(A few small steps...)

Towards A Non-Commutative Logic of Effects

Noam Zeilberger

WORK-IN-PROGRESS

with

Jonas Frey & Paul-André Melliès

England, December 2010

What's up with Linear Logic?

After years of promise, why is linear logic still not a common language for talking about computation?

The leading suspects:

(1) Involutive negation ($A^{\perp\perp} = A$) ?

(2) Exchange ($A \otimes B = B \otimes A$) ?

I suggest (1) is harmless ultimately, but (2) is deadly:

<u>effects do not (in general) commute!</u>

Okay, so let's use
<u>non</u>-<u>commutative</u>
linear logic — problem solved?

We'd like to do better: the tools may already be there for understanding <u>why</u> this is a nice conceptual space, and how it relates to other nice spaces.

Main idea of this talk: it is possible to <u>derive</u> a non-commutative monoidal structure from an abstract duality between types & contexts.

Caveat: this is work in progress, X% baked for some X probably significantly below 50.

A Beautiful Idea

$\boxed{\text{a type is a set of values}}$

$\mathbb{B} = \{\text{true}, \text{false}\}$ $\quad \mathbb{N} = \{0, 1, 2, \ldots\}$

$\text{string} = \{\text{"foo"}, \text{"bar"}, \text{"baz"} \ldots\}$

$\boxed{\text{a map } A \longrightarrow B \text{ transforms } A\text{-values to } B\text{-values}}$

$\text{and} : \mathbb{B} \times \mathbb{B} \longrightarrow \mathbb{B}$
$\text{and}\,(\text{true}, \text{true}) = \text{true}$
$\text{and}\,(\text{true}, \text{false}) = \text{false}$
$\text{and}\,(\text{false}, \text{true}) = \text{false}$
$\text{and}\,(\text{false}, \text{false}) = \text{false}$

$\text{Length} : \text{string} \longrightarrow \mathbb{N}$
$\text{length "foo"} = 3$
$\text{length "bar"} = 3$
$\text{length "baz"} = 3$
$\vdots$

A Slightly Less Naive Idea

$\boxed{\text{a type is a \underline{presheaf} of values}}$

i.e., a family of sets of values $\{v \mid \Gamma \vdash v : P\}_\Gamma$, indexed by context, compatible with substitution:

$$\frac{\Gamma' \vdash \sigma \div \Gamma \qquad \Gamma \vdash v : P}{\Gamma' \vdash \sigma v : P}$$

i.e., a contravariant functor $P : \mathbb{C}^{op} \to \mathrm{Set}$ over some "category of contexts" $\mathbb{C}$.

Collectively, these types can be organized into the functor category $[\mathbb{C}^{op}, \mathrm{Set}]$, which we call $\hat{\mathbb{C}}$.

Another Beautiful Idea

$\boxed{\text{a type is a set of continuations}}$

Intuition: an object is defined by the observations you can make on it.

(e.g., functions are defined by their behaviour on application, pairs are defined by their projections)

$\boxed{\begin{array}{l}\text{a map } A \to B \text{ transforms} \\ \text{B-continuations to A-continuations}\end{array}}$

(some concrete Haskell code in a few slides)

And now less naively...

$\boxed{\text{a type is a \underline{covariant} presheaf of continuations}}$

i.e., a family of sets of continuations $\{k \mid N \overset{k}{\vdash} \Delta\}_\Delta$, indexed by "co-context" ("answer type"), compatible with postcomposition:

$$\frac{N \overset{k}{\vdash} \Delta \qquad \Delta \overset{\sigma}{\vdash} \Delta'}{N \overset{k\sigma}{\vdash} \Delta'}$$

i.e., a covariant functor $N : \mathbb{C} \to \mathrm{Set}$ over some "category of co-contexts" $\mathbb{C}$.

