System L syntax for sequent calculi

Pierre-Louis Curien (in collaboration with Guillaume Munch-Maccagnoni)

February 3, 2012, University of Bath

(updated version of February 18, 2012)
Taxonomy of the talk

The eight systems of the talk, by order of appearance in the presentation (more or less the historical order as well):

- LL

- focalised systems in “direct style”: $\text{LK}_{pol}$, focalised LL

(for these 2 systems, the syntax presented here goes back to Guillaume Munch-Maccagnoni’s master thesis, cf. his CSL 2009 paper)

- focalised systems in “indirect style”: $\text{LL}_\downarrow$, Melliès’ tensor logic, $\text{LKQ}$, $\text{LK}_\downarrow$, (a sequent calculus version of) CBPV.

(the syntax for LKQ was presented in Curien - Munch-Maccagnoni’s IFIP TCS Conference 2010 paper)

Warning: this talk is very syntactic: (terms, typing rules, reduction rules)\(^8\).
I) Linear logic
Syntax for linear logic

Formulas :

\[ A ::= P | N \quad P ::= X | A \otimes A | A \oplus A | !A \quad N ::= \overline{X} | A \otimes A | A \& A | ?A \]

We use overlining for De Morgan duality.

There are three kinds of judgements :

<table>
<thead>
<tr>
<th>Commands</th>
<th>Positive terms</th>
<th>Negative terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c : (\vdash \Gamma) )</td>
<td>( \vdash t^+ : P</td>
<td>\Gamma )</td>
</tr>
</tbody>
</table>

Terms :

\[ c ::= \langle t^+ | t^- \rangle \quad \text{which we also write if needed as} \quad \langle t^- | t^+ \rangle \]
\[ t ::= t^+ | t^- \]
\[ x ::= x^+ | x^- \]
\[ t^+ ::= x^+ | \mu x^- . c \ | (t_1, t_2) \ | \text{inl}(t) \ | \text{inr}(t) \ | \mu x^! . c \]
\[ t^- ::= x^- | \mu x^+ . c | \mu(x_1, x_2).c | \mu[\text{inl}(x_1).c_1, \text{inr}(x_2).c_2] | t^! | w(c) | c_{x_1^+, x_2^+}(c) \]
Typing rules for $\mathbb{LL}$

Contexts $\Gamma$ consist of declarations $x^+ : N$ and $x^- : P$:

\[
\begin{align*}
& \vdash x : A \mid x : \overline{A} \\
& \vdash \mu x. c : A \mid \Gamma \\
& \vdash t^+ : P \mid \Gamma \vdash t^- : \overline{P} \mid \Delta \\
& \langle t^+ \mid t^- \rangle : (\vdash \Gamma, \Delta) \\
& \vdash t_1 : A_1 \mid \Gamma \vdash t_2 : A_2 \mid \Delta \\
& \vdash (t_1, t_2) : A_1 \otimes A_2 \mid \Gamma, \Delta \\
& \vdash \text{inl}(t_1) : A_1 \oplus A_2 \mid \Gamma \\
& \vdash \mu(x_1, x_2). c : A_1 \overline{\&} A_2 \mid \Gamma \\
& \vdash \mu[\text{inl}(x_1). c_1, \text{inr}(x_2). c_2] : A_1 \& A_2 \mid \Gamma \\
& \vdash x \mid \Gamma \\
& \vdash t : A \mid \Gamma \\
& \vdash \mu x^!. c : !A \mid ?\Gamma \\
& \vdash t^! : ?A \mid \Gamma \\
& \vdash \text{w}(c) : ?A \mid \Gamma \\
& \vdash \mu x_1^+, x_2^+. c : ?A \mid \Gamma \\
\end{align*}
\]
Reduction rules for LL

\[
\begin{align*}
\langle t^+ | \mu x+.c \rangle &\rightarrow c[t^+/x^+] \\
\langle \mu x-.c | t^- \rangle &\rightarrow c[t^-/x^-] \\
\langle (t_1, t_2) | \mu(x_1, x_2).c \rangle &\rightarrow c[t_1/x_1, t_2/x_2] \\
\langle inl(t_1) | \mu[inl(x_1).c_1, inr(x_2).c_2] \rangle &\rightarrow c_1[t_1/x_1] \\
\langle \mu x^! .c | t^! \rangle &\rightarrow c[t/x] \\
\langle t^+ | w(c) \rangle &\rightarrow W(c) \\
\langle t^+ | c_{x_1^+, x_2^+}(c) \rangle &\rightarrow C(c[t^+/x_1^+, t^+/x_2^+])
\end{align*}
\]

if the free variables of \( t^+ \) are in a list \( l = y_1, \ldots, y_n \) (with each \( y_i \) of type \(?B_i\)), then

- \( W(c) \) stands for \( W_l(c) \), where \( W_{\text{nil}}(c) = c \quad W_{y^+.l} = \langle y^+ | w(W_l(c)) \rangle \)
- \( C(c[t^+/x_1^+, t^+/x_2^+]) \) stands for \( C_l(c[t^+[l'/l]/x_1^+, t^+[l''/l]/x_2^+]) \), where

\[
C_{\text{nil}}(c) = c \quad C_{y^+.l} = \langle y^+ | c_{y^+, y^+(C_l(c))} \rangle
\]

(by \( l' \) we mean \( y_1'^+, \ldots, y_n'^+ \), and by \( t^+[l'/l] \) we mean the simultaneous substitutions of the \( y_i^+ \)'s by the \( y_i'^+ \)'s).
On the confluence of cut elimination in linear logic

The only critical pair is

\[ \langle \mu x_1^- . c_1 | \mu x_2^+ . c_2 \rangle \]

The “space” between \( c_1[\mu x_2^+ . c_2/x_1^-] \) and \( c_2[\mu x_1^- . c_1/x_2^+] \) can be filled by elementary commutations.

\[
\begin{align*}
\langle \mu x_1^- . C_1[\langle t_1^+ | x_1^- \rangle] | \mu x_2^+ . C_2[\langle x_2^+ | t_2^- \rangle] \rangle \\
\rightarrow C_1[\langle t_1^+ | x_1^- \rangle] \ [\mu x_2^+ . C_2[\langle x_2^+ | t_2^- \rangle]] / x_1^- \\
\Rightarrow C_1[\langle t_1^+ | \mu x_2^+ . C_2[\langle x_2^+ | t_2^- \rangle] \rangle] \\
\rightarrow C_1[C_2[\langle t_1^+ | t_2^- \rangle]]
\end{align*}
\]

while the other branch of the critical pair extends symmetrically to

\[
\begin{align*}
\langle \mu x_1^- . C_1[\langle t_1^+ | x_1^- \rangle] | \mu x_2^+ . C_2[\langle x_2^+ | t_2^- \rangle] \rangle \rightarrow^* C_2[C_1[\langle t_1^+ | t_2^- \rangle]]
\end{align*}
\]

The system is thus (locally) confluent modulo commutation.
Linear logic versus classical logic

A syntax for (a polarised version of) classical logic is obtained from the above by removing the exponential modalities and the term constructions for dereliction and promotion, but keeping explicit contraction and weakening, now expressed as:

\[
\frac{c : (\vdash \Gamma)}{\vdash w(c) : A | \Gamma} \quad \frac{c : (\vdash x_1 : A, x_2 : A, \Gamma)}{\vdash c_{x_1, x_2}(c) : A | \Gamma}
\]

Note that now explicit weakenings and contractions can be either positive or negative terms.
A syntax for a “polarity-aware” version of classical logic

Formulas :

\[
A ::= P | N \quad P ::= X | A \otimes A | A \oplus A \quad N ::= \overline{X} | A \& A | A\&A
\]

Terms :

\[
c ::= \langle t^+ | t^- \rangle \\
t ::= t^+ | t^- \\
x ::= x^+ | x^- \\
t^+ ::= x^+ | \mu x^-.c | (t_1, t_2) | inl(t) | inr(t) | w(c) | c_{x_1,x_2}(c) \\
t^- ::= x^- | \mu x^+.c | \mu(x_1, x_2).c | \mu[inl(x_1).c_1, inr(x_2).c_2] | w(c) | c_{x_1^+,x_2^+}(c)
\]
Tentative reduction rules for classical logic

\[ \langle t^+ | \mu x^+.c \rangle \rightarrow c[t^+/x^+] \]
\[ \langle \mu x^-.c | t^- \rangle \rightarrow c[t^-/x^-] \]
\[ \langle (t_1, t_2) | \mu(x_1, x_2).c \rangle \rightarrow c[t_1/x_1, t_2/x_2] \]
\[ \langle \text{inl}(t_1) | \mu[\text{inl}(x_1).c_1, \text{inr}(x_2).c_2)] \rangle \rightarrow c_1[t_1/x_1] \]
\[ \langle t^+ | w(c) \rangle \rightarrow W(c) \]
\[ \langle w(c) | t^- \rangle \rightarrow W(c) \]
\[ \langle t^+ | c_{x_1^+, x_2^+}(c) \rangle \rightarrow C(c[t^+/x_1^+, t^+/x_2^+]) \]
\[ \langle c_{x_1^-, x_2^-}(c) | t^- \rangle \rightarrow C(c[t^-/x_1^-, t^-/x_2^-]) \]

But now there are more critical pairs: the weakening/weakening pair (known as Lafont’s critical pair),

\[ W(c_1) \leftarrow \langle w(c_1) | w(c_2) \rangle \rightarrow W(c_2) \]

(for arbitrary proofs \( c_1, c_2 \)), as well as the contraction/contraction pair, also known to be problematic.
Discussion

Under these glasses, and in retrospect, one can see linear logic and focalisation as two alternative routes to get out of Lafont’s critical pair:

– focalised cut elimination (see below) restricts the dynamics in such a way that all the reduction rules are only applicable when they substitute values for positive variables. Then the bad critical pairs (as well as the non harmful ones) disappear, and one gets a confluent system without the need of appealing to commutation rules. This is a constraint on the syntax that still makes sense in an untyped setting.

