A sequent calculus with dependent types for classical arithmetic

Étienne MIQUEY

Équipe Gallinette, INRIA
LS2N, Université de Nantes

Workshop Réalisabilité
13 Juin 2018
A constructive proof of dependent choice compatible with classical logic
# The Curry-Howard correspondence

<table>
<thead>
<tr>
<th>Mathematics</th>
<th>Computer Science</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proofs</td>
<td>Programs</td>
</tr>
<tr>
<td>Propositions</td>
<td>Types</td>
</tr>
<tr>
<td>Deduction rules</td>
<td>Typing rules</td>
</tr>
</tbody>
</table>

\[
\frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B} \quad (\Rightarrow_E)
\]

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

**Benefits:**

- *Program your proofs!*
- *Prove your programs!*
# Proofs-as-programs

## Limitations

<table>
<thead>
<tr>
<th>Mathematics</th>
<th>Computer Science</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A \lor \neg A$</td>
<td>try... catch...</td>
</tr>
<tr>
<td>$\neg\neg A \Rightarrow A$</td>
<td>x := 42</td>
</tr>
<tr>
<td>All sets can be well-ordered</td>
<td>random()</td>
</tr>
<tr>
<td>Sets that have the same elements are equal</td>
<td>stop</td>
</tr>
<tr>
<td></td>
<td>goto</td>
</tr>
</tbody>
</table>

↬ *We want more!*
Extending Curry-Howard

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

1990: Griffin discovered that call/cc can be typed by Peirce’s law
(well-known fact: Peirce’s law $\Rightarrow A \lor \neg A$)

Classical Curry-Howard:

$\lambda$-calculus + call/cc

Other examples:
- quote instruction $\sim$ dependent choice
- monotonic memory $\sim$ Cohen’s forcing
- ...

The motto

*With side-effects come new reasoning principles.*
Extending Curry-Howard

Classical logic \(=\) Intuitionistic logic + \(A \lor \neg A\)

1990: Griffin discovered that call/cc can be typed by Peirce’s law
(well-known fact: Peirce’s law \(\Rightarrow A \lor \neg A\))

Classical Curry-Howard:

\[\lambda\text{-calculus } + \text{ call/cc}\]

Other examples:
- quote instruction \(\sim\) dependent choice
- monotonic memory \(\sim\) Cohen’s forcing
- ...

The motto

*With side-effects come new reasoning principles.*
**Extending Curry-Howard**

**Classical logic** = **Intuitionistic logic** + $A \lor \neg A$

**1990**: Griffin discovered that call/cc can be typed by Peirce’s law
(well-known fact: Peirce’s law $\Rightarrow A \lor \neg A$)

**Classical Curry-Howard**:

$\lambda$-calculus + call/cc

Other examples:
- quote instruction $\sim$ dependent choice
- monotonic memory $\sim$ Cohen’s forcing
- ...

**The motto**

*With side-effects come new reasoning principles.*
Teaser

The motto

With side-effects come new reasoning principles.

We will use several computational features:

- dependent types
- lazy evaluation
- streams
- shared memory

...to get a proof for the axioms of dependent and countable choice that is compatible with classical logic.
The axiom of choice

Axiom of Choice:

\[ AC : \forall x^A. \exists y^B. P(x, y) \rightarrow \exists f^{A \rightarrow B}. \forall x^A. P(x, f(x)) \]
The axiom of choice

Axiom of Choice:

\[
AC : \forall x^A. \exists y^B.P(x, y) \rightarrow \exists f^{A\rightarrow B}. \forall x^A.P(x, f(x)) \\
:= \lambda H. (\lambda x. \text{wit}(H \, x), \lambda x. \text{prf}(H \, x))
\]

Computational content through dependent types:

\[
\frac{\Gamma, x : T \vdash t : A}{\Gamma \vdash \lambda x. t : \forall x^T.A} \quad (\forall_I) \quad \frac{\Gamma \vdash p : A[t/x]}{\Gamma \vdash t : T} \quad (\exists_I) \\
\frac{\Gamma \vdash p : \exists x^T.A(x) \quad \Gamma \vdash t : T}{\Gamma \vdash (t, p) : \exists x^T.A} \quad (\exists_I) \quad \frac{\Gamma \vdash p : \exists x^T.A(x)}{\Gamma \vdash \text{wit} \, p : T} \quad (\text{wit}) \\
\frac{\Gamma \vdash p : \exists x^T.A(x)}{\Gamma \vdash \text{prf} \, p : A(\text{wit} \, p)} \quad (\text{prf})
\]
Incompatibility with classical logic

Bad news

dependent sum + classical logic = ☠

Choice:

\[ \vdash t : \forall x \in A. \exists y \in B. P(x, y) \rightarrow \exists f \in B^A. \forall x \in A. P(x, f(x)) \]

Excluded-middle:

\[ \vdash s : \forall x \in X. \exists y \in \{0, 1\}. (U(x) \land y = 1) \lor (\neg U(x) \land y = 0) \]

Take \( U \) undecidable:

\[ \vdash t \; s : \exists f \in \{0, 1\}^X. \forall x \in X. (U(x) \land f(x) = 1) \lor (\neg U(x) \land f(x) = 0) \]

\[ \dashv i.e. \; \text{wit}(t \; s) \; \text{computes the uncomputable...} \]
Incompatibility with classical logic

Bad news

dependent sum + classical logic = ☠

Choice:

\[
\vdash t : \forall x \in A. \exists y \in B. P(x, y) \rightarrow \exists f \in B^A. \forall x \in A. P(x, f(x))
\]

Excluded-middle:

\[
\vdash s : \forall x \in X. \exists y \in \{0, 1\}. (U(x) \land y = 1) \lor (\neg U(x) \land y = 0)
\]

Take $U$ undecidable:

\[
\vdash t \cdot s : \exists f \in \{0, 1\}^X. \forall x \in X. (U(x) \land f(x) = 1) \lor (\neg U(x) \land f(x) = 0)
\]

⇒ i.e. $\text{wit}(t \cdot s)$ computes the uncomputable...
Incompatibility with classical logic

Bad news

dependent sum + classical logic = 

One can define:

\[ H_0 := \text{call/cc}_\alpha (1, \text{throw}_\alpha (0, p)) : \exists x. x = 0 \]

and reach a contradiction:

\[ (\text{wit } H_0, \text{prf } H_0) \rightarrow (1, \underbrace{\begin{array}{c} 0 = 0 \\ p \\ \exists x. x = 0 \end{array}}_{\text{p}}) \]

We need to:

\[ \rightarrow \text{ share } \quad \rightarrow \text{ restrict dependent types} \]
Incompatibility with classical logic

Bad news

dependent sum + classical logic = ☠

One can define:

\[ H_0 := \text{call/cc}_\alpha(1, \text{throw}_\alpha(0,p)) : \exists x. x = 0 \]

and reach a contradiction:

\[ (\text{wit } H_0, \text{prf } H_0) \rightarrow \begin{cases} 0=0 \\ 1, p \\ \exists x. x=0 \end{cases} \]

We need to:

\[ \mapsto \text{share} \quad \mapsto \text{restrict} \text{ dependent types} \]
Toward a solution?

- Restriction to countable choice:

\[ AC_N : \forall x^\mathbb{N}. \exists y^B. P(x, y) \rightarrow \exists f^{\mathbb{N}\rightarrow B}. \forall x^\mathbb{N}. P(x, f(x)) \]

- Proof:

\[ AC := \lambda H.(\lambda n.\text{if } n = 0 \text{ then } \text{wit}(H \ 0) \text{ else } \text{if } n = 1 \text{ then } \text{wit}(H \ 1) \text{ else } \ldots , \\\n\quad \lambda n.\text{if } n = 0 \text{ then } \text{prf}(H \ 0) \text{ else } \text{if } n = 1 \text{ then } \text{prf}(H \ 1) \text{ else } \ldots ) \]
Toward a solution?

- Restriction to countable choice:

\[ AC_N : \forall x^\mathbb{N}. \exists y^B.P(x,y) \rightarrow \exists f^{\mathbb{N}\rightarrow B}. \forall x^\mathbb{N}. P(x,f(x)) \]

- Proof:

\[ AC_N := \lambda H. \text{let } H_0 = H \ 0 \ \text{in} \]
\[ \text{let } H_1 = H \ 1 \ \text{in} \]
\[ ... \]
\[ (\lambda n. \text{if } n = 0 \ \text{then} \ \text{wit } H_0 \ \text{else} \]
\[ \text{if } n = 1 \ \text{then} \ \text{wit } H_1 \ \text{else} ... , \]
\[ \lambda n. \text{if } n = 0 \ \text{then} \ \text{prf } H_0 \ \text{else} \]
\[ \text{if } n = 1 \ \text{then} \ \text{prf } H_1 \ \text{else} ... ) \]
Toward a solution?

- Restriction to countable choice:

\[ AC_N : \forall x^\mathbb{N}.\exists y^B.P(x, y) \rightarrow \exists f^{\mathbb{N}\rightarrow B}.\forall x^\mathbb{N}.P(x, f(x)) \]

- Proof:

\[ AC_N := \lambda H. \text{let } H_\infty = (H_0, H_1, \ldots, H_n, \ldots) \text{ in } \]
\[ (\lambda n. \text{wit}(\text{nth } n \ H_\infty), \lambda n. \text{prf}(\text{nth } n \ H_\infty)) \]
Toward a solution?

- **Restriction to countable choice:**

\[
AC_N : \forall x^\mathbb{N}. \exists y^B. P(x, y) \rightarrow \exists f^{\mathbb{N} \rightarrow B}. \forall x^\mathbb{N}. P(x, f(x))
\]

- **Proof:**

\[
AC_N := \lambda H. \text{let } H_\infty = \text{cofix}^0_{bn}(H \ n, \ b(S(n))) \text{ in } \\
(\lambda n. \text{wit}(\text{nth} \ n \ H_\infty), \lambda n. \text{prf}(\text{nth} \ n \ H_\infty))
\]
dPA^ω (Herbelin’s recipe)

A proof system:
- **classical:**
  \[ p, q ::= ... | \text{catch}_\alpha p | \text{throw}_\alpha p \]

- with stratified **dependent types**:
  - terms: \[ t, u ::= ... | \text{wit} p \]
  - formulas: \[ A, B ::= ... | \forall x^T A | \exists x^T A | \Pi_{(a:A)} B | t = u \]
  - proofs: \[ p, q ::= ... | \lambda x. p | (t, p) | \lambda a. p \]

- **a syntactical restriction** of dependencies to NNF proofs
- **call-by-value and sharing:**
  \[ p, q ::= ... | \text{let} a = q \text{ in } p \]

- with inductive and **coinductive** constructions:
  \[ p, q ::= ... | \text{fix}^{t}_{bn} [p_0 | p_S] | \text{cofix}^{t}_{bn} p \]

- **lazy evaluation** for the cofix
dPA\(^{\omega}\) (Herbelin’s recipe)

A proof system:

- **classical**:

  \[ p, q ::= \ldots \mid \text{catch}_{\alpha} p \mid \text{throw}_{\alpha} p \]

- with stratified **dependent types**:
  
  - terms: \( t, u ::= \ldots \mid \text{wit} \, p \)
  
  - formulas: \( A, B ::= \ldots \mid \forall x^T.A \mid \exists x^T.A \mid \Pi_{(a:A)}.B \mid t = u \)
  
  - proofs: \( p, q ::= \ldots \mid \lambda x. p \mid (t, p) \mid \lambda a. p \)

- a **syntactical restriction** of dependencies to NEF proofs
- **call-by-value** and **sharing**:

  \[ p, q ::= \ldots \mid \text{let} \, a = q \, \text{in} \, p \]

- **with inductive and coinductive constructions**:

  \[ p, q ::= \ldots \mid \text{fix}^t_{bn}[p_0 | p_S] \mid \text{cofix}^t_{bn} \, p \]

- **lazy evaluation** for the cofix
State of the art

Subject reduction

If $\Gamma \vdash p : A$ and $p \rightarrow q$, then $\Gamma \vdash q : A$.