Collectively, these types can be organized into the functor category $[\mathbb{C}, \mathrm{Set}]^{op}$, which we call $\check{\mathbb{C}}$

An analogy

Polarity (in linear logic)

is like

"Isbell Duality"

Isbell Conjugacy

$$
\begin{tikzcd}
{[\mathbb{C}^{op}, \mathcal{V}] = \hat{\mathbb{C}}} \arrow[rr, bend left, "P \mapsto \lambda B.\, \hat{\mathbb{C}}(P, yB)"] \arrow[rr, phantom, "\perp"] & & {\check{\mathbb{C}} = [\mathbb{C}, \mathcal{V}]^{op}} \arrow[ll, bend left, "N \mapsto \lambda A.\, \check{\mathbb{C}}(\tilde{y}A, N)"] \\
& \mathbb{C} \arrow[ul, "y"] \arrow[ur, "\tilde{y}"'] &
\end{tikzcd}
$$

where $yB = \mathbb{C}(-, B)$
$\tilde{y}A = \mathbb{C}(A, -)$

An Equation

$$\boxed{\text{intvar} \stackrel{\text{def}}{=} \text{intexp} \,\&\, \text{intacc}}$$

John Reynolds, "Replacing Complexity with Generality: The Programming Language Forsythe"

An Equation

applying the Girard-Reynolds isomorphism...

$$\boxed{\mathrm{intvar} \stackrel{\mathrm{def}}{=} \,!(\mathrm{intexp} \;\&\; \mathrm{intacc})}$$

John Reynolds, "Replacing Complexity with Generality: The Programming Language Forsythe"

where $\mathrm{intexp} \stackrel{\mathrm{def}}{=} \uparrow \mathrm{int}$
$\mathrm{intacc} \stackrel{\mathrm{def}}{=} \mathrm{int} \multimap \uparrow 1$

Negative Types in Haskell

```haskell
data (&) n1 n2 x where
  P1 :: n1 x -> (n1 & n2) x
  P2 :: n2 x -> (n1 & n2) x

data (-o) p1 n2 x where
  App :: p1 -> n2 x -> (p1 -o n2) x

data ↑ p x where
  Plug :: (p ⇝ x) -> ↑ p x
```

```haskell
newtype Value n = Val {runVal ::
                         forall r. n r -> r}

plug :: n x -> Value n -> x
k `plug` v = runVal v k
```

Programming w/ Variables

```
data (!) n x where
    Return :: x → ! n x                     } "Free monad"
    Unfold :: n (! n x) → ! n x             }  construction

type Variable s = ! (↑s & (s ⊸ ↑()))

new :: s → Value (Variable s)
new s = Val new'
   where
    new' (Unfold (P1 (Plug ks))) = ks s `plug` new s
    new' (Unfold (P2 (App s' (Plug k)))) = k ()
                                             `plug` new s'
    new' (Return x) = x

example = Unfold $ P1 $ Plug $ \s →
          Unfold $ P2 $ App (s+2) $ Plug $ \() →
          Unfold $ P1 $ Plug $ \s' →
              Return (s * s')

fifteen = example `plug` new 3
```

(cf. sigfpe's "Programming with impossible functions")

Exercise: write reify :: Variable s x → (s → (x, s))
and reflect :: (s → (x, s)) → Variable s x

Types - as - Bimodules
("Profunctors", "Distributors")

Let $\mathcal{L}$ and $\mathcal{R}$ be categories related by a bimodule

$$\# : \mathcal{L}^{op} \times \mathcal{R} \longrightarrow \mathrm{Set}$$

(or more generally "$\mathcal{V}$")

A <u>positive</u> type P is an $(\mathcal{L}, \mathcal{R})$-bimodule which "represents" a "set of values", i.e.

$$\begin{aligned} P \in [\mathcal{L}^{op} \times \mathcal{R}, \mathrm{Set}] &= [\mathcal{R} \times \mathcal{L}^{op}, \mathrm{Set}] \\ &= [\mathcal{R}, [\mathcal{L}^{op}, \mathrm{Set}]] \\ &= [\mathcal{R}, \hat{\mathcal{L}}] \end{aligned}$$

A <u>negative</u> type N is an $(\mathcal{L}, \mathcal{R})$-bimodule which "represents" a "set of continuations", i.e.

$$\begin{aligned} N \in [\mathcal{L}^{op} \times \mathcal{R}, \mathrm{Set}]^{op} &= [\mathcal{L}^{op}, [\mathcal{R}, \mathrm{Set}]]^{op} \\ &= [\mathcal{L}, [\mathcal{R}, \mathrm{Set}]^{op}] \\ &= [\mathcal{L}, \check{\mathcal{R}}] \end{aligned}$$

Sequent Calculus Intuition

$$\#(\Gamma, \Delta) = \text{“}\Gamma \vdash \Delta\text{”}$$

$$P(\Gamma, \Delta) = \text{“}\Gamma \vdash [P]\, \Delta\text{”}$$

$$N(\Gamma, \Delta) = \text{“}\Gamma [N] \vdash \Delta\text{”}$$

formula “in focus”

---

other examples...