– the introduction of the modalities makes the two bad critical pairs ill-typed. This is a constraint on types.
II) Focalised systems

1. $\text{LK}_{\text{pol}}$, where focalisation is badly needed for confluence
2. focalised LL, where focalisation was discovered first
II.1) Focalised classical logic
Syntax for focalised classical logic $\text{LK}_{\text{pol}}$

Same formulas:

\[
A ::= P \mid N \quad P ::= X \mid A \otimes A \mid A \oplus A \mid N ::= \overline{X} \mid A \otimes A \mid A \& A
\]

There are now four kinds of judgements:

- **Commands**
  - $c : (\vdash \Gamma) \vdash V^+P ; \Gamma$
- **Values**
  - $\vdash V^+ : P | \Gamma$
- **Positive terms**
  - $\vdash t^+ : P | \Gamma$
- **Negative terms**
  - $\vdash t^- : N | \Gamma$

It will be useful to have a common notation for values and negative terms:

\[
V ::= V^+ \mid t^- \quad \text{and} \quad \vdash V : A \parallel \Gamma \quad \text{stands for either} \quad \vdash V^+ : P ; \Gamma \quad \text{or} \quad \vdash t^- : N | \Gamma
\]

Terms:

\[
c ::= \langle t^+ \mid t^- \rangle \\
x ::= x^+ \mid x^- \\
V^+ ::= x^+ \mid (V_1, V_2) \mid \text{inl}(V) \mid \text{inr}(V) \\
t^+ ::= V^+ \mid \mu x^- . c \mid w(c) \mid \text{c}_{x_1^- , x_2^-} (c) \\
t^- ::= x^- \mid \mu x^+ . c \mid \mu (x_1, x_2) . c \mid \mu [\text{inl}(x_1) . c_1, \text{inr}(x_2) . c_2] \mid w(c) \mid \text{c}_{x_1^+ , x_2^+} (c)
\]
Typing rules for $\mathsf{LK}_{\text{pol}}$

\[ \vdash x^+ : P \; ; \; x^+ : \overline{P} \quad \vdash x^- : N \; | \; x^- : \overline{N} \]

\[ \vdash t^+ : P \; | \Gamma \quad \vdash t^- : \overline{P} \; | \Delta \]

\[ \langle t^+ \; | t^- \rangle : (\vdash \Gamma, \Delta) \]

\[ \vdash V^+ : P \; | \Gamma \quad c : (\vdash x : A, \Gamma) \]

\[ \vdash \mu x . c : A \; | \Gamma \]

\[ \vdash V_1 : A_1 \parallel \Gamma \quad \vdash V_2 : A_2 \parallel \Delta \]

\[ \vdash (V_1, V_2) : A_1 \otimes A_2 \; ; \Gamma, \Delta \]

\[ \vdash inl(V_1) : A_1 \oplus A_2 \; ; \Gamma \]

\[ c : (\vdash x_1 : A_1, x_2 : A_2, \Gamma) \]

\[ \vdash \mu (x_1, x_2) . c : A_1 \& A_2 \; | \Gamma \]

\[ c_1 : (\vdash x_1 : A_1, \Gamma) \quad c_2 : (\vdash x_2 : A_2, \Gamma) \]

\[ \vdash \mu [\text{inl}(x_1).c_1, \text{inr}(x_2).c_2] : A_1 \& A_2 \; | \Gamma \]

\[ c : (\vdash \Gamma) \]

\[ \vdash w(c) : A \; | \Gamma \]

\[ \vdash x_1 : A, x_2 : A, \Gamma \]

\[ \vdash c_{x_1,x_2}(c) : A \; | \Gamma \]
Reduction rules for $\mathrm{LK}_{\mathrm{pol}}$

\[
\begin{align*}
\langle V^+ \mid \mu x^+.c \rangle & \to c[V^+/x^+] \\
\langle \mu x^- .c \mid t^- \rangle & \to c[t^- / x^-] \\
\langle (V_1, V_2) \mid \mu (x_1, x_2).c \rangle & \to c[V_1/x_1, V_2/x_2] \\
\langle \text{inl}(V_1) \mid \mu [\text{inl}(x_1).c_1, \text{inr}(x_2).c_2) \rangle \rangle & \to c_1[V_1/x_1] \\
\langle V^+ \mid w(c) \rangle & \to W(c) \\
\langle w(c) \mid t^- \rangle & \to W(c) \\
\langle V^+ \mid c_{x_1^+, x_2^+}(c) \rangle & \to C(c[V^+/x_1^+, V^+/x_2^+]) \\
\langle c_{x_1^-, x_2^-}(c) \mid t^- \rangle & \to C(c[t^- / x_1^-, t^- / x_2^-])
\end{align*}
\]

Note that there is no critical pair anymore. We have regained consistency. The system presented here is a (close) variant of Girard’s LC.
Plotkin meets Andreoli

We have
– a call-by-value regime for positive variables
– a call-by-name regime for negative variables
Plotkin’s values correspond to positive phases in the focalisation discipline.
Removing a bit of bureaucracy

Now that we have carefully discussed the barriers to confluence, we can keep weakening and contraction implicit in the syntax (both for LL and LK\textsubscript{pol}) by defining $\overline{w}(c) = \mu x . c$ (with $x$ fresh) and $c_{x_1,x_2}(c) = \mu x_1 . c[x_2/x_1]$. Then the reductions rules consists only of:

\[
\begin{align*}
\langle V^+ | \mu x^+.c \rangle & \rightarrow c[V^+/x^+] \\
\langle \mu x^- . c | t^- \rangle & \rightarrow c[t^-/x^-] \\
\langle (V_1, V_2) | \mu(x_1, x_2).c \rangle & \rightarrow c[V_1/x_1, V_2/x_2] \\
\langle \text{inl}(V_1) | \mu[\text{inl}(x_1).c_1, \text{inr}(x_2).c_2] \rangle & \rightarrow c_1[V_1/x_1]
\end{align*}
\]

since now the dynamics of weakening and contraction is integrated in the dynamics of (implicit) substitution. This is the choice adopted for the rest of this talk.
A short perspective on focalisation

Focalisation appeared first in the context of linear logic (Andreoli), in the context of proof search: the goal was to *reduce the search space*.

The work of Andreoli influenced Girard for the design of LC.

The line of work of **Griffin** is independent, but (negative) polarisation is implicit in **Felleisen**'s CBN $\lambda C$-calculus, and focalisation is implicit in natural deduction (see e.g. Pfenning’s course notes).

