$\partial$ is for Dialectica

Marie Kerjean

CNRS & LIPN, Université Sorbonne Paris Nord

Work in collaboration with Pierre-Marie Pédrot
Differentiable programming

A new area triggered by the advances of deep learning algorithms on neural networks, it tries to attach two very old domains:

- Algorithmic Differentiation.
- $\lambda$-calculus.

**Goal:** Exploring modular way to express (algorithmic) differentiation in functional programming languages:

- Abadi & Plotkin, POPL20. (traces and big-step semantics)
- Brunel & Mazza & Pagani, POPL20, POPL21.
- Elliot, ICFP18, (compositional differentiation)
- Wang and al., ICFP 19, (delimited continuations)
- Interactions with probabilistic programming...
The real inventor of deep learning
Outline of the talk

1. Reverse differentiation and differentiable programming.

2. Dialectica acting on formulas.

3. Dialectica acting on $\lambda$-terms.

4. Factorizing Dialectica through differential linear logic.

5. Applications and related work.
Automatic Differentiation

How does one compute the differentiation of an algebraic expression, computed as a sequence of elementary operations?

E.g.:
\[
x_1 = x_0^2 \quad x_1' = 2x_0x_0'
\]
\[
x_2 = \cos(x_1) \quad x_2' = -x_0' \sin(x_0)
\]
\[
z = y + x_2 \quad z' = y' + 2x_2x_2'
\]

The computation of the final results requires the computation of the derivative of all partial computation. But in which order?

**Forward Mode differentiation** [Wengert, 1964]
\[
(x_1, x_1') \rightarrow (x_2, x_2') \rightarrow (z, z')
\]

**Reverse Mode differentiation**: [Speelpenning, Rall, 1980s]
\[
x_1 \rightarrow x_2 \rightarrow z \rightarrow z' \rightarrow x_2' \rightarrow x_1'
\]
*while keeping formal the unknown derivative.*
I hate graphs

\[ D_u(f \circ g) = D_{g(u)} f \circ D_u(g) \]

- **Forward Mode differentiation:**
  \[ g(u) \rightarrow D_u g \rightarrow f(g(u)) \rightarrow D_{g(u)} f \rightarrow D_{g(u)} f \circ D_u(g). \]

- **Reverse Mode differentiation:**
  \[ g(u) \rightarrow f(g(u)) \rightarrow D_{g(u)} f \rightarrow D_u g \rightarrow D_{g(u)} f \circ D_u(g). \]

The choice of an algorithm is due to complexity considerations:

- **Forward mode for** \( f \circ g : \mathbb{R} \rightarrow \mathbb{R}^n \).
- **Reverse mode for** \( f \circ g : \mathbb{R}^n \rightarrow \mathbb{R} \)

\( \leadsto \) Differentiation is about *linearizing* a function/program. Some people have a very specific idea of what a *linear program* or a *linear type* should be.
Idea: Reverse Differentials are contravariant

- **Forward Mode differentiation:**
  \[ h : A \Rightarrow B \sim \overrightarrow{\mathcal{D}} h : A \Rightarrow A \rightarrow B. \]

- **Reverse Mode differentiation:**
  \[ h : A \Rightarrow B \sim \overleftarrow{\mathcal{D}} h : A \Rightarrow B^\perp \rightarrow A^\perp. \]
AD from a functorial point of view

How to make differentiation functorial? Make it act on pairs!

\[ f : E \Rightarrow F \]

**forward:**
\[
\overrightarrow{D}(f) : \begin{cases} E \times E \rightarrow F \times F \\
(a, x) \mapsto (f(a), (D_a f \cdot x)) \end{cases}
\]

**backward:**
\[
\overleftarrow{D}(f) : \begin{cases} E \times F' \rightarrow F \times E' \\
(a, \ell) \mapsto (f(a), (\ell \circ D_a f)) \end{cases}
\]
Key Idea

Reverse derivatives are typed by linear negation.

Consider \( f : \mathbb{R}^n \to \mathbb{R}^m \) a function variable.

\[
\overleftarrow{D}(f) : \begin{cases} 
\mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^m \times \mathbb{R}^n \perp \\
(a, x) \mapsto (f(a), (\nu \mapsto x \cdot (D_a f \cdot \nu)))
\end{cases}
\]

This leads to a \textbf{compositional reverse derivative} transformation over the \textit{linear substitution calculus}, and proven complexity results.

\[
A, B, C ::= R \mid A \times B \mid A \to B \mid R^\perp_d \\
t, u ::= x \mid x^! \mid \lambda x. t \mid (t)u \mid t[x^{(i)}_! := u] \mid < t, u > \mid t + u...
\]
A Dialectica Transformation

- Gödel Dialectica transformation [1958]: a translation from intuitionistic arithmetic to a finite type extension of primitive recursive arithmetic.

\[ A \leadsto \exists u : \mathbb{W}(A), \forall x : \mathbb{C}(A), A^D[u, x] \]

- De Paiva [1991]: the linearized Dialectica translation operates on Linear Logic (types) and \(\lambda\)-calculus (terms).