- $\mathcal{L} = \mathcal{R} = \mathbb{C}$, $\# = \mathbb{C}(-,-)$
- $\mathcal{L} = \mathbb{C}$, $\mathcal{R} = 1$, $\#(A, *) = \mathbb{C}(A, \bot)$
- $\mathcal{L} = [\mathbb{C}, \mathbb{C}]^{op}$, $\mathcal{R} = \mathbb{C}$, $\#(F, X) = F(X)$

Bimodule Semantics of Polarised Linear Logic

assume $\mathcal{L}$ and $\mathcal{R}$ are monoidal categories (overloading operations $\cdot$ and ε)

$$P_1 \otimes P_2 (\Gamma, \Delta) = \exists \Gamma_1, \Gamma_2, \Delta_1, \Delta_2.\ P_1(\Gamma_1, \Delta_1) \times P_2(\Gamma_2, \Delta_2) \times \mathcal{L}(\Gamma, \Gamma_1 \cdot \Gamma_2) \times \mathcal{R}(\Delta_1 \cdot \Delta_2, \Delta)$$

$$1(\Gamma, \Delta) = \mathcal{L}(\Gamma, \varepsilon) \times \mathcal{R}(\varepsilon, \Delta)$$

$$P_1 \oplus P_2 (\Gamma, \Delta) = P_1(\Gamma, \Delta) + P_2(\Gamma, \Delta)$$

$$0(\Gamma, \Delta) = \emptyset$$

$$N_1 ⅋ N_2 (\Gamma, \Delta) = \exists \Gamma_1 \Gamma_2 \Delta_1 \Delta_2.\ N_1(\Gamma_1, \Delta_1) \times N_2(\Gamma_2, \Delta_2) \times \mathcal{L}(\Gamma, \Gamma_1 \cdot \Gamma_2) \times \mathcal{R}(\Delta_1 \cdot \Delta_2, \Delta)$$

$$\bot(\Gamma, \Delta) = \mathcal{L}(\Gamma, \varepsilon) \times \mathcal{R}(\varepsilon, \Delta)$$

$$N_1 \& N_2 (\Gamma, \Delta) = N_1(\Gamma, \Delta) + N_2(\Gamma, \Delta)$$

$$\top(\Gamma, \Delta) = \emptyset$$

$$N^\perp(\Gamma, \Delta) = N(\Gamma, \Delta) \qquad P^\perp(\Gamma, \Delta) = P(\Gamma, \Delta)$$

$$\boxed{\begin{aligned} \downarrow N(\Gamma, \Delta) &= \forall \Gamma', \Delta'.\ N(\Gamma', \Delta') \to \#(\Gamma \cdot \Gamma', \Delta' \cdot \Delta) \\ \uparrow P(\Gamma, \Delta) &= \forall \Gamma', \Delta'.\ P(\Gamma', \Delta') \to \#(\Gamma \cdot \Gamma', \Delta' \cdot \Delta) \end{aligned}}$$

Isbell Revisited

$$
\begin{tikzcd}
{[\mathcal{R}, \hat{\mathcal{L}}] = \mathrm{Pos}} \arrow[rr, bend left, "P \mapsto \lambda\Gamma,\Delta.\, \mathrm{Pos}(P,\, {}_\Gamma\#^+_\Delta)"] \arrow[rr, phantom, "\perp"] & & {\mathrm{Neg} = [\mathcal{L}, \check{\mathcal{R}}]} \arrow[ll, bend left, "N \mapsto \lambda\Gamma,\Delta.\, \mathrm{Neg}({}_\Gamma\#^-_\Delta,\, N)"] \\
& \mathcal{L} \times \mathcal{R}^{op} \arrow[ul, "\#^+"] \arrow[ur, "\#^-"'] &
\end{tikzcd}
$$

where $\quad {}_\Gamma\#^+_\Delta = \#(\Gamma\cdot -, - \cdot \Delta) = {}_\Gamma\#^-_\Delta$

Building a more abstract Picture

Proof theory is about the <u>interaction</u> of types and contexts.