Normalization

If $\Gamma \vdash p : A$ then $p$ is normalizable.

Consistency

$\not\vdash_{dPA^\omega} \bot$
Roadmap

\[ \text{dPA}^\omega [\text{Herbelin'12}]: \]
- control operators
- dependent types
- co-fixpoints
- sharing & laziness

Subject reduction

CPS-translation?

?-calculus

Normalization
Remark: CPS usually factorize through sequent calculi!

Roadmap
Constructive proof of DC
Semantic artifacts
Classical call-by-need
dL
Roadmap

A sequent calculus with dependent types for classical arithmetic
Roadmap

$\lambda_{[l\nu\tau\star]}$

$\dPA^\omega$ [Herbelin’12]:
- control operators
- dependent types
- co-fixpoints
- sharing & laziness

Subject reduction

typing/reduction preservation

$\dLPA^\omega$?

- sequent calculus
- dependent types
- co-fixpoints
- sharing & laziness

Subject reduction

CPS-translation?

?-calculus

Normalization
Roadmap

\begin{itemize}
\item dPA^\omega [Herbelin’12]:
  \begin{itemize}
  \item control operators
  \item dependent types
  \item co-fixpoints
  \item sharing & laziness
  \end{itemize}
  \end{itemize}

\begin{itemize}
\item dLPA^\omega?
  \begin{itemize}
  \item sequent calculus
  \item dependent types
  \item co-fixpoints
  \item sharing & laziness
  \end{itemize}
  \end{itemize}

\begin{itemize}
\item \(?\)-calculus
\end{itemize}
Danvy’s semantic artifacts
CPS translation

Continuation-passing style translation: \([\cdot] : \text{source} \rightarrow \lambda^{\text{machin}}\)
- preserving reduction
  \[ t \xrightarrow{1} t' \quad \Rightarrow \quad [t] \xrightarrow{\dagger} [t'] \]
- preserving typing
  \[ \Gamma \vdash t : A \quad \Rightarrow \quad [\Gamma] \vdash [t] : [A] \]
- the type \([\bot]\) is not inhabited

Benefits

If \(\lambda^{\text{machin}}\) is sound and normalizing:
1. If \([t]\) normalizes, then \(t\) normalizes
2. If \(t\) is typed, then \(t\) normalizes
3. The source language is sound, \(i.e.\) there is no term \(\vdash t : \bot\)
CPS translation

Continuation-passing style translation: 

\[
\llbracket \cdot \rrbracket : \text{source} \rightarrow \lambda^{\text{machin}}
\]

- preserving reduction
- preserving typing
- the type \([\bot]\) is not inhabited

Benefits

If \(\lambda^{\text{machin}}\) is sound and normalizing:

1. If \(\llbracket t \rrbracket\) normalizes, then \(t\) normalizes
2. If \(t\) is typed, then \(t\) normalizes
3. The source language is sound, i.e. there is no term \(\vdash t : \bot\)

Danvy’s methodology

1. an operational semantics
2. a small-step calculus or abstract machine
3. a continuation-passing style translation
4. a realizability model

Defunctionalized Interpreters for Call-by-Need Evaluation
Danvy et al. (2010)
The $\lambda\mu\tilde{\mu}$-calculus

Syntax:

(Proofs) \[ p ::= a | \lambda a.p | \mu\alpha.c \]

(Contexts) \[ e ::= \alpha | p \cdot e | \tilde{\mu}\alpha.c \]

(Commands) \[ c ::= \langle p \parallel e \rangle \]

Typing rules:

\[
\begin{array}{c}
\Gamma \vdash t : A | \Delta \quad \Gamma \mid e : A \vdash \Delta \\
\hline
\langle t \parallel e \rangle : (\Gamma \vdash \Delta)
\end{array}
\]

\[
\begin{array}{c}
(a : A) \in \Gamma \\
\hline
\Gamma \vdash a : A \mid \Delta
\end{array}
\]

\[
\begin{array}{c}
\Gamma, a : A \vdash p : B \mid \Delta \\
\hline
\Gamma \vdash \lambda a.p : A \rightarrow B \mid \Delta
\end{array}
\]

\[
\begin{array}{c}
(\alpha : A) \in \Delta \\
\hline
\Gamma \mid \alpha : A \vdash \Delta
\end{array}
\]

\[
\begin{array}{c}
\Gamma \vdash p : A \mid \Delta \quad \Gamma \mid e : B \vdash \Delta \\
\hline
\Gamma \mid p \cdot e : A \rightarrow B \vdash \Delta
\end{array}
\]

\[
\begin{array}{c}
\Gamma \vdash \mu\alpha.c : A \mid \Delta \\
\hline
\Gamma \mid \tilde{\mu}\alpha.c : A \vdash \Delta
\end{array}
\]

The duality of computation
Curien/Herbelin (2000)
The $\lambda\mu\tilde{\mu}$-calculus

Syntax:

(Proofs) \[ p ::= a \mid \lambda a.p \mid \mu\alpha.c \]

(Contexts) \[ e ::= \alpha \mid p \cdot e \mid \tilde{\mu}a.c \]

(Commands) \[ c ::= \langle p \parallel e \rangle \]

Typing rules:

\[
\frac{\Gamma \vdash A \mid \Delta \quad \Gamma \mid A \vdash \Delta}{\Gamma \vdash \Delta} \quad (\Gamma \vdash \Delta)
\]

\[
\frac{A \in \Gamma}{\Gamma \vdash A \mid \Delta}
\]

\[
\frac{\Gamma, A \vdash B \mid \Delta}{\Gamma \vdash A \rightarrow B \mid \Delta}
\]

\[
\frac{\Gamma \vdash \Delta, \ A}{\Gamma \vdash A \mid \Delta}
\]

\[
\frac{A \in \Delta}{\Gamma \mid A \vdash \Delta}
\]

\[
\frac{\Gamma \vdash A \mid \Delta \quad \Gamma \mid B \vdash \Delta}{\Gamma \mid A \rightarrow B \vdash \Delta}
\]

\[
\frac{\Gamma, \ A \vdash \Delta}{\Gamma \mid A \vdash \Delta}
\]

\[
\frac{\Gamma \vdash \Delta, \ A}{\Gamma \vdash A \mid \Delta}
\]

The duality of computation by Curien/Herbelin (2000)
The $\lambda\mu\tilde{\mu}$-calculus

Syntax:

(Proofs) \[ p ::= a \mid \lambda a. p \mid \mu \alpha. c \]

(Contexts) \[ e ::= \alpha \mid p \cdot e \mid \tilde{\mu} \alpha. c \]

(Commands) \[ c ::= \langle p \parallel e \rangle \]

Typing rules:

\[
\frac{\Gamma \vdash t : A \mid \Delta \quad \Gamma \mid e : A \vdash \Delta}{\langle t \parallel e \rangle : (\Gamma \vdash \Delta)}
\]

\[
\frac{(a : A) \in \Gamma}{\Gamma \vdash a : A \mid \Delta}
\]

\[
\frac{\Gamma, a : A \vdash p : B \mid \Delta}{\Gamma \vdash \lambda a. p : A \to B \mid \Delta}
\]

\[
\frac{c : (\Gamma \vdash \Delta, \alpha : A)}{\Gamma \vdash \mu \alpha. c : A \mid \Delta}
\]

\[
\frac{(\alpha : A) \in \Delta}{\Gamma \mid \alpha : A \vdash \Delta}
\]

\[
\frac{\Gamma \vdash p : A \mid \Delta \quad \Gamma \mid e : B \vdash \Delta}{\Gamma \mid p \cdot e : A \to B \vdash \Delta}
\]

\[
\frac{c : (\Gamma, a : A \vdash \Delta)}{\Gamma \mid \tilde{\mu} a. c : A \vdash \Delta}
\]
The λμ��-calculus

Syntax:

(Proofs) \( p ::= a \mid \lambda a.p \mid \mu \alpha.c \)

(Contexts) \( e ::= \alpha \mid p \cdot e \mid \tilde{\mu}a.c \)

(Commands) \( c ::= \langle p \parallel e \rangle \)

Reduction:

\[ \langle \lambda a.p \parallel q \cdot e \rangle \rightarrow \langle q \parallel \tilde{\mu}a.\langle p \parallel e \rangle \rangle \]

\[ \langle p \parallel \tilde{\mu}a.c \rangle \rightarrow c[p/a] \quad p \in \mathcal{P} \]

\[ \langle \mu \alpha.c \parallel e \rangle \rightarrow c[e/\alpha] \quad e \in \mathcal{E} \]

Critical pair:

\[ \langle \mu \alpha.c \parallel \tilde{\mu}a.c' \rangle \]

\[ c[\tilde{\mu}a.c'/\alpha] \]

\[ c'[\mu \alpha.c/a] \]
### The $\lambda\mu\tilde{\mu}$-calculus

**Syntax:**

<table>
<thead>
<tr>
<th>Proofs</th>
<th>$p ::= V \mid \mu \alpha. c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contexts</td>
<td>$e ::= E \mid \tilde{\mu} a \cdot c$</td>
</tr>
<tr>
<td>Commands</td>
<td>$c ::= \langle p \parallel e \rangle$</td>
</tr>
</tbody>
</table>

**Values**

$V ::= a \mid \lambda a. p$

**Co-values**

$E ::= \alpha \mid p \cdot e$

**Reduction:**

\[
\langle \lambda a. p \parallel q \cdot e \rangle \rightarrow \langle q \parallel \tilde{\mu} a. \langle p \parallel e \rangle \rangle
\]

\[
\langle p \parallel \tilde{\mu} a. c \rangle \rightarrow c[p/a]
\]

\[
\langle \mu \alpha. c \parallel e \rangle \rightarrow c[e/\alpha]
\]

**Critical pair:**

\[
\langle \mu \alpha. c \parallel \tilde{\mu} a. c' \rangle
\]

- CbV: $c[\tilde{\mu} a. c'/\alpha]$
- CbN: $c'[\mu \alpha. c/a]$
Call-by-name $\lambda\mu\tilde{\mu}$-calculus

Syntax:

(Proofs) $p ::= V \mid \mu\alpha.c$

(Values) $V ::= a \mid \lambda a.p$

(Contexts) $e ::= E \mid \tilde{\mu}a.c$

(Co-values) $E ::= \alpha \mid p \cdot e$

(Command) $c ::= \langle p \parallel e \rangle$

Reduction rules:

$\langle p \parallel \tilde{\mu}a.c \rangle \rightarrow c[p/a]$

$\langle \mu\alpha.c \parallel E \rangle \rightarrow c[E/\alpha]$

$\langle \lambda a.p \parallel q \cdot e \rangle \rightarrow \langle q \parallel \tilde{\mu}a.(p \parallel e) \rangle$
Semantic artifacts

(Proofs) \( p ::= V \mid \mu \alpha . c \)  
(Values) \( V ::= a \mid \lambda a . p \)  
(Contexts) \( e ::= E \mid \tilde{\mu} a . c \)  
(Co-values) \( E ::= \alpha \mid p \cdot e \)  

(Command) \( c ::= \langle p \parallel e \rangle \)  

Small steps

\[
\begin{align*}
\langle p \parallel \tilde{\mu} a . c \rangle_e & \rightsquigarrow c_e[p/a] \\
\langle p \parallel E \rangle_e & \rightsquigarrow \langle p \parallel E \rangle_p \\
\langle \mu \alpha . c \parallel E \rangle_p & \rightsquigarrow c_e[E/\alpha] \\
\langle V \parallel E \rangle_p & \rightsquigarrow \langle V \parallel E \rangle_E \\
\langle V \parallel q \cdot e \rangle_E & \rightsquigarrow \langle V \parallel q \cdot e \rangle_V \\
\langle \lambda a . p \parallel q \cdot e \rangle_V & \rightsquigarrow \langle q \parallel \tilde{\mu} a . \langle p \parallel e \rangle \rangle_e
\end{align*}
\]
## Semantic artifacts

- **(Proofs)** \( p ::= V \mid \mu \alpha.c \)**
- **(Values)** \( V ::= a \mid \lambda a.p \)**
- **(Contexts)** \( e ::= E \mid \tilde{\mu}a.c \)**
- **(Co-values)** \( E ::= \alpha \mid p \cdot e \)**
- **(Commands)** \( c ::= \langle p \parallel e \rangle \)**

### Small steps

<table>
<thead>
<tr>
<th>Expression</th>
<th>Reduction to</th>
<th>Value Substitution</th>
<th>CPS Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>( e ) ( \langle p \parallel \tilde{\mu}a.c \rangle_e )</td>
<td>( \leadsto )</td>
<td>( c_e[p/a] )</td>
<td>( [\tilde{\mu}a.c]_e p \triangleq (\lambda a.[c]_c) p )</td>
</tr>
<tr>
<td>( p ) ( \langle p \parallel E \rangle_p )</td>
<td>( \leadsto )</td>
<td>( \langle p \parallel E \rangle_p )</td>
<td>( [E]_e p \triangleq p [E]_E )</td>
</tr>
<tr>
<td>( p ) ( \langle \mu \alpha.c \parallel E \rangle_p )</td>
<td>( \leadsto )</td>
<td>( c_e[E/\alpha] )</td>
<td>( [\mu \alpha.c]_p E \triangleq (\lambda \alpha.[c]_c) E )</td>
</tr>
<tr>
<td>( p ) ( \langle V \parallel E \rangle_p )</td>
<td>( \leadsto )</td>
<td>( \langle V \parallel E \rangle_p )</td>
<td>( [V]_p E \triangleq E [V]_V )</td>
</tr>
<tr>
<td>( E ) ( \langle V \parallel q \cdot e \rangle_E )</td>
<td>( \leadsto )</td>
<td>( \langle V \parallel q \cdot e \rangle_E )</td>
<td>( [q \cdot e]_E V \triangleq V [q]_p [e]_e )</td>
</tr>
<tr>
<td>( V ) ( \langle \lambda a.p \parallel q \cdot e \rangle_V )</td>
<td>( \leadsto )</td>
<td>( \langle q \parallel \tilde{\mu}a.\langle p \parallel e \rangle \rangle_e )</td>
<td>( [\lambda a.p]_V q e \triangleq (\lambda a.e [p]_p) q )</td>
</tr>
</tbody>
</table>
Semantic artifacts

(Proofs) \( p ::= V \mid \mu\alpha.c \)

(Values) \( V ::= a \mid \lambda a.p \)

(Contexts) \( e ::= E \mid \tilde{\mu}a.c \)

(Co-values) \( E ::= \alpha \mid p \cdot e \)

(Command) \( c ::= \langle p \parallel e \rangle \)

Small steps

\[ \begin{array}{ll}
\langle p \parallel \tilde{\mu}a.c \rangle_e & \rightsquigarrow \quad c_e[p/a] \\
\langle p \parallel E \rangle_e & \rightsquigarrow \quad \langle p \parallel E \rangle_p \\
\langle \mu\alpha.c \parallel E \rangle_p & \rightsquigarrow \quad c_e[E/\alpha] \\
\langle V \parallel E \rangle_p & \rightsquigarrow \quad \langle V \parallel E \rangle_E \\
\langle V \parallel q \cdot e \rangle_E & \rightsquigarrow \quad \langle V \parallel q \cdot e \rangle_V \\
\langle \lambda a.p \parallel q \cdot e \rangle_V & \rightsquigarrow \quad \langle q \parallel \tilde{\mu}a.\langle p \parallel e \rangle \rangle_e \\
\end{array} \]

CPS

\[ \begin{array}{ll}
[\tilde{\mu}a.c]_e p \triangleq (\lambda a.[c]_c)_p \\
[E]_e p \triangleq p [E]_E \\
[\mu\alpha.c]_p E \triangleq (\lambda \alpha.[c]_c)_E \\
[V]_p E \triangleq E [V]_V \\
[q \cdot e]_E V \triangleq V [q]_p [e]_e \\
[\lambda a.p]_V q e \triangleq (\lambda a.e [p]_p)_q \\
\end{array} \]

\[ c \rightsquigarrow c' \quad \Rightarrow \quad [[c]]_c \xrightarrow{\beta} [[c']]_c \]
## Semantic artifacts

(Proofs) \( p ::= V \mid \mu \alpha . c \)

(Values) \( V ::= a \mid \lambda a . p \)

(Contexts) \( e ::= E \mid \tilde{\mu} a . c \)

(Co-values) \( E ::= \alpha \mid p \cdot e \)

(Commands) \( c ::= \langle p \parallel e \rangle \)

### CPS

<table>
<thead>
<tr>
<th>Expression</th>
<th>CPS Form</th>
<th>Types translation</th>
</tr>
</thead>
<tbody>
<tr>
<td>([\tilde{\mu} a . c]_e p \triangleq (\lambda a . [c]_c) p)</td>
<td>([A]_e \triangleq [A]_p \rightarrow \bot)</td>
<td></td>
</tr>
<tr>
<td>([E]_e p \triangleq p [E]_E)</td>
<td>([A]_p \triangleq [A]_E \rightarrow \bot)</td>
<td></td>
</tr>
<tr>
<td>([\mu \alpha . c]_p E \triangleq (\lambda \alpha . [c]_c) E)</td>
<td>([A]_E \triangleq [A]_V \rightarrow \bot)</td>
<td></td>
</tr>
<tr>
<td>([V]_p E \triangleq E [V]_V)</td>
<td>([A \rightarrow B]_V \triangleq [A]_p \rightarrow [A]_e \rightarrow \bot)</td>
<td></td>
</tr>
<tr>
<td>([q \cdot e]_E V \triangleq V [q]_p [e]_e)</td>
<td>([A \rightarrow B]_V \triangleq [A]_p \rightarrow [A]_e \rightarrow \bot)</td>
<td></td>
</tr>
<tr>
<td>([\lambda a . p]_V q e \triangleq (\lambda a . e [p]_p) q)</td>
<td>([A \rightarrow B]_V \triangleq [A]_p \rightarrow [A]_e \rightarrow \bot)</td>
<td></td>
</tr>
</tbody>
</table>

\[\Gamma \vdash p : A \mid \Delta \quad \Rightarrow \quad [[\Gamma]]_p, [[\Delta]]_E \vdash [[p]]_p : [[A]]_p\]
Consequences

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

Inhabitation
There is no simply-typed $\lambda$-term $t$ such that $\vdash t : [[\bot]]_p$.

Proof. $[[\bot]]_p = (\bot \rightarrow \bot) \rightarrow \bot$ and $\lambda x . x$ is of type $\bot \rightarrow \bot$.

Soundness
There is no proof $p$ such that $\vdash p : \bot \mid$.
Realizability à la Krivine (1/2)

**Intuition**
- falsity value $\|A\|$: contexts, opponent to $A$
- truth value $|A|$: proofs, player of $A$
- pole $\bot$: commands, referee

\[
\langle p \parallel e \rangle > c_0 > \cdots > c_n \in \bot?
\]

$\leadsto \bot \subset \Lambda \boxtimes \Pi$ closed by anti-reduction

Truth value defined by orthogonality:

\[
|A| = \|A\| \downarrow = \{ p \in \Lambda : \forall e \in \|A\|, \langle p \parallel e \rangle \in \bot \}
\]
Realizability à la Krivine (1/2)

Intuition

- falsity value $\|A\|$: contexts, opponent to $A$
- truth value $|A|$: proofs, player of $A$
- pole $\bot$: commands, referee

\[ \langle p \parallel e \rangle > c_0 > \cdots > c_n \in \bot \? \]

$\sim \bot \subset \Lambda \star \Pi$ closed by anti-reduction

Truth value defined by orthogonality:

\[ |A| = \|A\| \downarrow = \{ p \in \Lambda : \forall e \in \|A\|, \langle p \parallel e \rangle \in \bot \} \]
Realizability à la Krivine (1/2)

Intuition

- falsity value $\|A\|$: contexts, opponent to $A$
- truth value $|A|$: proofs, player of $A$
- pole $\bot$: commands, referee

$$\langle p \parallel e \rangle > c_0 > \cdots > c_n \in \bot?$$

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

Truth value defined by orthogonality:

$|A| = \|A\| \bot = \{p \in \Lambda : \forall e \in \|A\|, \langle p \parallel e \rangle \in \bot\}$
Realizability à la Krivine (1/2)

Intuition

- falsity value $\|A\|$: contexts, opponent to $A$
- truth value $|A|$: proofs, player of $A$
- pole $\bot$: commands, referee

$\langle p \parallel e \rangle > c_0 > \cdots > c_n \in \bot$?

