\, E \rangle_E \\
E & \langle V \,\|\, u \cdot e \rangle_E & \rightsquigarrow & \langle V \,\|\, u \cdot e \rangle_V \\
V & \langle \lambda x.t \,\|\, u \cdot e \rangle_V & \rightsquigarrow & \big\langle u \,\big\|\, \tilde{\mu}x.\langle t \,\|\, e \rangle \big\rangle_e
\end{array}
$$

Semantic artifacts, bis

Terms $t ::= V \mid \mu\alpha.c$

Values $V ::= x \mid \lambda x.t$

Contexts $e ::= E \mid \tilde{\mu}x.c$

Co-values $E ::= \alpha \mid t \cdot e$

Commands $c ::= \langle t \,\|\, e \rangle$

Small steps

$$
\begin{array}{llcl}
e & \langle t \,\|\, \tilde{\mu}x.c\rangle_e & \rightsquigarrow & c_e[t/x] \\
 & \langle t \,\|\, E\rangle_e & \rightsquigarrow & \langle t \,\|\, E\rangle_t \\
t & \langle \mu\alpha.c \,\|\, E\rangle_t & \rightsquigarrow & c_e[E/\alpha] \\
 & \langle V \,\|\, E\rangle_t & \rightsquigarrow & \langle V \,\|\, E\rangle_E \\
E & \langle V \,\|\, u \cdot e\rangle_E & \rightsquigarrow & \langle V \,\|\, u \cdot e\rangle_V \\
V & \langle \lambda x.t \,\|\, u \cdot e\rangle_V & \rightsquigarrow & \big\langle u \,\big\|\, \tilde{\mu}x.\langle t \,\|\, e\rangle \big\rangle_e
\end{array}
$$

Realizability

$$\|A\|_e \triangleq |A|_t{}^{\perp\!\!\!\perp}$$

$$|A|_t \triangleq \|A\|_E{}^{\perp\!\!\!\perp}$$

$$\|A \to B\|_E \triangleq \{u \cdot e : \quad u \in |A|_t \;\wedge\; e \in \|B\|_e\}$$

Adequacy

1. $\cdot \vdash t : A \mid \cdot \quad \Rightarrow \quad t \in |A|_t$
2. $\cdot \mid e : A \vdash \cdot \quad \Rightarrow \quad e \in \|A\|_e$
3. $c : (\cdot \vdash \cdot) \quad \Rightarrow \quad c \in \perp\!\!\!\perp$

Consequences

Normalizing commands

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

Proof. If $c \to c'$ and c' normalizes, so does c. □

Normalization

For any command c, if $c : \Gamma \vdash \Delta$, then c normalizes.

Proof. By adequacy, any typed command c belongs to the pole $\perp\!\!\!\perp_{\Downarrow}$. □

Soundness

There is no proof t such that $\vdash t : \perp \mid$.

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

Polarized sequent calculus

You added sums to your favorite λ-calculus, it broke all your proofs:

Problem

What can I do?

Solution

Use sequent calculus + polarities!

Negative polarity	Every expression is a value (CBN)
Positive polarity	Every context is a covalue (CBV)

Slogan:

Polarized $\lambda\mu\tilde{\mu}$ is a good, regular syntax for programs.

$\hookrightarrow$ a.k.a. system L, a great syntax for call-by-push-value

Method: see Munch-Maccagnoni & Scherer's paper (LICS'15)

Polarized sequent calculus

You added sums to your favorite λ-calculus, it broke all your proofs:

Problem

What can I do?

Solution

Use sequent calculus + polarities!

Negative polarity	Every expression is a value (CBN)
Positive polarity	Every context is a covalue (CBV)

Slogan:

Polarized $\lambda\mu\tilde{\mu}$ is a good, regular syntax for programs.

↬ a.k.a. system L, a great syntax for call-by-push-value

Method: see Munch-Maccagnoni & Scherer's paper (LICS'15)

Polarized sequent calculus

You added sums to your favorite λ-calculus, it broke all your proofs:

Problem

What can I do?

Solution

Use sequent calculus + polarities!

Negative polarity	Every expression is a value (CBN)
Positive polarity	Every context is a covalue (CBV)

Slogan:

Polarized $\lambda\mu\tilde{\mu}$ is a good, regular syntax for programs.

↬ a.k.a. system L, a great syntax for call-by-push-value

Method: see Munch-Maccagnoni & Scherer's paper (LICS'15)

Take away

Sequent calculus:

- is more *regular* than natural deduction
- corresponds to *abstract-machine-like* calculi (*e.g.* $\lambda\mu\tilde{\mu}$-calculus)
- provides great insights on *operational semantics*

A **flexible** tool:

- can be decomposed with connectives of *linear logic*
- can be *polarized* (Munch-Maccagnoni's system L)
- supports *effectful* constructors
- ...

If you don't use it already,

What are you waiting for?

Take away

Sequent calculus:

- is more *regular* than natural deduction
- corresponds to *abstract-machine-like* calculi (*e.g.* $\lambda\mu\tilde{\mu}$-calculus)
- provides great insights on *operational semantics*

A **flexible** tool:

- can be decomposed with connectives of *linear logic*
- can be *polarized* (Munch-Maccagnoni's system L)
- supports *effectful* constructors
- ...

If you don't use it already,

What are you waiting for?

Take away

Sequent calculus:

- is more *regular* than natural deduction
- corresponds to *abstract-machine-like* calculi (*e.g.* $\lambda\mu\tilde{\mu}$-calculus)
- provides great insights on *operational semantics*

A **flexible** tool:

- can be decomposed with connectives of *linear logic*
- can be *polarized* (Munch-Maccagnoni's system L)
- supports *effectful* constructors
- ...

If you don't use it already,

What are you waiting for?

SELECTED REFERENCES

On **sequent calculus/proofs-as-programs**:

- *The Duality of Computation.* Curien & Herbelin (2000)
- *A Tutorial on Computational Classical Logic and the Sequent Calculus* Downen & Ariola (2018)
- *LKQ and LKT: Sequent calculi for second order logic ...* Danos, Joinet & Schellinx (1995)

On **polarized sequent calculi**:

- *Polarised Intermediate Representation of Lambda Calculus with Sums* Munch-Maccagnoni & Scherer (2015)
- *Syntax and Models of a Non-Associative Composition of Programs and Proofs* Munch-Maccagnoni (2013)
- *A Preview of a Tutorial on Polarised L Calculi* Maillard, M., Montillet, Munch-Maccagnoni, Scherer (2018)

Use cases of **Danvy's semantics artifacts**:

- *A Constructive Proof of Dependent Choice in Classical Arithmetic via Memoization.* M. (2019)