So, let us return to linear logic and present its focalised version.
II.2 focalised linear logic
Syntax for focalised \( \text{LL} \)

The formulas are those of linear logic

\[
A ::= P | N \quad P ::= X | A \otimes A | A \oplus A | !A \quad N ::= \overline{X} | A \otimes A | A \& A | ?A
\]

Judgements:

<table>
<thead>
<tr>
<th>Commands</th>
<th>Values</th>
<th>Positive terms</th>
<th>Negative terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c : (\vdash \Gamma) )</td>
<td>( \vdash V^+P ; \Gamma )</td>
<td>( \vdash t^+ : P</td>
<td>\Gamma )</td>
</tr>
</tbody>
</table>

As above, we use a common notation for values and negative terms:

\( V ::= V^+ | t^- \) and \( \vdash V : A \| \Gamma \) stands for either \( \vdash V^+ : P ; \Gamma \) or \( \vdash t^- : N | \Gamma \)

Terms:

\[
c ::= \langle t^+ | t^- \rangle \\
x ::= x^+ | x^- \\
V^+ ::= x^+ | (V_1, V_2) | \text{inl}(V) | \text{inr}(V) | \mu x^! . c \\
t^+ ::= V^+ | \mu x^- . c \\
t^- ::= x^- | \mu x^+ . c | \mu(x_1, x_2).c | \mu[\text{inl}(x_1).c_1, \text{inr}(x_2).c_2] | V^!
\]
Typing rules for focalised $\mathbb{L}$

\[
\begin{align*}
\vdash x^+: P & \quad x^+: P \quad \vdash x^-: N & \quad x^-: N \\
\vdash \langle t^+\mid t^-\rangle : (\vdash \Gamma, \Delta) & \\
\vdash V^+: P & \quad c : (\vdash x : A, \Gamma) & \quad c : (\vdash x : A, ?\Gamma) \\
\vdash V^+: P \mid \Gamma & \quad \vdash \mu x. c : A \mid \Gamma & \quad \vdash \mu x^! . c : !A \mid ?\Gamma \\
\vdash V_1 : A_1 \parallel \Gamma & \quad \vdash V_2 : A_2 \parallel \Delta \\
\vdash (V_1, V_2) : A_1 \otimes A_2 \mid \Gamma, \Delta \\
\vdash inl(V_1) : A_1 \oplus A_2 \mid \Gamma \\
\vdash \mu (x_1, x_2). c : A_1 \otimes A_2 \mid \Gamma \\
\vdash \mu [inl(x_1). c_1, inr(x_2). c_2] : A_1 \& A_2 \mid \Gamma \\
\vdash c : (\vdash \Gamma) \\
\vdash c : (\vdash x^+: ?A, \Gamma) \\
\vdash c : (\vdash x^+_1 : ?A, x^+_2 : ?A, \Gamma) \\
\vdash c[x^+/x^+_1, x^+/x^+_2] : (\vdash x^+ : ?A, \Gamma)
\end{align*}
\]
Reduction rules for focalised $LL$

\[
\langle V^+ | \mu x^+.c \rangle \rightarrow c[V^+/x^+]
\]
\[
\langle \mu x^- .c \mid t^- \rangle \rightarrow c[t^-/x^-]
\]
\[
\langle (V_1, V_2) \mid \mu(x_1, x_2).c \rangle \rightarrow c[V_1/x_1, V_2/x_2]
\]
\[
\langle \text{inl}(V_1) \mid \mu[\text{inl}(x_1).c_1, \text{inr}(x_2).c_2] \rangle \rightarrow c_1[V_1/x_1]
\]
\[
\langle \mu x^! .c \mid V^! \rangle \rightarrow c[V/x]
\]

Note that there are no critical pairs anymore.
Completeness of focalised LL

Following a technique in Girard’s LC paper (adapted to LL in Laurent’s notes on focalisation), we exhibit a translation from LL proofs to focalised LL proofs. The translation is the identity on formulas and on judgements. The translation maps variables to themselves, commutes with all \(\mu\) constructs, and with the command building construct. The remaining cases are:

\[
\begin{align*}
[(t_1, t_2)]_{foc} &= \mu x^-.([t_2]_{foc} \mid \mu x_2.([t_1]_{foc} \mid \mu x_1.((x_1, x_2) \mid x^-))) \\
[inl(t_1)]_{foc} &= \mu x^-.([t_1]_{foc} \mid \mu x_1.(inl(x_1) \mid x^-)) \\
[t!]_{foc} &= \mu y^+.([t]_{foc} \mid \mu x.(y^+ \mid x!))
\end{align*}
\]

Note the (arbitrary) choice of order of evaluation in the first rule.

The translation introduces cuts, which are then eliminated by cut-elimination. Therefore, every provable sequent of LL (possibly with cuts) admits a cut-free focalised proof (Andreoli). The translation achieves the most important part of the job of CPS translations, which is to fix an order of evaluation!
III) Indirect style

1. \( LL \downarrow \)
2. Translation into (a subset \( TL \)) of tensor logic
3. \( LKQ \) (monolateral presented as bilateral, positively)
4. \( LK \downarrow \) (bilateral, distinguishing lazy programs from contexts of positive type)
5. Levi’s CBPV
III.1) Linear logic

(focalised, with shifts)
Focalised syntax in indirect style for $\text{LL}$

We move from polarised formulas to polarised connectives: we now force positive connectives to have positive formulas as arguments, and the same for negative connectives.

For achieving this, we need two new connectives, which crystallised in ludics and game semantics (Girard, Laurent): the shifts (for which one may also have a monadic reading, as we shall see, whence the title of this slide).

We shall call the resulting logic $\text{LL}_\downarrow$. 
Syntax for $\text{LL}_{\downarrow}$

Formulas:

$$P ::= X \mid P \otimes P \mid P \oplus P \mid !N \mid \downarrow N \quad N ::= \overline{X} \mid N \otimes N \mid N \& N \mid ?P \mid \uparrow P$$

We still have the same four kinds of judgements:

<table>
<thead>
<tr>
<th>Commands</th>
<th>Values</th>
<th>Positive terms</th>
<th>Negative terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c : (\vdash \Gamma)$</td>
<td>$\vdash V^+ : P \mid \Gamma$</td>
<td>$\vdash t^+ : P \mid \Gamma$</td>
<td>$\vdash t^- : N \mid \Gamma$</td>
</tr>
</tbody>
</table>

Terms:

$$c ::= \langle t^+ | t^- \rangle$$

$$V^+ ::= x^+ \mid (V_1^+, V_2^+) \mid \text{inl}(V^+) \mid \text{inr}(V^+) \mid \mu(x^+) !.c \mid (t^-)_{\downarrow}$$

$$t^+ ::= V^+ \mid \mu x^-.c$$

$$t^- ::= x^- \mid \mu x^+.c \mid \mu(x_1^+, x_2^+).c \mid \mu[\text{inl}(x_1^+).c_1, \text{inr}(x_2^+).c_2] \mid (V^+)_{\downarrow} \mid \mu(x^-)_{\downarrow}.c$$
Typing rules for $\mathbb{LL}_{\downarrow}$

$\frac{}{\vdash x^+ : P ; x^+ : \overline{P}} \quad \frac{}{\vdash x^- : N \mid x^- : \overline{N}} \quad \frac{\vdash t^+ : P \mid \Gamma \quad \vdash t^- : \overline{P} \mid \Delta}{\langle t^+ \mid t^- \rangle : (\vdash \Gamma, \Delta)}$

$\frac{\vdash V^+ : P ; \Gamma}{\vdash V^+ : P \mid \Gamma} \quad \frac{c : (\vdash x^- : P, \Gamma)}{\vdash \mu x^- . c : P \mid \Gamma} \quad \frac{c : (\vdash x^+ : N, \Gamma)}{\vdash \mu x^+ . c : N \mid \Gamma}$

$\frac{\vdash V_1^+ : P_1 \mid \Gamma}{} \quad \frac{\vdash V_2^+ : P_2 ; \Delta}{\vdash (V_1^+ , V_2^+) : P_1 \otimes P_2 ; \Gamma, \Delta} \quad \frac{\vdash V_1^+ : P_1 \mid \Gamma}{\vdash \text{inl}(V_1^+) : P_1 \oplus P_2 ; \Gamma}$

$\frac{c : (\vdash x_1^+ : N_1 , x_2^+ : N_2 , \Gamma)}{\vdash \mu(x_1^+ , x_2^+) . c : N_1 \& N_2 \mid \Gamma} \quad \frac{c_1 : (\vdash x_1^+ : N_1 , \Gamma) \quad c_2 : (\vdash x_2^+ : N_2 , \Gamma)}{\vdash \mu[\text{inl}(x_1^+) . c_1 , \text{inr}(x_2^+) . c_2] : N_1 \& N_2 \mid \Gamma}$

$\frac{c : (\vdash x^+ : N, ?\Gamma)}{\vdash \mu(x^+) ! . c : !N ; ?\Gamma} \quad \frac{\vdash V^+ : P ; \Gamma}{\vdash (V^+) ! : ?P \mid \Gamma} \quad \frac{\vdash t^- : N \mid \Gamma}{\vdash (t^-) \downarrow : \downarrow N ; \Gamma} \quad \frac{c : (\vdash x^- : P, \Gamma)}{\vdash \mu(x^-) \downarrow . c : \uparrow P \mid \Gamma}$
Reduction rules for $LL\downarrow$

\[
\langle V^+ | \mu x^+.c \rangle \rightarrow c[V^+/x^+]
\]
\[
\langle \mu x^-.c | t^- \rangle \rightarrow c[t^-/x^-]
\]
\[
\langle (V_1^+, V_2^+) | \mu (x_1^+, x_2^+).c \rangle \rightarrow c[V_1^+/x_1^+, V_2^+/x_2^+]
\]
\[
\langle inl(V_1^+) | \mu [inl(x_1^+).c_1, inr(x_2^+).c_2] \rangle \rightarrow c_1[V_1^+/x_1^+]
\]
\[
\langle \mu(x^+)!.c | (V^+)! \rangle \rightarrow c[V^+/x^+]
\]
\[
\langle (t^-)\downarrow | \mu (x^-)\downarrow .c \rangle \rightarrow c[t^-/x^-]
\]
Decomposing the exponentials: \( !N = \downarrow \# N \) ... 