$\leadsto \bot \subseteq \Lambda \star \Pi$ closed by anti-reduction

Truth value defined by orthogonality:

$|A| = \|A\| \bot = \{ p \in \Lambda : \forall e \in \|A\|, \langle p \parallel e \rangle \in \bot \}$
Realizability à la Krivine (1/2)

Intuition

- falsity value $\|A\|$: contexts, opponent to $A$
- truth value $|A|$: proofs, player of $A$
- pole $\bot$: commands, referee

$$\langle p \parallel e \rangle > c_0 > \cdots > c_n \in \bot ?$$

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

Truth value defined by orthogonality:

$$|A| = \|A\| \bot = \{ p \in \Lambda : \forall e \in \|A\|, \langle p \parallel e \rangle \in \bot \}$$
Realizability à la Krivine (1/2)

Intuition

- falsity value $\|A\|$: contexts, opponent to $A$
- truth value $|A|$: proofs, player of $A$
- pole $\bot$: commands, referee

$$\langle p \parallel e \rangle > c_0 > \cdots > c_n \in \bot?$$

$\leadsto \bot \subset \Lambda \star \Pi$ closed by anti-reduction

Truth value defined by orthogonality:

$$|A| = \|A\| \upharpoonright \bot = \{ p \in \Lambda : \forall e \in \|A\|, \langle p \parallel e \rangle \in \bot \}$$
Semantic artifacts++

(Terms) \( p ::= \mu \alpha . c \mid a \mid V \)

(Contexts) \( e ::= \tilde{\mu} a . c \mid E \)

(Values) \( V ::= \lambda a . p \)

(Co-values) \( E ::= \alpha \mid p \cdot e \)

Small steps

\[
\begin{align*}
\langle p \parallel \tilde{\mu} a . c \rangle_e & \rightsquigarrow c_e[p/a] \\
\langle p \parallel E \rangle_e & \rightsquigarrow \langle p \parallel E \rangle_p \\
\langle \mu \alpha . c \parallel E \rangle_p & \rightsquigarrow c_e[E/\alpha] \\
\langle V \parallel E \rangle_p & \rightsquigarrow \langle V \parallel E \rangle_E \\
\langle V \parallel q \cdot e \rangle_E & \rightsquigarrow \langle V \parallel q \cdot e \rangle_V \\
\langle \lambda a . p \parallel q \cdot e \rangle_V & \rightsquigarrow \langle q \parallel \tilde{\mu} a . \langle p \parallel e \rangle \rangle_e
\end{align*}
\]
Semantic artifacts++

**Terms**
\[ p ::= \mu a.c \mid a \mid V \]

**Values**
\[ V ::= \lambda a.p \]

**Contexts**
\[ e ::= \tilde{\mu} a.c \mid E \]

**Co-values**
\[ E ::= \alpha \mid p \cdot e \]

### Small steps

\[ e \]
\[ \langle p \parallel \tilde{\mu} a.c \rangle_e \rightsquigarrow c_e[p/a] \]
\[ \langle p \parallel E \rangle_e \rightsquigarrow \langle p \parallel E \rangle_p \]

\[ p \]
\[ \langle \mu a.c \parallel E \rangle_p \rightsquigarrow c_e[E/\alpha] \]
\[ \langle V \parallel E \rangle_p \rightsquigarrow \langle V \parallel E \rangle_E \]

\[ E \]
\[ \langle V \parallel q \cdot e \rangle_E \rightsquigarrow \langle V \parallel q \cdot e \rangle_V \]

\[ \langle \lambda a.p \parallel q \cdot e \rangle_V \rightsquigarrow \langle q \parallel \tilde{\mu} a.\langle p \parallel e \rangle \rangle_e \]

### Realizability

\[ \| A \|_e \triangleq \| A \|_p \upharpoonright \]
\[ | A \|_p \triangleq \| A \|_E \upharpoonright \]
\[ \| A \rightarrow B \|_E \triangleq \{ q \cdot e : q \in | A \|_p \wedge e \in \| B \|_e \} \]
Extension to second-order

\[
\frac{\Gamma \mid e : A[n/x] \vdash \Delta}{\Gamma \mid e : \forall x.A \vdash \Delta} \quad (\forall^1_i)
\]

\[
\frac{\Gamma \mid e : A[B/X] \vdash \Delta}{\Gamma \mid e : \forall X.A \vdash \Delta} \quad (\forall^2_i)
\]

\[
\frac{\Gamma \vdash p : A \mid \Delta \quad x \notin \text{FV}(\Gamma, \Delta)}{\Gamma \vdash p : \forall x.A \mid \Delta} \quad (\forall^1_r)
\]

\[
\frac{\Gamma \vdash p : A \mid \Delta \quad X \notin \text{FV}(\Gamma, \Delta)}{\Gamma \vdash p : \forall X.A \mid \Delta} \quad (\forall^2_r)
\]

(Curry-style)
Realizability à la Krivine (2/2)

Standard model $\mathbb{N}$ for 1$^\text{st}$-order expressions

Definition (Pole)

$\bot \subseteq \Lambda \times \Pi$ of commands s.t.:

$$\forall c, c', (c' \in \bot \land c \rightarrow c') \Rightarrow c \in \bot$$

Truth value (player):

$$|A|_p = \Vert A \Vert_E^\bot = \{ p \in \Lambda : \forall e \in \Vert A \Vert, \langle p \parallel e \rangle \in \bot \}$$

Falsity value (opponent):

$$\Vert F(e_1, \ldots, e_k) \Vert_E = F([e_1], \ldots, [e_k])$$

$$\Vert A \rightarrow B \Vert_E = \{ q \cdot e : q \in |A|_p \land e \in \Vert B \Vert_e \}$$

$$\Vert \forall x. A \Vert_E = \bigcup_{n \in \mathbb{N}} \Vert A[n/x] \Vert_E$$

$$\Vert \forall X. A \Vert_E = \bigcup_{F : \mathbb{N}^k \rightarrow \mathcal{P}(\Pi)} \Vert A[\bar{F}/X] \Vert_E$$

$$|A|_p = \Vert A \Vert_E^\bot = \{ p : \forall e \in \Vert A \Vert_E, \langle p \parallel e \rangle \in \bot \}$$

$$\Vert A \Vert_e = |A|_p^\bot = \{ e : \forall p \in |A|_p, \langle p \parallel e \rangle \in \bot \}$$
Realizability à la Krivine (2/2)

Standard model \( \mathbb{N} \) for 1\(^{\text{st}}\)-order expressions

**Definition (Pole)**

\[ \bot \subseteq \Lambda \times \Pi \text{ of commands s.t.:} \]

\[ \forall c, c', (c' \in \bot \land c \to c') \implies c \in \bot \]

**Truth value (player):**

\[ |A|_p = \|A\|_E \bot = \{ p \in \Lambda : \forall e \in \|A\|, \langle p \parallel e \rangle \in \bot \} \]

**Falsity value (opponent):**

\[ \| \hat{F}(e_1, \ldots, e_k) \|_E = F([e_1], \ldots, [e_k]) \]
\[ \| A \to B \|_E = \{ q \cdot e : q \in |A|_p \land e \in \|B\|_e \} \]
\[ \| \forall x.A \|_E = \bigcup_{n \in \mathbb{N}} \| A[n/x] \|_E \]
\[ \| \forall X.A \|_E = \bigcup_{F : \mathbb{N}^k \to \mathcal{P}(\Pi)} \| A[\hat{F}/X] \|_E \]
\[ |A|_p = \|A\|_E \bot = \{ p : \forall e \in \|A\|_E, \langle p \parallel e \rangle \in \bot \} \]
\[ \|A\|_e = |A|_p \bot = \{ e : \forall p \in |A|_p, \langle p \parallel e \rangle \in \bot \} \]
Realizability à la Krivine (2/2)

Standard model \( \mathbb{N} \) for 1st-order expressions

Definition (Pole)

\( \bot \subseteq \Lambda \times \Pi \) of commands s.t.:

\[ \forall c, c', (c' \in \bot \land c \rightarrow c') \Rightarrow c \in \bot \]

Truth value (player):

\[ |A|_p = \|A\|_E \bot = \{p \in \Lambda : \forall e \in \|A\|, \langle p \parallel e \rangle \in \bot \} \]

Falsity value (opponent):

\[ \|F(e_1, \ldots, e_k)\|_E = F([e_1], \ldots, [e_k]) \]
\[ \|A \rightarrow B\|_E = \{q \cdot e : q \in |A|_p \land e \in \|B\|_e \} \]
\[ \|\forall x.A\|_E = \bigcup_{n \in \mathbb{N}} \|A[n/x]\|_E \]
\[ \|\forall X.A\|_E = \bigcup_{F: \mathbb{N}^k \rightarrow \mathcal{P}^{(\Pi)}} \|A[\hat{F}/X]\|_E \]
\[ |A|_p = \|A\|_E \bot = \{p : \forall e \in \|A\|_E, \langle p \parallel e \rangle \in \bot \} \]
\[ \|A\|_e = |A|_p \bot = \{e : \forall p \in |A|_p, \langle p \parallel e \rangle \in \bot \} \]
Adequacy

Valuation $\rho$:

$$\rho(x) \in \mathbb{N} \quad \rho(X) : \mathbb{N}^k \rightarrow \mathcal{P}(\Pi)$$

Substitution $\sigma$:

$$\sigma ::= \varepsilon \mid \sigma, a := p \mid \sigma, \alpha ::= E$$

$$\sigma \models \Gamma \overset{\Delta}{=} \begin{cases} 
\sigma(a) \in |A|_p \\
\forall (a : A) \in \Gamma \\
\sigma(\alpha) \in \|A\|_E \\
\forall (\alpha : A \bot) \in \Gamma
\end{cases}$$

If $\sigma \models (\Gamma \cup \Delta)[\rho]$, then:

1. $\Gamma \vdash p : A \mid \Delta \Rightarrow p[\sigma] \in |A[\rho]|_p$
2. $\Gamma \mid e : A \vdash \Delta \Rightarrow e[\sigma] \in \|A[\rho]\|_e$
3. $c : (\Gamma \vdash \Delta) \Rightarrow c[\sigma] \in \bot$

Proof. By mutual induction over the typing derivation.
Results

Normalizing commands

\[ \bot \downarrow \triangleq \{ c : c \text{ normalizes} \} \text{ defines a valid pole.} \]

Proof. If \( c \rightarrow c' \) 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 \( \bot \downarrow \).

Soundness

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

Proof. Otherwise, \( p \in \bot \vdash = \Pi_{\bot} \) for any pole, absurd (\( \bot \triangleq \emptyset \)).
Classical call-by-need
Reminder

\[ \lambda_{[l\nu\tau\star]} \]

**dPA^\omega** [Herbelin'12]:
- control operators
- dependent types
- co-fixpoints
- sharing & laziness

Subject reduction

typing/reduction preservation

**dLPA^\omega?**
- sequent calculus
- dependent types
- co-fixpoints
- sharing & laziness

Subject reduction

CPS-translation?