In particular, types can be placed <u>inside</u> of contexts.

Where does the (monoidal) structure of contexts come from?

Type-Context Adjunction

Suppose that a positive type

$$\mathcal{R} \xrightarrow{P} \hat{\mathcal{L}}$$

can be <u>lifted</u> along $\sharp$

$$\begin{tikzcd} & \hat{\mathcal{L}} \\ \mathcal{R} \arrow[ur, "\sharp^+"'] & \end{tikzcd}$$

to yield an operation $P^*: \mathcal{R} \to \mathcal{R}$ together with a 2-cell $\eta : P \Rightarrow \sharp^+ \cdot P^*$:

$$\begin{tikzcd} \mathcal{R} \arrow[rr, "P"] \arrow[dr, "P^*"'] & \arrow[d, phantom, "\Downarrow\eta"] & \hat{\mathcal{L}} \\ & \mathcal{R} \arrow[ur, "\sharp^+"'] & \end{tikzcd}$$

(left Kan lift)

We can view P^* as the operation of extending the right-context with P, and η precisely as the "focalisation rule":

$$\frac{\Gamma \vdash [P]\,\Delta}{\Gamma \vdash P, \Delta}\,\eta$$

Dually...

$$\begin{tikzcd} \mathcal{L} \arrow[rr, "N"] \arrow[dr, "N^*"'] & \arrow[d, phantom, "\Uparrow \varepsilon"] & \check{\mathcal{R}} \\ & \mathcal{L} \arrow[ur, "\#"'] & \end{tikzcd}$$

(right Kan lift)

$$\frac{\Gamma [N] \vdash \Delta}{\Gamma, N \vdash \Delta}\,\varepsilon$$

Type-Context Adjunction

More generally, for any $L : \mathcal{L} \to \mathcal{L}$, define
P^L as a left adjoint "at P" of postcomposition with $\#(L-,-)$, i.e. such that

$$\mathrm{End}(\mathcal{R})(P^L, R) \simeq [\mathcal{R}, \hat{\mathcal{L}}](P, \#(L-, R-))$$

Likewise, for any $R : \mathcal{R} \to \mathcal{R}$, define
N_R as a right adjoint "at N" of postcomposition with $\#(L-, R-)$, i.e. such that

$$\mathrm{End}(\mathcal{L})(L, N_R) \simeq [\mathcal{L}, \check{\mathcal{R}}](\#(L-, R-), N)$$

P^L is a sort of "weighted colimit"
N_R is a sort of "weighted limit"

Monoidal Structure Revisited

A monoidal product on $\mathcal{L}$ and $\mathcal{R}$ can be <u>derived</u> for contexts composed of types:

$$\varepsilon \cdot \Delta = \Delta$$
$$(P^L \Delta_1) \cdot \Delta_2 = P^L (\Delta_1 \cdot \Delta_2)$$

$$\Gamma \cdot \varepsilon = \Gamma$$
$$\Gamma_1 \cdot (N_R \Gamma_2) = N_R (\Gamma_1 \cdot \Gamma_2)$$

Structural Properties

In general, we won't have exchange:

$$\frac{\Gamma, N_1, N_2 \vdash \Delta}{\Gamma, N_2, N_1 \vdash \Delta} \qquad \frac{\Gamma \vdash P_1, P_2, \Delta}{\Gamma \vdash P_2, P_1, \Delta}$$

(only when P_1 and P_2 (N_1 and N_2) commute)

In general, we won't have weakening & contraction:

$$\frac{\Gamma \vdash \Delta}{\Gamma, N \vdash \Delta} \qquad \frac{\Gamma, N, N \vdash \Delta}{\Gamma, N \vdash \Delta} \qquad \frac{\Gamma \vdash P, P, \Delta}{\Gamma \vdash P, \Delta} \qquad \frac{\Gamma \vdash \Delta}{\Gamma \vdash P, \Delta}$$