Confronting the rules for the exponentials above with the rules for shifts, one may be tempted by the following decomposition: \( !N = \downarrow \# N \) with the following syntax of formulas: 
\[
P ::= \ldots \mid \flat! P \quad N ::= \ldots \mid \flat\# N
\]
(with \ldots as before, minus the “normal” exponentials \(!\) and \(?\)), and with the following rules:
\[
\begin{align*}
\Gamma & \vdash x^+ : N, \flat\Gamma \\
\Gamma & \vdash \mu(x^+)^\flat . c : \# N \mid \flat\Gamma \\
\Gamma & \vdash V^+ : P ; \Gamma \\
\Gamma & \vdash (V^+)^\flat : \flat P ; \Gamma \\
\langle (V^+)^\flat \mid \mu(x^+)^\flat . c \rangle & \rightarrow c[V^+ / x^+]
\end{align*}
\]

This decomposition of the exponential modality appeared also in works on proof nets and on light linear logic (Girard and/or folklore). These rules make good sense in terms of focalised proof search (dereliction is irreversible, promotion is reversible). But the first typing rule with its side condition involving the old ? after all does not make it really convincing.
... or the other way around: $!N = !^+ \downarrow N$

One can also decompose “of course” (as in tensor logic) as $!N = !^+ \downarrow N$:

$$P ::= \ldots | !^+ P \quad N ::= \ldots | ?^- N$$

with the following rules:

$$c : (\vdash x^- : P, ?^- \Gamma)$$

$$\vdash \mu(x^-)^!^+.c : !^+ P ; ?^- \Gamma$$

$$\vdash t^- : N | \Gamma$$

$$\vdash (t^-)^!^+ : ?^- N | \Gamma$$

$$\langle \mu(x^-)^!^+.c \mid (t^-)^!^+ \rangle \to c[t^- / x^-]$$

It is easy to see that conversely, if one keeps $!, ?$ as primitive and if one defines $!^+ P = ! \uparrow P$ and $?^-$ dually, then the above rules are derivable.
Translating focalised \( \text{LL} \) into \( \text{LL} \downarrow \) (types)

Translation of types (the translation goes the same for \( \oplus \) as for \( \otimes \) and the same for \( \& \) as for \( \& \)):

\[
\begin{align*}
[[X]] \downarrow &= X \\
[[P_1 \otimes P_2]] \downarrow &= [[P_1]] \downarrow \otimes [[P_2]] \downarrow \\
[[N_1 \otimes P_2]] \downarrow &= \downarrow[[N_1]] \downarrow \otimes [[P_2]] \downarrow \\
[[P_1 \otimes N_2]] \downarrow &= [[P_1]] \downarrow \otimes \downarrow[[N_2]] \downarrow \\
[[N_1 \otimes N_2]] \downarrow &= \downarrow[[N_1]] \downarrow \otimes \downarrow[[N_2]] \downarrow \\
[[!P]] \downarrow &= !\uparrow[[P]] \downarrow \\
[[!N]] \downarrow &= ![[N]] \downarrow \\
[[X]] \downarrow &= \overline{X} \\
[[N_1 \& N_2]] \downarrow &= [[N_1]] \downarrow \& [[N_2]] \downarrow \\
[[N_1 \& P_2]] \downarrow &= [[N_1]] \downarrow \& \uparrow[[P_2]] \downarrow \\
\vdots \downarrow \\
[[?P]] \downarrow &= ?[[P]] \downarrow
\end{align*}
\]
Translating focalised $\mathsf{LL}$ into $\mathsf{LL}_\downarrow$ (terms)

Variables are translated to themselves. We give only the cases where the translation does not commute with the constructors:

$$\llbracket (t_1^-, V_2^+) \rrbracket_\downarrow = (\llbracket t_1^- \rrbracket_\downarrow, \llbracket V_2^+ \rrbracket_\downarrow) \quad \text{(idem for } \llbracket (V_1^+, t_2^-) \rrbracket_\downarrow)$$

$$\llbracket (t_1^-, t_2^-) \rrbracket_\downarrow = (\llbracket t_1^- \rrbracket_\downarrow, \llbracket t_2^- \rrbracket_\downarrow)$$

$$\llbracket \text{inl}(t^-) \rrbracket_\downarrow = \text{inl}(\llbracket t^- \rrbracket_\downarrow)$$

$$\llbracket \mu(x^-)!c \rrbracket_\downarrow = \mu(y^+)!\langle y^+ | \mu(x^-)^.\llbracket c \rrbracket_\downarrow \rangle$$

$$\llbracket \mu(x_1^-, x_2^+)c \rrbracket_\downarrow = \mu(y_1^+, x_2^+).\langle y_1^+ | \mu(x_1^-)^.\llbracket c \rrbracket_\downarrow \rangle \quad \text{(idem for } \llbracket \mu(x_1^+, x_2^-).c \rrbracket_\downarrow)$$

$$\llbracket \mu(x_1^-, x_2^-).c \rrbracket_\downarrow = \mu(y_1^+, y_2^+).\langle y_2^+ | \mu(x_2^-)^.\langle y_1^+ | \mu(x_1^-)^.\llbracket c \rrbracket_\downarrow \rangle \rangle$$

$$\llbracket \mu[\text{inl}(x_1^-).c_1, \text{inr}(x_2^+).c_2] \rrbracket_\downarrow = \mu[\text{inl}(y_1^+).\langle y_1^+ | \mu(x_1^-)^.\llbracket c_1 \rrbracket_\downarrow, \text{inr}(x_2^+).c_2]$$

$$\llbracket (t^-)! \rrbracket_\downarrow = (\llbracket t^- \rrbracket_\downarrow)!$$

The translation is reduction-preserving.
III.2) Tensor logic
Translating into Melliès’ tensor logic

Morally, tensor logic is the intuitionistic restriction of $\text{LL}_\downarrow$, where sequents admit at most one positive formula. More precisely, we shall consider a subsystem of tensor logic. The formulas are:

$$P ::= X | P \otimes P | P \oplus P | !^+ P | \downarrow N$$

$$N ::= \overline{X} | N \otimes N | N \& N | ?^- N | \uparrow P$$

There are only three kinds of judgements ($\Gamma$ consists of negative formulas only):

<table>
<thead>
<tr>
<th>Commands</th>
<th>Values</th>
<th>Negative terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>$c : (\vdash \Gamma)$</td>
<td>$\vdash V^+ : P ; \Gamma$</td>
<td>$\vdash t^- : N</td>
</tr>
</tbody>
</table>

But we have to move to different rules for the shifts and (polarity-keeping) exponentials: there is no space anymore to form $\mu(x^-)^{!^+}$ and $\mu(x^-)^{\downarrow}.c$ and (see also the discussion on syntactic adjunctions below).

The relation of $\text{LL}_\downarrow$ to tensor logic is the same as the relation of $\text{LKQ}$ to Laurent’s LLP.
Syntax for TL

Terms:

\[ c ::= \langle V^+ \mid t^- \rangle \]
\[ V^+ ::= x^+ \mid (V_1^+, V_2^+) \mid inl(V^+) \mid inr(V^+) \mid (V^+)^! \mid \mu(x^+)^\downarrow.c \]
\[ t^- ::= \mu x^+.c \mid \mu(x_1^+, x_2^+).c \mid \mu[\text{inl}(x_1^+).c_1, \text{inr}(x_2^+).c_2] \mid \mu(x^+)^! .c \mid (V^+)^\downarrow \]
Typing rules for TL

\[
\begin{align*}
\vdash x^+ : P ; x^+ : \overline{P} & \quad \vdash \mu x^+.c : N | \Gamma \\
\vdash V_1^+ : P_1 ; \Gamma_1 & \quad \vdash V_2^+ : P_2 ; \Gamma_2 \\
\vdash (V_1^+, V_2^+) : P_1 \otimes P_2 ; \Gamma_1, \Gamma_2 \\
c : (\vdash x_1^+ : N_1, x_2^+ : N_2, \Gamma) & \quad c_1 : (\vdash x_1^+ : N_1, \Gamma) \quad c_2 : (\vdash x_2^+ : N_2, \Gamma) \\
\vdash \mu(x_1^+, x_2^+).c : N_1 \otimes N_2 | \Gamma \\
\vdash V^+ : P ; \Gamma & \quad \vdash \mu(x^+).c : \downarrow N | \Gamma \\
\vdash (V^+)^\downarrow : \downarrow P ; \Gamma \\
c : (\vdash x^+ : N, \Gamma) & \quad \vdash V^+ : P ; \Gamma \\
\vdash \mu(x^+).c : \downarrow N ; \Gamma \\
\vdash (V^+)^\downarrow : \uparrow P | \Gamma \\
\end{align*}
\]

\[\langle V^+ | t^- \rangle : (\vdash \Gamma_1, \Gamma_2)\]
Reduction rules for TL

$$\langle V^+ \mid \mu x^+.c \rangle \rightarrow c[V^+/x^+]$$
$$\langle (V_1^+, V_2^+) \mid \mu (x_1^+, x_2^+).c \rangle \rightarrow c[V_1^+/x_1^+, V_2^+/x_2^+]$$
$$\langle inl(V_1^+) \mid \mu [inl(x_1^+).c_1, inr(x_2^+).c_2]\rangle \rightarrow c_1[V_1^+/x_1^+]$$
$$\langle (V^+)^!^+ \mid \mu (x^+)^!^+.c \rangle \rightarrow c[V^+/x^+]$$
$$\langle \mu (x^+)\downarrow.c \mid (V^+)\downarrow \rangle \rightarrow c[V^+/x^+]$$

Melliès authorises (at most) one positive formula in the non-value judgements. But the target of our translation does not need this liberality.
Translating $\text{LL} \downarrow$ to $\text{TL}$

Expand $!N$ as $!^+ \downarrow N$.