?-calculus

Normalization
A constructive proof of DC
Semantic artifacts
Classical call-by-need
dL
dLPA

Classical call-by-need

The $\lambda[l\nu\tau\star]$-calculus:
- a sequent calculus with explicit “stores”
- Danvy’s method of semantics artifact:
  1. derive a small-step reduction system
  2. derive context-free small-step reduction rules
  3. derive an (untyped) CPS

Questions:
- Does it normalize?
- Can the CPS be typed?
- Can we define a realizability interpretation?
## The $\bar{\lambda}[l\nu\tau\star]$-calculus

### Syntax:

| Proofs | $p ::= V | \mu\alpha.c$ |
|--------|--------------------------|
| Weak values | $V ::= v | a$ |
| Strong values | $v ::= \lambda a.p | k$ |

| Proofs | $e ::= E | \tilde{\mu}a.c$ |
|--------|--------------------------|
| Contexts | $E ::= \alpha | F | \tilde{\mu}[a].\langle a || F \rangle \tau$ |
| Catchable contexts | $F ::= p \cdot E | \kappa$ |

| Commands | $c ::= \langle p || e \rangle$ |
|----------|--------------------------|

| Closures | $l ::= c\tau$ |

| Store | $\tau ::= e | \tau[a := p]$ |

### Reduction rules:

| Lazy storage | $\langle p || \tilde{\mu}a.c \rangle \tau \rightarrow c\tau[a := p]$ |
|--------------|--------------------------------------------------------|
| Lookup       | $\langle \mu\alpha.c || E \rangle \tau \rightarrow (c[E/\alpha])\tau$ |
| Forced eval. | $\langle a || F \rangle \tau[a := p] \tau' \rightarrow \langle p || \tilde{\mu}[a].\langle a || F \rangle \tau' \rangle \tau$ |
| Forced eval. | $\langle V || \tilde{\mu}[a].\langle a || F \rangle \tau' \rangle \tau \rightarrow \langle V || F \rangle \tau[a := V] \tau'$ |
| Forced eval. | $\langle \lambda a.p || q \cdot E \rangle \tau \rightarrow \langle q || \tilde{\mu}a.\langle p || E \rangle \rangle \tau$ |
Semantic artifacts

Small steps:

\[
\begin{align*}
e & : 
\langle p \parallel \tilde{\mu}a.c \rangle_e \tau \rightarrow c_e \tau [a := p] \\
p & : 
\langle \mu\alpha.c \parallel E \rangle_p \tau \rightarrow (c[E/\alpha]) \tau \\
E & : 
\langle V \parallel F \rangle_{E\tau} \rightarrow \langle V \parallel F \rangle_{V\tau} \\
V & : 
\langle a \parallel F \rangle_{V\tau} [a := p] \tau' \rightarrow \langle p \parallel \tilde{\mu}[a].\langle a \parallel F \rangle_{\tau'} \rangle_p \tau \\
F & : 
\langle \lambda a.p \parallel q \cdot E \rangle_{F\tau} \rightarrow \langle q \parallel \tilde{\mu}a.\langle p \parallel E \rangle \rangle_e \tau
\end{align*}
\]
Semantic artifacts

CPS :

\[
\begin{align*}
\llbracket \langle p \parallel e \rangle \tau \rrbracket & := [e]_{e} \llbracket \tau \rrbracket \llbracket p \rrbracket_{p} \\
\llbracket \tilde{\mu} a . c \rrbracket_{e} & := \lambda \tau p . \llbracket c \rrbracket \tau [a := p] \\
\llbracket E \rrbracket_{e} & := \lambda \tau p . p \tau \llbracket E \rrbracket_{E} \\
\llbracket \mu \alpha . c \rrbracket_{p} & := \lambda \tau E . (\llbracket c \rrbracket_{c} \tau)[E/\alpha] \\
\llbracket V \rrbracket_{p} & := \lambda \tau E . E \tau \llbracket V \rrbracket_{V} \\
\llbracket \tilde{\mu} [a] . \langle a \parallel F \rangle \tau' \rrbracket_{E} & := \lambda \tau V . V \tau [a := V] \tau' \llbracket F \rrbracket_{F} \\
\llbracket F \rrbracket_{E} & := \lambda \tau V . V \tau \llbracket F \rrbracket_{F} \\
\llbracket a \rrbracket_{v} & := \lambda \tau F . \tau (a) \tau (\lambda \tau V . V \tau [a := V] \tau' \llbracket F \rrbracket_{F}) \\
\llbracket \lambda a . p \rrbracket_{v} & := \lambda \tau F . F \tau (\lambda q \tau E . \llbracket p \rrbracket_{p} \tau [a := q] E) \\
\llbracket q \cdot E \rrbracket_{F} & := \lambda \tau v . v [q]_{p} \tau \llbracket E \rrbracket_{E}
\end{align*}
\]
## Semantic artifacts

### Small-step:

<table>
<thead>
<tr>
<th>e</th>
<th>(\langle p \parallel \tilde{\mu}a.c \rangle_e \tau \rightarrow \ldots)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(\langle p \parallel E \rangle_e \tau \rightarrow \ldots)</td>
</tr>
<tr>
<td>p</td>
<td>(\langle \mu\alpha.c \parallel E \rangle_p \tau \rightarrow \ldots)</td>
</tr>
<tr>
<td></td>
<td>(\langle V \parallel E \rangle_p \tau \rightarrow \ldots)</td>
</tr>
<tr>
<td>E</td>
<td>(\langle V \parallel \tilde{\mu}[a].\langle a \parallel F \rangle \tau' \rangle_E \tau \rightarrow \ldots)</td>
</tr>
<tr>
<td></td>
<td>(\langle V \parallel F \rangle_E \tau \rightarrow \ldots)</td>
</tr>
<tr>
<td>V</td>
<td>(\langle a \parallel F \rangle_V \tau[a := p] \tau' \rightarrow \ldots)</td>
</tr>
<tr>
<td></td>
<td>(\langle \nu \parallel F \rangle_V \tau \rightarrow \ldots)</td>
</tr>
<tr>
<td>F</td>
<td>(\langle \nu \parallel q \cdot E \rangle_F \tau \rightarrow \ldots)</td>
</tr>
<tr>
<td>\nu</td>
<td>(\langle \lambda a.p \parallel q \cdot E \rangle_{\nu} \tau \rightarrow \ldots)</td>
</tr>
</tbody>
</table>
A constructive proof of DC

Semantic artifacts

Classical call-by-need
dL
dLPA

Realizability:

\[( \bot \subseteq ? )\]

\[\|A\|_e := \{ e? \in |A|_p \ \bot \}\]  

\[|A|_p := \{ p? \in \|A\|_E \ \bot \}\]  

\[\|A\|_E := \{ E? \in |A|_V \ \bot \}\]  

\[|A|_V := \{ V? \in \|A\|_F \ \bot \}\]  

\[\|A\|_F := \{ F? \in |A|_V \ \bot \}\]  

\[|A \rightarrow B|_v := \{ \lambda a.p? : q? \in |A|_t \Rightarrow p[q/a]? \in |B|_t\}\]
**Semantic artifacts**

**Small-step:**

- $e$
  \[ \langle \mu \alpha . c \| E \rangle_{p \tau} \rightarrow \ldots \]
  \[ \langle p \| E \rangle_{e \tau} \rightarrow \ldots \]

- $p$
  \[ \langle V \| E \rangle_{p \tau} \rightarrow \ldots \]
  \[ \langle \mu \alpha . c \| E \rangle_{p \tau} \rightarrow \ldots \]

- $E$
  \[ \langle V \| \tilde{\mu}[a] . \langle a \| F \rangle_{\tau'} \rangle_{E \tau} \rightarrow \ldots \]
  \[ \langle V \| F \rangle_{E \tau} \rightarrow \ldots \]

- $V$
  \[ \langle a \| F \rangle_{V \tau} [a := p]_{\tau'} \rightarrow \ldots \]
  \[ \langle \nu \| F \rangle_{V \tau} \rightarrow \ldots \]

- $F$
  \[ \langle \nu \| q \cdot E \rangle_{F \tau} \rightarrow \ldots \]

- $\nu$
  \[ \langle \lambda a . p \| q \cdot E \rangle_{\nu \tau} \rightarrow \ldots \]

**Realizability:**

\[ (\bot \subseteq \Lambda \times \Pi \times \tau) \]

\[ \| A \|_{e} := \{ e? \in |A|_{p} \} \]

\[ |A|_{p} := \{ p? \in \| A\|_{E} \} \]

\[ |A|_{E} := \{ E? \in \| A\|_{V} \} \]

\[ |A|_{V} := \{ V? \in \| A\|_{F} \} \]

\[ |A|_{F} := \{ F? \in \| A\|_{\nu} \} \]

\[ |A \rightarrow B|_{\nu} := \{ \lambda a . p? : q? \in |A|_{t} \Rightarrow p[q/a]? \in |B|_{t} \} \]
A constructive proof of DC

Semantic artifacts

Classical call-by-need
dL
dLPA

Small-step:

\[ \langle p \parallel \mu a.c \rangle_e \tau \rightarrow \ldots \]
\[ \langle p \parallel E \rangle_e \tau \rightarrow \ldots \]
\[ \langle \mu \alpha.c \parallel E \rangle_p \tau \rightarrow \ldots \]
\[ \langle V \parallel E \rangle_p \tau \rightarrow \ldots \]
\[ \langle V \parallel \tilde{\mu}[a].\langle a \parallel F \rangle \tau' \rangle_E \tau \rightarrow \ldots \]
\[ \langle V \parallel F \rangle_E \tau \rightarrow \ldots \]
\[ \langle a \parallel F \rangle_V \tau [a := p] \tau' \rightarrow \ldots \]
\[ \langle v \parallel F \rangle_V \tau \rightarrow \ldots \]
\[ \langle v \parallel q \cdot E \rangle_F \tau \rightarrow \ldots \]
\[ \langle \lambda a.p \parallel q \cdot E \rangle_v \tau \rightarrow \ldots \]

Realizability:

\[ (\bot \subseteq \Lambda \times \Pi \times \tau) \]
\[ \|A\|_e := \{(e|\tau) \in |A|_p \bot \} \]
\[ |A|_p := \{(p|\tau) \in \|A\|_E \bot \} \]
\[ \|A\|_E := \{(E|\tau) \in |A|_V \bot \} \]
\[ |A|_V := \{(V|\tau) \in \|A\|_F \bot \} \]
\[ \|A\|_F := \{(F|\tau) \in |A|_v \bot \} \]
\[ |A \rightarrow B|_v := \{(\lambda a.p|\tau) : (q|\tau') \in |A|_t \Rightarrow (p|\tau\tau'[a := q]) \in |B|_t \} \]
Realizability interpretation

A few novelties:

- **Term-in-store** \((t|\tau)\):

\[
FV(t) \subseteq \text{dom}(\tau), \ \tau \text{ closed}
\]

- **Pole**: set of closures \(\bot\) which is:
  - **saturated**:

\[
c'\tau' \in \bot \quad \text{and} \quad c\tau \rightarrow c'\tau' \quad \text{implies} \quad c\tau \in \bot
\]

- **closed by store extension**:

\[
c\tau \in \bot \quad \text{and} \quad \tau \triangleleft \tau' \quad \text{implies} \quad c\tau' \in \bot
\]

- **Orthogonality**:

\[
(t|\tau) \perp (e|\tau') \triangleq \tau, \tau' \text{ compatible} \quad \land \quad \langle t \parallel e \rangle \tau\tau' \in \bot.
\]

- **Realizers**: definitions derived from the small-step rules!
Realizability interpretation

A few novelties:

- **Term-in-store** \( (t|\tau) \): 
  \[
  \text{FV}(t) \subseteq \text{dom}(\tau), \tau \text{ closed}
  \]

- **Pole** : set of closures \( \bot \) which is:
  - **saturated**:
    \[
    c'\tau' \in \bot \quad \text{and} \quad c\tau \to c'\tau' \quad \text{implies} \quad c\tau \in \bot
    \]
  - **closed by store extension**:
    \[
    c\tau \in \bot \quad \text{and} \quad \tau \sqsubset \tau' \quad \text{implies} \quad c\tau' \in \bot
    \]

- **Orthogonality** :
  \[
  (t|\tau) \bot (e|\tau') \triangleq \tau, \tau' \text{ compatible} \land \langle t \parallel e \rangle_{\tau\tau'} \in \bot.
  \]

- **Realizers**: definitions derived from the small-step rules!
Realizability interpretation

Adequacy

For all \( \Downarrow \), if \( \tau \vdash \Gamma \) and \( \Gamma \vdash_c c \), then \( c\tau \in \Downarrow \).