(only when P (N) defines a (co)monad on $\mathcal{R}$ ($\mathcal{L}$))

General Principle:

Definition-by-Kan extension

$$\begin{tikzcd}
\mathcal{L}_0^{op} \times \mathcal{R}_0 \arrow[d, "i"'] \arrow[drr, "P_0"] & & \\
\mathcal{L}^{op} \times \mathcal{R} \arrow[rr, "P"'] \arrow[urr, phantom, "\Swarrow"] & & \mathrm{Set}
\end{tikzcd}$$

$$P(\Gamma, \Delta) = \exists \Gamma_0, \Delta_0.\ P_0(\Gamma_0, \Delta_0) \times \mathcal{L}(\Gamma, \Gamma_0) \times \mathcal{R}(\Delta_0, \Delta)$$

An abstract "subformula property"

The Exponentials

a candidate definition:

$$!N(\Gamma, \Delta) = \#(\Gamma, \Delta) + N(\Gamma, !N, \Delta)$$

$$?P(\Gamma, \Delta) = \#(\Gamma, \Delta) + P(\Gamma, ?P, \Delta)$$

caveats:

- Why is this well-defined?
- These may or may not be related to the exponentials in LL (cf. Melliès-Tabareau-Tasson, "An explicit formula for the free exponential modality of linear logic)

A Proto-Calculus

Contexts and context transformers

$$\Gamma ::= X \mid L(\Gamma) \qquad L ::= \alpha \mid N_R$$

$$\Delta ::= Y \mid R(\Delta) \qquad R ::= \beta \mid P^L$$

Judgments

$$\Gamma_1 \overset{\mathcal{L}}{\vdash} \Gamma_2 \;\mid\; L_1 \overset{E\mathcal{L}}{\vdash} L_2 \;\mid\; R_1 \overset{E\mathcal{R}}{\vdash} R_2 \;\mid\; \Delta_1 \overset{\mathcal{R}}{\vdash} \Delta_2$$

$$\Gamma[N] \vdash \Delta \;\mid\; \Gamma \vdash \Delta \;\mid\; \Gamma \vdash [P]\Delta$$

Rules

$$X \overset{\mathcal{L}}{\vdash} X \;\mid\; \alpha \overset{E\mathcal{L}}{\vdash} \alpha \;\mid\; \beta \overset{E\mathcal{R}}{\vdash} \beta \;\mid\; Y \overset{\mathcal{R}}{\vdash} Y$$

$$\frac{\Gamma \vdash [P]\Delta}{L(\Gamma) \vdash P^L(\Delta)} \qquad \frac{L_1 \overset{E\mathcal{L}}{\vdash} L_2 \quad \Gamma_1 \overset{\mathcal{L}}{\vdash} \Gamma_2}{L_1\Gamma_1 \overset{\mathcal{L}}{\vdash} L_2\Gamma_2}$$

$$\frac{R_1 \overset{E\mathcal{R}}{\vdash} R_2 \quad \Delta_1 \overset{\mathcal{R}}{\vdash} \Delta_2}{R_1\Delta_1 \overset{\mathcal{R}}{\vdash} R_2\Delta_2} \qquad \frac{\Gamma[N] \vdash \Delta}{N_R(\Gamma) \vdash R\Delta}$$

$$\frac{\Gamma \vdash \Gamma_0 \quad \Gamma_0 \vdash [P_0]\Delta_0 \quad \Delta_0 \vdash \Delta}{\Gamma \vdash [P]\Delta} \qquad \frac{\Gamma_0 \vdash [P_0]\Delta_0 \longrightarrow L\Gamma_0 \vdash R}{P^L \overset{E\mathcal{R}}{\vdash} R}$$

$$\frac{\Gamma_0[N_0] \vdash \Delta_0 \longrightarrow L\Gamma_0 \vdash R\Delta_0}{L \overset{E\mathcal{L}}{\vdash} N_R} \qquad \frac{\Gamma \vdash \Gamma_0 \quad \Gamma_0[N_0] \vdash \Delta_0 \quad \Delta_0 \vdash}{\Gamma[N] \vdash \Delta}$$