For judgements ($\Gamma$ negative, $\Delta = x^- : P, \ldots$ positive, $\uparrow \Delta = k^+_x : \uparrow P, \ldots$):

$c : (\vdash \Gamma, \Delta) \vdash V^+ : P ; \Gamma, \Delta \vdash t^+ : P | \Gamma, \Delta \vdash t^- : N | \Gamma, \Delta$

$\llbracket c \rrbracket_{\text{TL}} : (\vdash \Gamma, \uparrow \Delta) \vdash \llbracket V^+ : P \rrbracket_{\text{TL}} ; \Gamma, \uparrow \Delta \vdash \llbracket t^+ \rrbracket_{\text{TL}} : \uparrow P | \Gamma, \uparrow \Delta \vdash \llbracket t^- \rrbracket_{\text{TL}} : N | \Gamma, \uparrow \Delta$

The only cases where the translation does not commute with the constructors are the following:

\[
\begin{align*}
\llbracket x^- \rrbracket_{\text{TL}} &= \mu y^+.\llbracket k^+_x | (y^+) \downarrow \rrbracket \\
\llbracket (t^+ | t^-) \rrbracket_{\text{TL}} &= \langle \mu(x^+) \downarrow . \llbracket x^+ | \llbracket t^- \rrbracket_{\text{TL}} \rrbracket | \llbracket t^+ \rrbracket_{\text{TL}} \rangle \\
\llbracket t^+ \rrbracket_{\text{TL}} &= \begin{cases} 
\llbracket V^+ \rrbracket_{\text{TL}}_{\text{TL}} & \text{if } t^+ = V^+ \\
\mu k^+_x . \llbracket c \rrbracket_{\text{TL}} & \text{if } t^+ = \mu x^- . c 
\end{cases} \\
\llbracket (\mu(x^+)^! . c) \rrbracket_{\text{TL}} &= (\mu(x^+) \downarrow . \llbracket c \rrbracket_{\text{TL}})^! \\
\llbracket (V^+)^! \rrbracket_{\text{TL}} &= \mu (y^+)^! . \langle y^+ | (\llbracket V^+ \rrbracket_{\text{TL}})^! \rangle \\
\llbracket (t^-)^! \rrbracket_{\text{TL}} &= \mu (x^+) \downarrow . \langle y^+ | t^- \rangle \\
\llbracket (\mu(x^-)^! . c) \rrbracket_{\text{TL}} &= \mu k^+_x . \llbracket c \rrbracket_{\text{TL}}
\end{align*}
\]
III.3) Classical logic

(focalised, with shifts, “monolateral”)
Mono-sided versus homogeneous two-sided

We can turn mono-sided sequents ⊢ ..., Pᵢ, ..., Nⱼ, ... into mono-polarised two-sided sequents:

... , \overline{Nⱼ} , ... ⊢ ..., Pᵢ, ... (positive presentation)
...
... , \overline{Pᵢ} , ... ⊢ ..., Nⱼ, ... (negative presentation)

We shall do this for the indirect style version of (monolateral) LK\textsubscript{pol}. The resulting positive presentation is known as LKQ (terminology going back to Danos et al.). In this presentation, one “sees” only positive formulas:

\[ P ::= X \mid P \otimes P \mid P \oplus P \mid \neg^+ P \]

where \( \neg^+ P \) stands for \( \downarrow \overline{P} \).
Two flavours of negation

We should not confuse:

– The involutive negation $\neg$ given implicitly (as in LL) de Morgan duality, or explicitly if we allow two-polarised two-sided sequents

$$\ldots, P_i, \ldots, N_j, \ldots \vdash \ldots, P_k, \ldots, N_l, \ldots$$

from which we can infer, say, $\ldots, P_i, \ldots, \ldots \vdash \neg N_j, \ldots, P_k, \ldots, N_l, \ldots$
(we’ll come to such sequents later).

– The negations $\neg^+ P$, and dually $\neg^- N$, which are the ones arising implicitly in call-by-value and call-by-name $\lambda$-calculus, as we shall see.
Two notions of symmetry

In the mono-polarised two-sided sequents, we assimilate “left” with “negative” and “right” with “positive”.

But from the programming point of view they should be kept distinct:

the duality left-right corresponds to the duality input - output
the duality positive-negative corresponds the duality eager-lazy

We can restore the distinction by going to two-polarised two-sided sequents. We shall illustrate this later with CBPV.

For the time being, we remain “monolateral”, in the guise of homogeneous positive two-sided.
Preparation for the syntax and rules of LKQ

Assumptions $x^- : P$ (resp. $x^+ : N$) are renamed as $\alpha : P$ on the right (resp. as $x : P$ on the left), in the tradition of (CBV) $\lambda\mu$-calculus.

Negative terms $t^-$ are now positive contexts $e$. The $\mu$’s on the left are now $\tilde{\mu}$ (as in Curien and Herbelin’s duality of computation paper).

Also, for the rest of this talk, we push weakening to the axioms, we merge cut and contraction in an additive contraction rule, and give an additive formulation of the right tensor rule. The resulting system has no explicit weakening nor contraction rule.
Syntax for LKQ

Formulas:

\[ P ::= X \mid P \otimes P \mid P \oplus P \mid \neg^+ P \]

Typing judgements:

<table>
<thead>
<tr>
<th>Commands</th>
<th>Values</th>
<th>Expressions</th>
<th>Contexts</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c : (\Gamma \vdash \Delta) )</td>
<td>( \Gamma \vdash V^+ : P ; \Delta )</td>
<td>( \Gamma \vdash v : P</td>
<td>\Delta )</td>
</tr>
</tbody>
</table>

Terms:

<table>
<thead>
<tr>
<th>Commands</th>
<th>Values</th>
<th>Expressions</th>
<th>Contexts</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c ::= \langle v</td>
<td>e \rangle )</td>
<td>( v ::= V^+</td>
<td>\mu \alpha . c )</td>
</tr>
</tbody>
</table>

Reductions rules: first 4 rules as for LLL↓+

\[ \langle e^* | \tilde{\mu} \alpha^*.c \rangle \rightarrow c[e/\alpha] \]
Typing rules for LKQ

\[\begin{align*}
\Gamma, x : P & \vdash x : P; \Delta \\
\Gamma \alpha : P & \vdash \alpha : P, \Delta \\
\Gamma & \vdash v : P | \Delta \quad \Gamma | e : P \vdash \Delta \\
\langle v \mid e \rangle & : (\Gamma \vdash \Delta)
\end{align*}\]

\[\begin{align*}
c : (\Gamma, x : P \vdash \Delta) & \\
\Gamma | \tilde{\mu}x.c & : P \vdash \Delta \\
\Gamma & \vdash \mu \alpha.c : P | \Delta \\
\Gamma & \vdash v : P; \Delta \\
\Gamma & \vdash v : P | \Delta
\end{align*}\]

\[\begin{align*}
\Gamma & \vdash V_1^+ : P_1; \Delta \\
\Gamma & \vdash V_2^+ : P_2; \Delta \\
\Gamma & \vdash (V_1^+, V_2^+) : P_1 \otimes P_2; \Delta \\
\Gamma & \vdash \text{inl}(V_1^+) : P_1 \oplus P_2; \Delta
\end{align*}\]

\[\begin{align*}
c & : (\Gamma, x_1 : P_1, x_2 : P_2 \vdash \Delta) \\
\Gamma | \tilde{\mu}(x_1, x_2).c & : P_1 \otimes P_2 \vdash \Delta \\
c_1 & : (\Gamma, x_1 : P_1 \vdash \Delta) \\
\Gamma | \tilde{\mu}[\text{inl}(x_1).c_1|\text{inr}(x_2).c_2] & : P_1 \oplus P_2 \vdash \Delta
\end{align*}\]

\[\begin{align*}
c & : (\Gamma \vdash \alpha : P, \Delta) \\
\Gamma | \tilde{\mu}\alpha^\cdot.c & : \neg^+ P \vdash \Delta \\
\Gamma & \vdash e : P \vdash \Delta
\end{align*}\]

\[\begin{align*}
\Gamma & \vdash e^\cdot : \neg^+ P; \Delta
\end{align*}\]
Encoding call-by-value and call-by-name $\lambda$-calculus