Normalization

If \( \vdash_c c\tau \) then \( c\tau \) normalizes.

Proof: The set \( \Downarrow \Downarrow = \{ c\tau \in C_0 : c\tau \text{ normalizes} \} \) is a pole.
Realizability interpretation

Adequacy

For all $\bot$, if $\tau \vdash \Gamma$ and $\Gamma \vdash_c c$, then $c\tau \in \bot$.

Normalization

If $\vdash_I c\tau$ then $c\tau$ normalizes.

Proof: The set $\bot_{\downarrow\downarrow} = \{c\tau \in C_0 : c\tau$ normalizes $\}$ is a pole.

Initial questions:

- Does it normalize? Yes!
- Can the CPS be typed? Yes! (but it is complicated...)
- Can we define a realizability interpretation? Yes!
A sequent calculus with dependent types
Reminder

\[ \text{dPA}^{\omega} \ [\text{Herbelin'12}]: \]
- control operators
- dependent types
- co-fixpoints
- sharing & laziness

\[ \text{dLPA}^{\omega} \]
- sequent calculus
- dependent types
- co-fixpoints
- sharing & laziness

\( \text{dL} ? \)

\( \text{dLPA}^{\omega} ? \)

Subject reduction

typing/reduction preservation

CPS-translation?

?-calculus

Normalization
Can this work?

\[
\begin{align*}
\Gamma, a : A &\vdash p : B[a] \mid \Delta \quad (\rightarrow_r) \\
\Pi_p & \\
\Gamma &\vdash \lambda a.p : \Pi_{(a:A)}.B \mid \Delta \\
\Pi_q & \\
\Gamma + q : A &\mid \Delta \quad \Gamma \mid e : B[q] \vdash \Delta \quad q \in V \quad (\rightarrow_l) \\
\Pi_e & \\
\Gamma &\vdash q \cdot e : \Pi_{(a:A)}.B \mid \Delta \\
\Pi & \\
\langle \lambda a.p \parallel q \cdot e \rangle & : (\Gamma \vdash \Delta) \quad (\text{Cut})
\end{align*}
\]
A classical sequent calculus with dependent types

**Can this work?**

\[
\begin{align*}
\Pi_p & \quad \Pi_q & \quad \Pi_e \\
\vdots & \quad \vdots & \quad \vdots \\
\Gamma, a : A \vdash p : B[a] | \Delta & \quad \Gamma \vdash q : A | \Delta & \quad \Gamma \vdash e : B[q] \vdash \Delta \\
\Gamma \vdash \lambda a. p : \Pi_{(a:A)}. B | \Delta & \quad q \in V & \quad q \in V \\
\frac{\langle \lambda a. p \parallel q \cdot e \rangle : (\Gamma \vdash \Delta)}{(\rightarrow_r)} & \quad \frac{\langle \lambda a. p \parallel q \cdot e \rangle : (\Gamma \vdash \Delta)}{(\rightarrow_l)} & \quad \frac{\langle \lambda a. p \parallel q \cdot e \rangle : (\Gamma \vdash \Delta)}{(Cut)}
\end{align*}
\]

\[
\begin{align*}
\Pi_q & \\
\vdots & \\
\Gamma \vdash q : A | \Delta \\
\frac{\langle p \parallel e \rangle : (\Gamma, a : A \vdash \Delta)}{(\rightarrow_l)} & \quad \frac{\langle \lambda a. p \parallel q \cdot e \rangle : (\Gamma \vdash \Delta)}{(\rightarrow_l)} & \quad \frac{\langle \lambda a. p \parallel q \cdot e \rangle : (\Gamma \vdash \Delta)}{(Cut)}
\end{align*}
\]

Mismatch

Mismatch

Mismatch
A classical sequent calculus with dependent types

Can this work? ✓

\[
\begin{align*}
\Gamma, a : A \vdash p : B[a] | \Delta & \quad (\rightarrow_r) \\
\Gamma \vdash \lambda a. p : \Pi_{(a:A)} B | \Delta & \\
\langle \lambda a. p \parallel q \cdot e \rangle : (\Gamma \vdash \Delta)
\end{align*}
\]

\[
\begin{align*}
\Gamma \vdash q : A | \Delta & \\
\Gamma \vdash e : B[q] \vdash \Delta & \\
q \in V
\end{align*}
\]

\[
\begin{align*}
\Pi_p & \\
\Pi_q & \\
\Pi_e & \\
\Gamma \vdash q : A | \Delta & \\
\Gamma \vdash e : B[q] \vdash \Delta & \\
q \in V
\end{align*}
\]

\[
\begin{align*}
\langle \lambda a. p \parallel q \cdot e \rangle : (\Gamma \vdash \Delta)
\end{align*}
\]

\[
\begin{align*}
\Pi_q & \\
\Gamma, a : A \vdash p : B[a] | \Delta & \\
\Gamma, a : A \vdash e : B[q] \vdash \Delta; \{\cdot|p\cdot|a\cdotq\} & \quad (\text{Cut})
\end{align*}
\]

\[
\begin{align*}
\langle q \parallel \tilde{\mu} a. \langle p \parallel e \rangle \rangle : (\Gamma \vdash \Delta); \{\cdot|\cdot\cdot\}
\end{align*}
\]
\[ \lambda\mu\tilde{\mu}-\text{calculus} + \text{dependent types with:} \]

- a list of dependencies:

\[
\frac{\Gamma \vdash p : A | \Delta; \sigma \quad \Gamma \mid e : A' \vdash \Delta; \sigma \{ \cdot | p \} \quad A' \in A_\sigma}{\langle p \parallel e \rangle : (\Gamma \vdash \Delta; \sigma)} \quad \text{(Cut)}
\]

- a value restriction

Is it enough?

- subject reduction
- normalization
- consistency as a logic
- suitable for CPS translation
A constructive proof of DC

Semantic artifacts

Classical call-by-need
dL

dLPA

λμᵦ-calculus + dependent types with:

- a list of dependencies:

\[
\Gamma \vdash p : A \mid \Delta; \sigma \quad \Gamma \mid e : A' \vdash \Delta; \sigma\{\cdot \mid p\} \quad A' \in A_\sigma
\]

\[
\langle p \parallel e \rangle : (\Gamma \vdash \Delta; \sigma)
\]

- a value restriction

Is it enough?

- subject reduction ✓
- normalization ✓
- consistency as a logic ✓
- suitable for CPS translation ✗
\( \lambda \mu \tilde{\mu} \)-calculus + dependent types with:

- a list of dependencies:

\[
\frac{\Gamma \vdash p : A \mid \Delta; \sigma \quad \Gamma \mid e : A' \vdash \Delta; \sigma \{ \cdot | p \} \quad A' \in A_\sigma}{\langle p \parallel e \rangle : (\Gamma \vdash \Delta; \sigma)} \tag{Cut}
\]

- a value restriction

Is it enough?

- subject reduction \( \checkmark \)
- normalization \( \checkmark \)
- consistency as a logic \( \checkmark \)
- suitable for CPS translation \( \times \)

\[
\left[ q \right] \left[ \tilde{\mu} a. \langle p \parallel e \rangle \right] = \left[ q \right] (\lambda a. \left[ p \right] \left[ e \right])
\]

\[
\neg \neg A \quad \neg \neg B(a) \quad \neg B(q)
\]
Toward a CPS translation (1/2)

This is quite normal:

- we observed a desynchronization
- we compensated only within the type system

→ we need to do this already in the calculus!

Who’s guilty?

\[
\beta \langle q \parallel \tilde{\mu}a.(p \parallel e) \rangle = [q] (\lambda a.[p][e])
\]

**Motto:** \( [p] \) shouldn’t be applied to \( [e] \) before \( [q] \) has reduced

\[
([q] (\lambda a.[p]))[e]
\]

So, we’re looking for:
Toward a CPS translation (1/2)

This is quite normal:

- we observed a desynchronization
- we compensated only within the type system

→ we need to do this already in the calculus!

Who’s guilty?

\[ \langle q \parallel \mu a.\langle p \parallel e \rangle \rangle = [q] (\lambda a. [p][e]) \]

**Motto:** \([p]\) shouldn’t be applied to \([e]\) before \([q]\) has reduced

\[ ([q] (\lambda a. [p]))[e] \]

So, we’re looking for:
Toward a CPS translation (1/2)

This is quite normal:
- we observed a desynchronization
- we compensated only within the type system

⇒ we need to do this already in the calculus!

Who’s guilty?

\[ [[q \parallel \tilde{\mu}a.\langle p \parallel e \rangle]] = [[q]] (\lambda a.[[p]][[e]]) \]

**Motto:** \([p]\) shouldn’t be applied to \([e]\) before \([q]\) has reduced

\[ ([[q]] (\lambda a.[[p]]))[[e]] \]

So, we’re looking for:
Toward a CPS translation (1/2)

This is quite normal:
- we observed a desynchronization
- we compensated only within the type system

⇒ we need to do this already in the calculus!

Who’s guilty?

\[
\left[ \langle q \parallel \tilde{\mu}a.\langle p \parallel e \rangle \rangle \right] = \left[ q \right] (\lambda a.\left[ p \right][e])
\]

Motto: \([p]\) shouldn’t be applied to \([e]\) before \([q]\) has reduced

\[
\left( \left[ q \right] (\lambda a.\left[ p \right]) \right)[e]
\]

So, we’re looking for:

\[
\langle \lambda a. p \parallel q \cdot e \rangle \to \langle \mu ? .\langle q \parallel \tilde{\mu}a.\langle p \parallel ? \rangle \rangle \parallel e \rangle
\]
Toward a CPS translation (1/2)

This is quite normal:
- we observed a desynchronization
- we compensated only within the type system

⇒ we need to do this already in the calculus!