One can encode call-by-value implication as follows in LKQ:

$$P \rightarrow^v Q = \neg^+ (P \otimes (\neg^+ Q))$$

Then $\lambda$-terms are translated to expressions (and $\lambda$-abstractions to values)

Call-by-name $\lambda$-calculus can be translated in LKT, which is the negative two-sided presentation of mono-sided $\text{LK}_{pol}$ (the mirror image of LKQ), through the following encoding of call-by-name implication:

$$M \rightarrow^n N = (\neg^- M) \circ N$$

These translations extend straightforwardly to the CBV and CBN $\lambda\mu$-calculi.
III.4) Classical logic
(focalised, with shifts, bilateral)
Bilaterality

We now revert to the $\downarrow$ and $\uparrow$ connectives, and allow for mixed contexts on the left and on the right: in sequents, $\Gamma$ stands for $\ldots, x^+: P, \ldots, x^-: N, \ldots$, and $\Delta$ stands for $\ldots, \alpha^+: P, \ldots, \alpha^-: N, \ldots$.

$$P ::= X | P \otimes Q | P \oplus Q | \neg N | \downarrow N$$

$$N ::= \overline{X} | N \& N | N \& N | \neg P | \uparrow P$$

$$A ::= P | N$$

(we also added an explicit involutive negation $\neg$). We call this system (the last but one in this talk) $\text{LK}_\downarrow$. It extends both $\text{LKQ}$ and $\text{LKT}$.

There are now five kinds of judgements (we’ll stop there, don’t worry!):

- Commands
- Values
- Expressions
- Covalues
- Contexts

$c: (\Gamma \vdash \Delta) \quad \Gamma \vdash V^+: P ; \Delta \quad \Gamma \vdash v: A | \Delta \quad \Gamma ; E^- : N \vdash \Gamma \quad \Gamma | e : A \vdash \Gamma$

(We could have done this extension keeping shifts implicit, cf. Munch’s bilateral version of $\text{LK}_{pol}$.)
Syntax for $\text{LK}\downarrow$

(For the rest of the talk, we write $V, E$ rather than $V^+, E^-$, for short)

**Commands**

\[ c ::= \langle v^+ | e^+ \rangle \mid \langle v^- | e^- \rangle \]

**Expressions**

\[ v^+ ::= V \mid \mu\alpha^+.c \]
\[ v^- ::= x^- \mid \mu\alpha^- .c \mid \mu(\alpha^+)\uparrow .c \]
\[ \quad \mid \mu[\alpha^-_1, \alpha^-_2].c \mid \mu(\alpha^-_1[fst].c_1, \alpha^-_2[snd].c_2) \mid \mu(x^+)\neg .c \]

**Values**

\[ V ::= x^+ \mid (V, V) \mid \text{inl}(V) \mid \text{inr}(V) \mid (v^-)\downarrow \]

**Contexts**

\[ e^- ::= E \mid \tilde{\mu}x^- .c \]
\[ e^+ ::= \alpha^+ \mid \tilde{\mu}x^+ .c \mid \tilde{\mu}(x^-)\downarrow .c \]
\[ \quad \mid \tilde{\mu}(x^+, y^+) .c \mid \tilde{\mu}[\text{inl}(x^+_1).c_1, \text{inr}(x^+_2).c_2] \]

**Covalues**

\[ E ::= \alpha^- \mid [E, E] \mid E[fst] \mid E[snd] \mid (e^+)\uparrow \mid V^- \]

We can factorise a few rules using the following mergings:

\[ v ::= v^+ \mid v^- \quad \alpha ::= \alpha^+ \mid \alpha^- \quad e ::= e^+ \mid e^- \quad x ::= x^+ \mid x^- \]
**Typing rules for $\text{LK}_\downarrow$**

<table>
<thead>
<tr>
<th>Rule</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Gamma, x^+ : P \vdash x^+ : P; \Delta$</td>
<td>$\Gamma \vdash x^+ : P \vdash x^+ : P, \Delta$</td>
</tr>
<tr>
<td>$\Gamma \vdash v : A</td>
<td>\Delta$</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>$(v \mid e) : (\Gamma \vdash \Delta)$</td>
<td>$\Gamma \vdash \mu\alpha.c : A</td>
</tr>
<tr>
<td>$\Gamma \vdash \nu^\alpha.c : A \vdash \Delta$</td>
<td>$\Gamma \vdash \nu^\alpha.c : A \vdash \Delta$</td>
</tr>
<tr>
<td>$\Gamma \vdash (\nu^-) : \downarrow N; \Delta$</td>
<td>$\Gamma \vdash (V_1, V_2) : P_1 \otimes P_2; \Delta$</td>
</tr>
<tr>
<td>$\Gamma \vdash (V^-) \downarrow : \downarrow N; \Delta$</td>
<td>$\Gamma \vdash \mu(\alpha^+) \uparrow.c : \uparrow P; \Delta$</td>
</tr>
<tr>
<td>$c : (\Gamma \vdash \alpha^+ : P, \Delta)$</td>
<td>$c : (\Gamma \vdash \alpha^- : N_1, \alpha^- : N_2, \Delta)$</td>
</tr>
<tr>
<td>$\Gamma \vdash \mu(\alpha^+ \uparrow.c : \uparrow P) : \Delta$</td>
<td>$\Gamma \vdash [E_1, E_2] : N_1 &amp; N_2 \vdash \Delta$</td>
</tr>
<tr>
<td>$\Gamma \vdash \mu([\alpha_1, \alpha_2].c : N_1 &amp; N_2) : \Delta$</td>
<td></td>
</tr>
<tr>
<td>$\Gamma \vdash [E_1, E_2] : N_1 &amp; N_2 \vdash \Delta$</td>
<td></td>
</tr>
<tr>
<td>$\Gamma ; E_1 : N_1 \vdash \Delta$</td>
<td>$\Gamma ; E_2 : N_2 \vdash \Delta$</td>
</tr>
<tr>
<td>$\Gamma ; e^\uparrow : \uparrow P \vdash \Delta$</td>
<td>$\Gamma \vdash [E_1, E_2] : N_1 &amp; N_2 \vdash \Delta$</td>
</tr>
<tr>
<td>$c : (\Gamma, x^- : N \vdash \Delta)$</td>
<td>$c : (\Gamma, x_1^+ : P_1, x_2^+ : P_2 \vdash \Delta)$</td>
</tr>
<tr>
<td>$\Gamma \vdash \mu(x^-) \downarrow.c : \downarrow N \vdash \Delta$</td>
<td>$c_2 : (\Gamma, x_2^+ : P_2 \vdash \Delta)$</td>
</tr>
<tr>
<td>$\Gamma \vdash \mu(x^+) \uparrow.c : \uparrow N \vdash \Delta$</td>
<td>$c_1 : (\Gamma, x_1^+ : P_1 \vdash \Delta)$</td>
</tr>
<tr>
<td>$\Gamma \vdash \mu([\text{inl}(x_1^+).c_1, \text{inr}(x_2^+).c_2] : P_1 \oplus P_2 \vdash \Delta$</td>
<td>$c_2 : (\Gamma, x_2^+ : P_2 \vdash \Delta)$</td>
</tr>
</tbody>
</table>
Reduction rules for \( \mathbb{LK}_\downarrow \)

\[
\langle V \mid \tilde{\mu}x^+.c \rangle \rightarrow c[V/x^+]
\]
\[
\langle \mu\alpha^- .c \mid E \rangle \rightarrow c[E/\alpha^-]
\]
\[
\langle v^- \mid \tilde{\mu}x^-.c \rangle \rightarrow c[v^-/x^-]
\]
\[
\langle \mu\alpha^+.c \mid e^+ \rangle \rightarrow c[e^+/\alpha^+]
\]
\[
\langle (V_1, V_2) \mid \tilde{\mu}(x_1^+, x_2^+).c \rangle \rightarrow c[V_1/x_1^+, V_2/x_2^+]
\]
\[
\langle \mu[\alpha^-_1, \alpha^-_2].c \mid [E_1, E_2] \rangle \rightarrow c[E_1/\alpha^-_1, E_2/\alpha^-_2]
\]
\[
\langle \text{inl}(V_1) \mid \tilde{\mu}[\text{inl}(x_1^+).c_1, \text{inr}(x_2^+).c_2]\rangle \rightarrow c_1[V_1/x_1^+]
\]
\[
\langle \mu(\alpha^-_1[fst].c_1, \alpha^-_2[snd].c_2) \mid E_1[fst] \rangle \rightarrow c_1[E_1/\alpha^-_1]
\]
\[
\langle v^-\downarrow \mid \tilde{\mu}(x^-)^\downarrow .c \rangle \rightarrow c[v^-/x^-]
\]
\[
\langle \mu(\alpha^+)^\uparrow .c \mid e^+\uparrow \rangle \rightarrow c[e^+/\alpha^+]
\]
\[
\langle \tilde{\mu}(x^+)^-.c \mid V^- \rangle \rightarrow c[V/x^+]
\]
Two syntactic adjunctions

We have, in LL↓ as well as in LK↓:

↓ − ↑ at the level of positive contexts and negative terms
↑ − ↓ at the level of covalues and values

As we shall see, the first adjunction is more primitive than the second.