Who’s guilty?

\[
\left[\langle q \parallel \hat{\mu}a.\langle p \parallel e \rangle \rangle\right] = \left[q \right] (\lambda a.\left[p\right] [e])
\]

**Motto:** \([p]\) shouldn’t be applied to \([e]\) before \([q]\) has reduced

\[
\left[\left[q \right] (\lambda a.\left[p\right])\right]\left[e\right]
\]

So, we’re looking for:

\[
\langle \lambda a. p \parallel q \cdot e \rangle \rightarrow \langle \mu \hat{t}p.\langle q \parallel \hat{\mu}a.\langle p \parallel \hat{t}p \rangle \rangle \parallel e \rangle
\]
Toward a CPS translation (2/2)

\[ \llbracket \langle \lambda a.p \parallel q \cdot e \rangle \rrbracket \xrightarrow{?} (\llbracket q \rrbracket (\lambda a.\llbracket p \rrbracket))[e] \]

Questions:
1. Is any \( q \) compatible with such a reduction?
2. Is this typable?
Toward a CPS translation (2/2)

\[
\langle \lambda a. p \parallel q \cdot e \rangle \quad \rightarrow \quad ([q] (\lambda a. [p]))[e]
\]

Questions:

1. Is any \( q \) compatible with such a reduction?

   - If \( q \) eventually gives a value \( V \):
     \[
     ([q] (\lambda a. [p]))[e] \rightarrow ((\lambda a. [p])[V])[e] \rightarrow [p][[V]/a][e] = [p[V/a]][e]
     \]
     ✓

   - If \([q] \rightarrow \lambda_. t\) and drops its continuation (meaning \( t : \bot \)):
     \[
     ([q] (\lambda a. [p]))[e] \rightarrow ((\lambda_. t)\lambda a. [p])[e] \rightarrow t[e]
     \]
     ✗
Toward a CPS translation (2/2)

\[\left[\langle\lambda a.p \parallel q \cdot e\rangle\right] \overset{?}{\rightarrow} \left([q] (\lambda a.[p]))[e]\right]\]

Questions:

1. Is any \(q\) compatible with such a reduction?

   - If \(q\) eventually gives a value \(V\):
     \[
     ([q] (\lambda a.[p]))[e] \rightarrow ((\lambda a.[p])[V])[e] \rightarrow [p][[V]/a][e] = [p[V/a]][e] \checkmark
     \]
   - If \([q]\) \(\rightarrow \lambda_.t\) and drops its continuation (meaning \(t : \bot\)):
     \[
     ([q] (\lambda a.[p]))[e] \rightarrow ((\lambda_.t)\lambda a.[p])[e] \rightarrow t[e] \times
     \]
Toward a CPS translation (2/2)

\[ \langle \lambda a.p \parallel q \cdot e \rangle \rightarrow ([q][a.p][e]) \]

Questions:

1. Is any \( q \) compatible with such a reduction? \( \Rightarrow q \in \text{NEF} \)

- If \( q \) eventually gives a value \( V \):
  \[ ([q](\lambda a.[p]))[e] \rightarrow ((\lambda a.[p])[V])[e] \rightarrow [p][V/a][e] = [p[V/a]][e] \]
- If \([q] \rightarrow \lambda_.t \) and drops its continuation (meaning \( t : \bot \)):
  \[ ([q](\lambda a.[p]))[e] \rightarrow ((\lambda_.t)[p])[e] \rightarrow t[e] \]

Negative-elimination free (Herbelin’12)

Values + one continuation variable + no application
Toward a CPS translation (2/2)

\[
\llbracket \langle \lambda a. p \parallel q \cdot e \rangle \rrbracket \rightarrow (\llbracket q \rrbracket \llbracket \lambda a. [p] \rrbracket) [e]
\]

Questions:
1. Is any \( q \) compatible with such a reduction? \( \leadsto q \in \text{NEF} \)
2. Is this typable?

Naive attempt:

\[
\begin{align*}
( & \llbracket q \rrbracket ) \\
(A \rightarrow \bot) & \rightarrow \bot \\
( & \llbracket \lambda a. [p] \rrbracket ) \\
\Pi_{(a:A)}^{\neg \neg} B(a) & \rightarrow \neg B[q]
\end{align*}
\]
Toward a CPS translation (2/2)

Questions:
1. Is any \( q \) compatible with such a reduction?  
   \( \leadsto q \in \text{NEF} \)
2. Is this typable?

Naive attempt:

\[
\frac{(A \rightarrow ?) \rightarrow ?}{\Pi_{(a:A)} \neg \neg B(a)} \quad \frac{(\lambda a . [p])}{\neg B[q]} \quad \frac{[q]}{\neg \neg B(q)}
\]
Toward a CPS translation (2/2)

\[
\left[\langle \lambda a.p \mid q \cdot e\rangle\right] \rightarrow (\left[ q \right] (\lambda a.\left[ p \right]))[e]
\]

Questions:
1. Is any \( q \) compatible with such a reduction? \( \sim \sim q \in \text{NEF} \)
2. Is this typable?

Friedman’s trick:

\[
\begin{align*}
\forall R. (A \rightarrow ?) \rightarrow ? & \quad (\lambda a.\left[ p \right]) \quad \Pi_{(a:A)} \neg \neg B(a) \\
\neg \neg B & \quad \left[ q \right] & \quad [e]
\end{align*}
\]
Toward a CPS translation (2/2)

$[[\lambda a.p \parallel q \cdot e]] \rightarrow ([q] (\lambda a.[p]))[e]$

Questions:

1. Is any $q$ compatible with such a reduction? $\leadsto q \in \text{NEF}$
2. Is this typable? $\leadsto \text{parametric return-type}$

Better:

$$(\forall R. (\Pi_{(a:A)} R(a)) \rightarrow R(q)) \cdot (\Pi_{(a:A)} \neg \neg B(a)) \cdot \neg \neg B[q]$$

(Remark: not possible without $q \in \text{NEF}$)
An extension of \( dL \) with:

- **delimited continuations**
- dependent types restricted to the **NEF fragment**
A constructive proof of DC
Semantic artifacts
Classical call-by-need
dL

\[ \mathbf{dL}_{\hat{t}p} \]

An extension of \( \mathbf{dL} \) with:
- **delimited continuations**
- dependent types restricted to the \( \text{NEF} \) fragment

**Reduction rules:**

\[
\langle \mu \hat{t}p. \langle p \parallel \hat{t}p \rangle \parallel e \rangle \to \langle p \parallel e \rangle
\]
\[
c \to c' \Rightarrow \langle \mu \hat{t}p. c \parallel e \rangle \to \langle \mu \hat{t}p. c' \parallel e \rangle
\]
\[
\langle \lambda a. p \parallel q \cdot e \rangle \to \langle \mu \hat{t}p. \langle q \parallel \tilde{\mu}a. \langle p \parallel \hat{t}p \rangle \rangle \parallel e \rangle
\]
\[
\langle \lambda a. p \parallel q \cdot e \rangle \to \langle q \parallel \tilde{\mu}a. \langle p \parallel e \rangle \rangle
\]
\[
\langle \text{prf} \ p \parallel e \rangle \to \langle \mu \hat{t}p. \langle p \parallel \tilde{\mu}a. \langle \text{prf} \ a \parallel \hat{t}p \rangle \rangle \parallel e \rangle
\]

\( q \in \text{NEF} \)
\( q \notin \text{NEF} \)
An extension of $dL$ with:
- **delimited continuations**
- dependent types restricted to the **NEF** fragment

**Typing rules:**

**Regular mode**

$$
\frac{
\Gamma \vdash p : A \mid \Delta \quad \Gamma \mid e : A \vdash \Delta
}{\langle p \parallel e \rangle : \Gamma \vdash \Delta}
$$

**Dependent mode**

$$
\frac{
\Gamma \vdash p : A \mid \Delta \quad \Gamma \mid e : A \vdash_{d} \Delta, \hat{tp} : B; \sigma\{\cdot | p\}
}{\langle p \parallel e \rangle : \Gamma \vdash_{d} \Delta, \hat{tp} : B; \sigma}
$$
An extension of dL with:

- **delimited continuations**
- dependent types restricted to the NEF fragment

**Typing rules:**

**Regular mode**

\[ \Gamma \vdash p : A \mid \Delta \quad \Gamma \mid e : A \vdash \Delta \]

\[ \langle p \parallel e \rangle : \Gamma \vdash \Delta \]

**Dependent mode**

\[ \Gamma \vdash p : A \mid \Delta \quad \Gamma \mid e : A \vdash_d \Delta, \hat{t}p : B ; \sigma \{\cdot \mid p\} \]

\[ \langle p \parallel e \rangle : \Gamma \vdash_d \Delta, \hat{t}p : B ; \sigma \]

**Use of \( \sigma \) limited to \( \hat{t}p \):**

\[ c : (\Gamma \vdash_d \Delta, \hat{t}p : A ; \{\cdot \mid \cdot\}) \]

\[ \Gamma \vdash \mu \hat{t}p.c : A \mid \Delta \]

\[ \hat{t}p_I \]

\[ B \in A_{\sigma} \]

\[ \Gamma \mid \hat{t}p : A \vdash_d \Delta, \hat{t}p : B ; \sigma \{\cdot \mid p\} \]

\[ \hat{t}p_E \]
An extension of dL with:
- delimited continuations
- dependent types restricted to the NEF fragment

Typing rules:

Regular mode

\[
\Gamma \vdash p : A \mid \Delta \quad \Gamma \mid e : A \vdash \Delta
\]

\[
\langle p \parallel e \rangle : \Gamma \vdash \Delta
\]

Dependent mode

\[
\Gamma \vdash p : A \mid \Delta \quad \Gamma \mid e : A \vdash_d \Delta, \hat{tp} : B; \sigma\{\cdot|p\}
\]

\[
\langle p \parallel e \rangle : \Gamma \vdash_d \Delta, \hat{tp} : B; \sigma
\]

Use of \(\sigma\) limited to \(\hat{tp}\):

\[
c : (\Gamma \vdash_d \Delta, \hat{tp} : A; \{\cdot|\cdot\})
\]

\[
\Gamma \vdash \mu \hat{tp}.c : A \mid \Delta
\]

\[
\hat{tp}_I
\]

\[
B \in A_\sigma
\]

\[
\Gamma \mid \hat{tp} : A \vdash_d \Delta, \hat{tp} : B; \sigma\{\cdot|p\}
\]

\[
\hat{tp}_E
\]

\[
c : (\Gamma \vdash \Delta) \quad \land \quad c \rightarrow c' \quad \Rightarrow \quad c' : (\Gamma \vdash \Delta)
\]
Typed CPS translation

Target language:
\[ \top | \bot | t = u | \forall x^{\mathbb{N}}. A | \exists x^{\mathbb{N}}. A | \Pi_{(a:A)} B | \forall X. A \]

Normalization:
If \([c]\) normalizes so does \(c\).

Proof. Thorough analysis of the several reduction rules.

Types-preserving:
The translation is well-typed.

Proof. Using parametric return types for terms and NEF proofs.