The adjunctions are mediated by command judgements. We exhibit the inverse syntactic isomorphisms. We need two $\eta$-rules (which express invertibility):

\[
v^- = \mu \alpha^+ \cdot \langle v^- \mid \alpha^+ \rangle \quad \text{(for } \Gamma \vdash v^- : \uparrow P \mid \Delta)\]
\[
e^+ = \tilde{\mu} x^- \cdot \langle x^- \mid e^+ \rangle \quad \text{(for } \Gamma \mid e^+ : \downarrow N \vdash \Delta)\]
The first syntactic adjunction

We have

\[ \Gamma \vdash v^\rightarrow : \uparrow P \mid \Delta \]
\[ \uparrow \]
\[ c : (\Gamma \vdash \alpha^+ : P, \Delta) \]
\[ \langle v^\rightarrow \mid \alpha^+ \rangle \]
\[ \uparrow \]
\[ \mu \alpha^{++}.c \]

and

\[ \Gamma \mid e^+ : \downarrow N \vdash \Delta \]
\[ \uparrow \]
\[ c : (\Gamma, x^\rightarrow : N \vdash \Delta) \]
\[ \langle x^\rightarrow \mid e^+ \rangle \]
\[ \uparrow \]
\[ \tilde{\mu} x^-.c \]

so that putting these isos together we obtain isos between

\[ \Gamma, x^\rightarrow : N \vdash v^\rightarrow : \uparrow P \mid \Delta , \ c : (\Gamma, x^\rightarrow : N \vdash \alpha^+ : P, \Delta) , \ \Gamma \mid e^+ : \downarrow N \vdash \alpha^+ : P, \Delta \]
Preparation for the second syntactic adjunction

We define macros:

\[
\begin{align*}
\mu\alpha \downarrow \cdot c &= (\mu\alpha \downarrow \cdot c)\downarrow \\
\tilde{\mu}x \uparrow \cdot c &= (\tilde{\mu}x \uparrow \cdot c)\uparrow
\end{align*}
\]

\[
\begin{align*}
E\downarrow &= \tilde{\mu}x \downarrow \cdot \langle x \mid E \rangle \\
V\uparrow &= \mu\alpha \uparrow \cdot \langle V \mid \alpha\uparrow \rangle
\end{align*}
\]

with the following derived typing rules:

\[
\frac{c : (\Gamma \vdash \alpha^- : N, \Delta)}{\Gamma \vdash \mu\alpha \downarrow \cdot c : \downarrow N \mid \Delta} \quad \frac{\Gamma ; E : N \vdash \Delta}{\Gamma \vdash E\downarrow : \downarrow N \vdash \Delta}
\]

\[
\frac{c : (\Gamma, x^+ : P \vdash \Delta)}{\Gamma ; \tilde{\mu}x \uparrow \cdot c : \uparrow P \vdash \Delta} \quad \frac{\Gamma \vdash V : P \mid \Delta}{\Gamma \vdash V\uparrow : \uparrow P \mid \Delta}
\]

We need two new \(\eta\)-rules, which are “by value” (cf. \(\lambda x.Vx\) in CBV \(\lambda\)-calculus):

\[
\begin{align*}
V &= \mu\alpha \downarrow \cdot \langle V \mid \alpha\downarrow \rangle \quad \text{(for } \Gamma \vdash V : \downarrow N \mid \Delta) \\
E &= \tilde{\mu}x \uparrow \cdot \langle x^+ \uparrow \mid E \rangle \quad \text{(for } \Gamma ; E : \uparrow P \vdash \Delta)\]
\]
Second syntactic adjunction

We have
\[ \Gamma ⊢ V : ↓N ; \Delta \]
\[ V \quad \mu \alpha \downarrow \cdot c \]
\[ c : (\Gamma ⊢ \alpha \downarrow : N, \Delta) \quad \langle V | \alpha \downarrow \rangle \quad c \]

and
\[ \Gamma ; E : ↑P ⊢ \Delta \]
\[ E \quad \tilde{\mu}x \downarrow \cdot c \]
\[ c : (\Gamma, x \downarrow : P ⊢ \Delta) \quad \langle x \downarrow | E \rangle \quad c \]

so that putting these isos together we obtain isos between
\[ \Gamma, x \downarrow : P ⊢ V : ↓N ; \Delta , \ c : (\Gamma, x \downarrow : P ⊢ \alpha \downarrow : N, \Delta) , \ Γ ; E : ↑P ⊢ \alpha \downarrow : N, \Delta \]

Conversely, taking the macros as primitive, we cannot recover the first adjunction.
III.5) Call-By-Push-Value
From $\text{LK}_\downarrow$ to CBPV (in sequent calculus style)

By cutting down $\text{LK}_{pol}$ to intuitionistic judgements of the respective forms

\begin{align*}
\text{Commands} & \quad \text{Values} & \quad \text{Expressions} \\
\Gamma \vdash v : N & \mid \Gamma \vdash [.] : N & \Gamma \vdash v : N \\
\end{align*}

\begin{align*}
\text{Covalues} & \quad \text{Contexts} & \\
\Gamma ; E : N_1 \vdash [.] ; N_2 & \quad \Gamma \mid e : P \vdash [.] : N \\
\end{align*}

we arrive to a sequent calculus discussed by Pfenning in his course notes on focalisation, and which is exactly a sequent calculus version of Levy’s CBPV.

This independent appearance witnesses the fact that as long as no precision is given on the “kind of effects” we want to model with the monad associated to the adjunction ($\uparrow \dashv \downarrow$ or $F \dashv U$, see dictionary below), we are just doing (intuitionistic) polarisation.
System L style syntax for CBPV

Formulas:

\[
P ::= P \oplus P \mid P \otimes P \mid P \oplus P \mid \downarrow N
\]
\[
N ::= N \& N \mid P \rightarrow N \mid \uparrow P
\]

Values

\[
V ::= x \mid (V, V) \mid \text{inl}(V) \mid \text{inr}(V) \mid \mu[.]\downarrow .c
\]

Negative terms

\[
v ::= \mu[.] .c \mid \mu([\text{fst}]. c_1, [\text{snd}]. c_2) \mid \mu[x \cdot [.]].c \mid V^\uparrow
\]

Covalues

\[
E ::= [.] \mid E[\text{fst}] \mid E[\text{snd}] \mid [V \cdot E] \mid \bar{\mu}x^\uparrow .c
\]

Positive contexts

\[
e ::= \bar{\mu}x .c \mid \bar{\mu}(x, y).c \mid \bar{\mu}[\text{inl}(x_1).c_1, \text{inr}(x_2).c_2] \mid E^\downarrow
\]

Commands

\[
c ::= \langle v \mid E \rangle \mid \langle V \mid e \rangle
\]

where \( \Gamma \) is a context of positive formulas.
Dictionary wrt to P.B. Levy’s notation

value computation
positive Σ × UN FP Π P → N
negative ⊕ ⊗ ↓N ↑P & P → N (= \(\overline{P} \otimes N\))

V (value) M (computation) K (stack)
V (value) v (negative term) E (covalue)

(no counterpart for e, c).
System $\mathbb{L}$ style typing rules for CBPV

\[
\begin{array}{llll}
\Gamma, x : P \vdash x : P; & \Gamma; . : N \vdash . : N & \Gamma \vdash v : N & \Gamma; . : N \vdash [.] : N' \\
\frac{c : (\Gamma \vdash . : N)}{\Gamma \vdash \mu[.].c : N} & \frac{\Gamma \vdash V_1 : P_1; \Gamma \vdash V_2 : P_2;}{\Gamma \vdash V \uparrow : \uparrow P} & \frac{\Gamma \vdash \mu[.].c : \downarrow N;}{\Gamma \vdash (V_1, V_2) : P_1 \otimes P_2;} & \frac{\Gamma \vdash \mu[.].c : P \vdash . : N}{\Gamma \vdash \mu([f_{st}.c_1, [s_{nd}].c_2]) : N_1 \& N_2} \\
\frac{\Gamma \vdash V : P; \Gamma \vdash V \uparrow : \uparrow P\uparrow}{\Gamma \vdash \mu[x \cdot .].c : P \rightarrow N} & \frac{\Gamma \vdash E : N \vdash . : N'}{\Gamma \vdash [V \cdot E] : P \rightarrow N \vdash [.] : N'} & \frac{\Gamma; E_1 : N_1 \vdash [.] : N}{\Gamma; E_1[f_{st}] : N_1 \& N_2 \vdash [.] : N} & \frac{\Gamma; E : N \vdash . : N'}{\Gamma; E_1[f_{st}] : N_1 \& N_2 \vdash [.] : N} \\
\frac{\Gamma \vdash E \downarrow : \downarrow N \vdash [.] : N'}{\Gamma \vdash \mu(x_1, x_2).c : P_1 \otimes P_2 \vdash [.] : N} & \frac{\Gamma \vdash \mu(x_1, x_2).c : P_1 \otimes P_2 \vdash [.] : N}{\Gamma \vdash \mu([i_{nl}c_1, i_{nr}c_2]) : P_1 \oplus P_2 \vdash [.] : N} & \frac{\Gamma; E : N \vdash . : N'}{\Gamma; E_1[f_{st}] : N_1 \& N_2 \vdash [.] : N} & \frac{\Gamma; E : N \vdash . : N'}{\Gamma; E_1[f_{st}] : N_1 \& N_2 \vdash [.] : N}
\end{array}
\]
Only the second adjunction is available for \text{CBPV}

What happens when cutting down to intuitionistic systems such as \text{CBPV}, \text{LLP}, or, in the linear case, \text{TL}, is that there is no space to express the first adjunction.

In the \text{CBPV} case: there are no sequents in which there is a variable $x$ of negative type in the (left) context, and similarly no sequents with a variable $\alpha$ of positive type on the right (only $[.] : N$ is available).
System $\perp$ style reduction rules for CBPV

\[
\begin{align*}
\langle V \mid \tilde{\mu}x.c \rangle & \rightarrow c[V/x] \\
\langle \mu[.]c \mid E \rangle & \rightarrow c[E/[]] \\
\langle (V_1, V_2) \mid \tilde{\mu}(x_1, x_2).c \rangle & \rightarrow c[V_1/x_1, V_2/x_2] \\
\langle \mu[x \cdot \alpha].c \mid [V, E] \rangle & \rightarrow c[V/x, E/\alpha] \\
\langle inl(V_1) \mid \tilde{\mu}[inl(x_1).c_1, inr(x_2).c_2] \rangle & \rightarrow c_1[V_1/x_1] \\
\langle \mu([fst].c_1, [snd].c_2) \mid E_1[fst] \rangle & \rightarrow c_1[E_1/[]] \\
\langle \mu[.].c \mid E \downarrow \rangle & \rightarrow c[E/[]] \\
\langle V \uparrow \mid \tilde{\mu}x \uparrow .c \rangle & \rightarrow c[V/x]
\end{align*}
\]
Translation from CBPV to L style

(read “let $V$ (resp. $v, v_1, E, \ldots$) be the translation of $V$ (resp $M, M_1, K, \ldots$)

$x$ \quad \Rightarrow \quad x$

return $V$ \quad \Rightarrow \quad $V$↑

thunk $M$ \quad \Rightarrow \quad $\mu[.]^\uparrow \langle v | [. \rangle$

$\Sigma$ introduction \quad \Rightarrow \quad $\text{inl, inr}$

$(V, V')$ \quad \Rightarrow \quad $(V, V')$

$\lambda \{1. M_1, 2. M_2\}$ \quad \Rightarrow \quad $\mu([\text{fst}]. \langle v_1 | [. \rangle, [\text{snd}]. \langle v_2 | [. \rangle)\rangle$

$\lambda x. M$ \quad \Rightarrow \quad $\mu[.]^\uparrow \langle v | [. \rangle$

let $V$ be $x. M$ \quad \Rightarrow \quad $\mu[.]^\uparrow \langle v_1 | [. \rangle, \mu[.]^\uparrow \langle v_2 | [. \rangle)\rangle$

$M_1$ to $x. M_2$ \quad \Rightarrow \quad $\mu[.]^\uparrow \langle v_1 | [. \rangle$

force $V$ \quad \Rightarrow \quad $\mu[.]^\uparrow \langle v | [. \rangle$

$\text{pm} V$ as $\{(1, x_1). M_1, (2, x_2). M_2\}$ \quad \Rightarrow \quad $\mu[.]^\uparrow \langle v_1 | [. \rangle, \mu[.]^\uparrow \langle v_2 | [. \rangle)\rangle$

$\text{pm} V$ as $\langle x, y \rangle. M$ \quad \Rightarrow \quad $\mu[.]^\uparrow \langle v | [. \rangle[fst]\rangle$

$\text{\hat{\beta}}^\uparrow M$ \quad \Rightarrow \quad $\mu[.]^\uparrow \langle v | [V \cdot [. \rangle\rangle$

$V^\prime M$ \quad \Rightarrow \quad $\mu[.]^\uparrow \langle v | E \rangle$

nil \quad \Rightarrow \quad $\text{idem} \hat{\beta}^\uparrow, \text{snd}$

$[.]$ to $x. M :: K$ \quad \Rightarrow \quad $[V \cdot E]$

$\text{id} :: K$ \quad \Rightarrow \quad $E[fst]$ (idem $\hat{\beta}^\uparrow, \text{snd}$)

$V :: K$
Translation from L style to CBPV

The three categories $e, c, v$ are translated to computations $M$, while $V, E$ of course translate to values and stacks. The translation of contexts $e$ is parameterised by a variable $x$ (the place-holder of $e$ in the sequent). The translation makes use of the dismantling $M \bullet K$ (or read-back) of a state $(M, K)$ as a computation.

\[
x^\dagger = x
\]
\[
(inl(V))^\dagger = (\hat{1}, V^\dagger) \quad \text{(idem} \: inr\text{)}
\]
\[
(V_1, V_2)^\dagger = ((V_1)^\dagger, (V_2)^\dagger)
\]
\[
(\mu[\cdot].c)^\dagger = \text{thunk} \: c^\dagger
\]
\[
(\mu[\cdot].c)^\dagger = c^\dagger
\]
\[
(\mu([\text{fst}].c_1, [\text{snd}].c_2))^\dagger = \lambda \{1.(c_1)^\dagger, 2.(c_2)^\dagger\}
\]
\[
(\mu[x \cdot \cdot].c)^\dagger = \lambda x.\: c^\dagger
\]
\[
(V^\dagger)^\dagger = \text{return} \: V^\dagger
\]

\[
[\cdot]^\dagger = \text{nil}
\]
\[
E[\text{fst}]^\dagger = \hat{1} :: E^\dagger \quad \text{(idem} \: \text{snd}\text{)}
\]
\[
[V \cdot E]^\dagger = V^\dagger :: E^\dagger
\]
\[
(\tilde{\mu}x^\dagger.c)^\dagger = [\cdot] \text{ to } x.\: c^\dagger :: \text{nil}
\]
\[
(\tilde{\mu}x^\dagger.x)^\dagger = c^\dagger
\]
\[
(\tilde{\mu}(x_1, x_2).c)^\dagger = \text{pm} \: x \: \text{as} \: (x_1, x_2).\: c^\dagger
\]
\[
(\tilde{\mu}[\text{inl}(x_1).c_1, \text{inr}(x_2).c_2])^\dagger_x = \text{pm} \: x \: \text{as} \: \{(1, x_1).(c_1)_x^\dagger, (2, x_2).(c_2)_x^\dagger\}
\]
\[
(E^\dagger)^\dagger_x = (\text{force} \: x) \cdot E^\dagger
\]
\[
\langle v \mid E \rangle^\dagger = v^\dagger \cdot E^\dagger
\]
\[
\langle V \mid e \rangle^\dagger = e^\dagger_x[V/x]
\]
Equivalence

One checks easily that the two systems simulate each other.

To get that the translations are inverse to each other, we need the $\eta$-rules (cf. above).

Of course, this is the old story of inter-translating natural deduction and sequent calculus, but the fact that the target of the translation of CBPV is exactly the projection of a larger, symmetric picture reinforces its relevance.

(Analogy : saying that a Böhm tree is a strategy is not that interesting, what is most interesting is to characterise the strategies arising in this way (innocence).)
What next?

We just ask two questions: what is the categorical structure of which System L would be the internal language? (We started to work on this with Marcelo Fiore.)

Does the highlighted connection between a metalanguage designed to discuss effects (Levy, in the line of Moggi) and a metalanguage to discuss classical logic help us to understand Filinski’s reflection result?