Consistency:
\[ \not\exists p : \bot. \]

Proof. \([\bot] = (\bot \to \bot) \to \bot\).
Bilan

An extension of dL with:

- **delimited continuations**
- dependent types restricted to the **NEF fragment**

**Regular mode**

\[
\frac{\Gamma \vdash p : A | \Delta \quad \Gamma \mid e : A \vdash \Delta}{\langle p \parallel e \rangle : \Gamma \vdash \Delta}
\]

**Dependent mode**

\[
\frac{\Gamma \vdash p : A | \Delta \quad \Gamma \mid e : A \vdash_{d} \Delta, \hat{t}p : B; \sigma\{\cdot | p\}}{\langle p \parallel e \rangle : \Gamma \vdash_{d} \Delta, \hat{t}p : B; \sigma}
\]

**delimited scope of dependencies:**

\[
\frac{c : (\Gamma \vdash_{d} \Delta, \hat{t}p : A; \{\cdot | \cdot\})}{\Gamma \vdash \mu \hat{t}p.c : A \mid \Delta}
\]

\[
\frac{B \in A_{\sigma}}{\Gamma \mid \hat{t}p : A \vdash_{d} \Delta, \hat{t}p : B; \sigma\{\cdot | p\}}
\]

**Mission accomplished?**

- subject reduction
- normalization
- consistency as a logic
- CPS translation

**Bonus** embedding into Rodolphe’s calculus ✓

⇒ realizability interpretation
Bilan

An extension of dL with:

- **delimited continuations**
- dependent types restricted to the **NEF fragment**

\[
\frac{\Gamma \vdash p : A \mid \Delta \quad \Gamma \mid e : A \vdash \Delta}{\langle p \parallel e \rangle : \Gamma \vdash \Delta}
\]

\[
\frac{\Gamma \vdash p : A \mid \Delta \quad \Gamma \mid e : A \vdash_d \Delta, \hat{t}p : B; \sigma\{\cdot|p\}}{\langle p \parallel e \rangle : \Gamma \vdash_d \Delta, \hat{t}p : B; \sigma}
\]

- delimited scope of dependencies:

\[
\frac{c : (\Gamma \vdash_d \Delta, \hat{t}p : A; \{\cdot|\cdot\})}{\Gamma \vdash \mu \hat{t}p.c : A \mid \Delta}
\]

\[
\frac{B \in A_\sigma}{\Gamma \mid \hat{t}p : A \vdash_d \Delta, \hat{t}p : B; \sigma\{\cdot|p\}}
\]

- Mission accomplished ✓

  - subject reduction ✓
  - normalization ✓
  - consistency as a logic ✓
  - CPS translation ✓

- *(Bonus)* embedding into Rodolphe’s calculus ✓

  - realizability
  - interpretation
Rodolphe’s calculus in a nutshell

Recipe:
- Call-by-value evaluation
- Classical language (μα.t control operator)
- Second-order logic, with encoding of dependent product:

\[ \Pi_{(a:A)} B \triangleq \forall a (a \in A \rightarrow B) \]

- Semantical value restriction
- Soundness and type safety proved by a realizability model:

\[ \Gamma \vdash t : A \Rightarrow \rho \models \Gamma \Rightarrow t[\rho] \in \|A\|_{\rho}^⊥⊥ \]

Semantical value restriction:
- observational equivalence: \( t \equiv u \)
- \( u \in A \) restricted to values
- typing rules up to this equivalence (hence undecidable!)
Rodolphe’s calculus in a nutshell

Recipe:
- Call-by-value evaluation
- Classical language ($\mu \alpha . t$ control operator)
- Second-order logic, with encoding of dependent product:

$$\Pi_{(a:A)} B \triangleq \forall a (a \in A \rightarrow B)$$

- Semantical value restriction
- Soundness and type safety proved by a realizability model:

$$\Gamma \vdash t : A \Rightarrow \rho \models \Gamma \Rightarrow t[\rho] \in \|A\|_{\rho}^{\perp \perp}$$

Semantical value restriction:
- observational equivalence: $t \equiv u$
- $u \in A$ restricted to values
- typing rules up to this equivalence (hence undecidable!)
Embedding

Easy check:

\[ \text{NEF} \subseteq \text{semantical values} \]

We define an embedding of proofs and types that:

- is **correct** with respect to typing

\[ \Gamma \vdash p : A \mid \Delta \quad \Rightarrow \quad (\Gamma \cup \Delta)^* \vdash \llbracket p \rrbracket_p : A^* \]

- is **adequate** with his realizability model

\[ \Gamma \vdash p : A \mid \Delta \quad \land \quad \sigma \models (\Gamma \cup \Delta)^* \quad \Rightarrow \quad \llbracket p \rrbracket_p \sigma \in \llbracket A \rrbracket \]

- allows to transfer Rodolphe’s safety results

\[ \nabla p : \bot \]
dLPA\(\omega\): a sequent calculus with dependent types for classical arithmetic
A classical sequent calculus with:
- stratified dependent types:
  - terms: $t, u ::= ... \mid \text{wit } p$
  - formulas: $A, B ::= ... \mid \forall x^T. A \mid \exists x^T. A \mid \Pi(a:A).B \mid t = u$
  - proofs: $p, q ::= ... \mid \lambda x. p \mid (t, p) \mid \lambda a.p$
- a restriction to the NEF fragment
- arithmetical terms:
  $$t, u ::= ... \mid 0 \mid S(t) \mid \text{rec}^t_{xy}[t_0 \mid t_S] \mid \lambda x. t \mid t u$$
- stores:
  $$\tau ::= \varepsilon \mid \tau[a := p_\tau] \mid \tau[\alpha := e]$$
- inductive and coinductive constructions:
  $$p, q ::= ... \mid \text{fix}^t_{bn}[p \mid p] \mid \text{cofix}^t_{bn} p$$
- a call-by-value reduction and lazy evaluation of cofix
End of the road

$dPA^\omega$
- control operators
- dependent types
- co-fixpoints
- sharing & laziness

Subject reduction

$dLPA^\omega$
- sequent calculus
- dependent types
- co-fixpoints
- sharing & laziness

Subject reduction

?-calculus

Normalization
A constructive proof of DC

Semantic artifacts

Classical call-by-need

dL

dLPA

ω

End of the road

Subject reduction

Subject reduction

Normalization

dPA\(^ω\)
+ control operators
+ dependent types
+ co-fixpoints
+ sharing & laziness

macros


Subject reduction

realizability

dLPA\(^ω\)
+ sequent calculus
+ dependent types
+ co-fixpoints
+ sharing & laziness

Étienne Miquey
A sequent calculus with dependent types for classical arithmetic
43/46
Realizability interpretation

Same methodology:

1. small-step reductions
2. derive the realizability interpretation

Resembles $\bar{\lambda}_{l\nu\tau\star}$-interpretation, plus:

- dependent types from Rodolphe’s calculus
- co-inductive formulas
Realizability interpretation

Same methodology:
1. small-step reductions
2. derive the realizability interpretation

Resembles $\tilde{\lambda}_{[\nu \tau \star]}$-interpretation, plus:
- dependent types from Rodolphe’s calculus:

$$\Pi_{(a:A)}.B \triangleq \forall a.(a \in A \to B)$$

- co-inductive formulas
Realizability interpretation

Same methodology:
1. small-step reductions
2. derive the realizability interpretation

Resembles $\bar{\lambda}_{[\nu \tau \star]}$-interpretation, plus:
- dependent types from Rodolphe’s calculus
- co-inductive formulas: by finite approximations

$$\| v^t_{xx} A \|_f \overset{\Delta}{=} \bigcup_{n \in \mathbb{N}} \| F^n_{A,t} \|_f$$
Realizability interpretation

Same methodology:

1. small-step reductions
2. derive the realizability interpretation

Resembles $\tilde{\lambda}_{\nu\tau^\star}$-interpretation, plus:

- dependent types from Rodolphe’s calculus
- co-inductive formulas: by finite approximations

**Consequences of adequacy:**

**Normalization**

If $\Gamma \vdash_\sigma c$, then $c$ is normalizable.

**Consistency**

$\not\vdash_{dLPA^\omega} p : \bot$
Conclusion

What did we learn?

- classical call-by-need:
  - realizability interpretation
  - typed continuation-and-store-passing style translation

- dependent classical sequent calculus:
  - list of dependencies
  - use of delimited continuations for soundness
  - dependently-typed continuation-passing style translation

- $d\text{LPA}^{\omega}$:
  - soundness and normalization,
  - realizability interpretation of co-fixpoints
Further work

1. **Classical call-by-need:**
   - typing the CPS with Kripke forcing

2. **\(dL_{\hat{tp}}\):**
   - Connection with:
     - Pédrot-Tabareau’s Baclofen Type Theory?
     - Vákár’s categorical presentation?
     - Bowman *et. al.* CPS for CC?
   - Dependent types & effects:

3. **Realizability:**
   - Connection with realizer for DC using bar recursion?
   - Algebraic counterpart of side-effects in realizability structures?
Further work

1. **Classical call-by-need:**
   - Typing the CPS with Kripke forcing

2. **$dL_{\hat{t}p}$:**
   - Connection with:
     - Pédrot-Tabareau’s Baclofen Type Theory?
     - Vákár’s categorical presentation?
     - Bowman *et. al.* CPS for CC?

3. **Dependent types & effects:**

4. **Realizability:**
   - Connection with realizer for DC using bar recursion?
   - Algebraic counterpart of side-effects in realizability structures?
Further work

1. **Classical call-by-need:**
   - typing the CPS with Kripke forcing

2. **$dL_{\hat{tp}}$:**
   - Connection with:
     - Pédrot-Tabareau’s Baclofen Type Theory?
     - Vákár’s categorical presentation?
     - Bowman *et. al.* CPS for CC?

3. **Dependent types & effects:**

   ![Diagram of embeddings and CPS transformations]

   - $\lambda$-calculus $\text{CbN}$ embed. $\lambda\mu\bar{\mu}$-calculus $\text{CbN}$ CPS $\lambda$-calculus
   - $\lambda$-calculus $\text{CbV}$ embed. $\lambda\mu\bar{\mu}$-calculus $\text{CbV}$ CPS $\lambda$-calculus

3. **Realizability:**
   - Connection with realizer for DC using bar recursion?
   - Algebraic counterpart of side-effects in realizability structures?
Further work

1. **Classical call-by-need:**
   - Typing the CPS with Kripke forcing

2. **$dL_{tp}$:**
   - Connection with:
     - Pédrot-Tabareau’s Baclofen Type Theory?
     - Vákár’s categorical presentation?
     - Bowman *et. al.* CPS for CC?

3. **Dependent types & effects:**

4. **Realizability:**
   - Connection with realizer for DC using bar recursion?
   - Algebraic counterpart of side-effects in realizability structures?
Further work

1. **Classical call-by-need:**
   - typing the CPS with Kripke forcing

2. **dL\(\hat{p}\):**
   - Connection with:
     - Pédrot-Tabareau’s Baclofen Type Theory?
     - Vákár’s categorical presentation?
     - Bowman *et. al.* CPS for CC?

3. **Dependent types & effects:**

   ![Diagram](attachment://diagram.png)

   - MLTT (CbN) embed. \rightarrow L_{dep}\? \rightarrow CPS \rightarrow MLTT
   - MLTT (CbV) embed. \rightarrow L_{dep}\? \rightarrow CPS \rightarrow MLTT

3. **Realizability:**
   - Connection with realizer for DC using bar recursion?
   - Algebraic counterpart of side-effects in realizability structures?
Further work

1. **Classical call-by-need:**
   - Typing the CPS with Kripke forcing

2. **$dL_{\hat{t}p}$:**
   - Connection with:
     - Pédrot-Tabareau’s Baclofen Type Theory?
     - Vákár’s categorical presentation?
     - Bowman et. al. CPS for CC?

3. **Dependent types & effects:**

   - MLTT (CbN)
     - embed.
   - $L_{dep}$?
   - CPS
   - MLTT

4. **Realizability:**
   - Connection with realizer for DC using bar recursion?
   - Algebraic counterpart of side-effects in realizability structures?
Thank you for your attention.