- Pedrot [2014] A computational Dialectica translation preserving \(\beta\)-equivalence, via the introduction of an "abstract multiset constructor" on types on the target.
1. \((F \land G)' = (\exists yv) (zw) [A (y, z, x) \land B (v, w, u)].\)
2. \((F \lor G)' = (\exists yvt) (zw) [t=0 \land A (y, z, x) \lor t=1 \land B (v, w, u)].\)
3. \([(s) F]' = (\exists Y) (sz) A (Y (s), z, x).\)
4. \([(\exists s) F]' = (\exists sy) (z) A (y, z, x).\)
5. \((F \supset G)' = (\exists VZ) (yw) [A (y, Z (yw), x) \supset B (V (y), w, u)].\)
6. \((\neg F)' = (\exists \tilde{Z}) (y) \neg A (y, \tilde{Z} (y), x).\)
Gödel’s Dialectica

- Validates semi-classical axioms:
  - Markov’s principle: \( \neg \neg \exists x A \rightarrow \exists x A \) when \( A \) is decidable.
  - Independant of premises: \( (A \rightarrow \exists x B) \rightarrow (\exists x.(A \rightarrow B)) \)

- Numerous applications:
  - Soundness results
  - Proof mining

A further distinguishing feature of the D-interpretation is its nice behavior with respect to modus ponens. In contrast to cut-elimination, which entails a global (and computationally infeasible) transformation of proofs, the D-interpretation extracts constructive information through a purely local procedure: when proofs of \( \varphi \) and \( \varphi \rightarrow \psi \) are combined to yield a proof of \( \psi \), witnessing terms for the antecedents of this last inference are combined to yield a witnessing term for the conclusion. As a result of this modularity, the interpretation of a theorem can be readily obtained from the interpretations of the lemmata used in its proof.

A peek into Dialectica interpretation of functions

\[(A \rightarrow B)_D = \exists fg \forall xy (A_D(x, gxy) \rightarrow B_D(fx, y))\]

**Usual explanation**: least unconstructive prenexation.

- Start from \(\exists x, \forall u, A_D[x, u] \rightarrow \exists y, \forall v, B_D[y, v]\).
- Obvious prenexation: \(\forall x (\forall u, A_D[x, u] \rightarrow \exists y, \forall v, B_D[y, v])\).
- Weak form of IP: \(\forall x \exists y (\forall u, A_D[x, u] \rightarrow \forall v, B_D[y, v])\).
- Prenexation: \(\forall x \exists y, \forall v, \forall \neg \exists u (A_D[x, u] \rightarrow B_D[y, v])\).
- Markov: \(\forall x, \exists y, \forall v, \exists u(A_D[x, u] \rightarrow B_D[y, v])\).
- Axiom of choice: \(\exists f, \exists g, \forall u, \forall v, (A_D(u, guv) \rightarrow B_D[fx, y])\).

**Dynamic behaviour**: agrees to a chain rule.

**Mathematical meaning**: it’s some kind of approximation.
Dialectica verifies the chain rules

\((A \Rightarrow B)_D[\phi_1; \psi_1, u_1; v_1] := A_D(u_1, \psi_1 u_1 v_1) \Rightarrow B_D(\phi_1 u_1, v_1)\)

\((B \Rightarrow C)_D[\phi_2; \psi_2, u_2; v_2] := B_D(u_2, \psi_2 u_2 v_2) \Rightarrow C_D(\phi_2 u_2, v_2)\)

\((A \Rightarrow C)_D[\phi_3; \psi_3, u_3; v_3] := A_D(u_3, \psi_3 u_3 v_3) \Rightarrow C_D(\phi_3 u_3, v_3)\)

The Dialectica interpretation amounts to the following equations:

\[ u_3 = u_1 \quad \psi_3 u_3 v_3 = \psi_1 u_1 v_1 \]

\[ v_3 = v_2 \quad \phi_2 u_2 = \phi_1 u_1 \]

\[ u_2 = \phi_1 u_1 \quad v_2 = \phi_1 u_1 v_1 \]

which can be simplified to:

\[ \phi_3 u_3 = \phi_2 (\phi_1 u_3) \text{ composition of functions} \]

\[ \psi_3 u_3 v_3 = \psi_2 (\phi_1 u_3) (\psi_1 u_3 v_3) \text{ composition of their differentials} \]
Types!

\[ A \leadsto \exists x : \mathcal{W}(A), \forall u : \mathcal{C}(A), A_D[x, u] \]

Witness and counter types:

\[ \mathcal{C}(A \Rightarrow B) = \mathcal{C}(A) \times \mathcal{C}(B) \]

\[ \mathcal{W}(A \Rightarrow B) = (\mathcal{W}(A) \Rightarrow \mathcal{W}(B)) \times (\mathcal{W}(A) \Rightarrow \mathcal{C}(B) \Rightarrow \mathcal{C}(A)) \]
Types!

\[ A \rightsquigarrow \exists x : W(A), \forall u : C(A), A_D[x, u] \]

**Witness and counter types:**

\[ C(A \Rightarrow B) = C(A) \times C(B) \]

\[ W(A \Rightarrow B) = (W(A) \Rightarrow W(B)) \times \left( W(A) \Rightarrow C(B) \Rightarrow C(A) \right) \]

**Global witness**

**Local opponent**

**Function**

**Reverse derivative**
Let’s say $x$, $u$, $f$, $g$ are $\lambda$-terms.

A reverse Differential $\lambda$-calculus

"Behind every successful proof there is a program", Gödel’s wife
A computational Dialectica

Making Dialectica act on $\lambda$-terms instead of formulas:

An abstract multiset $M(-)$

$$\begin{align*}
\Gamma \vdash \emptyset & : M A \\
\Gamma \vdash t : A & \quad \Gamma \vdash m_1 : M A \\
\Gamma \vdash \{t\} : M A & \quad \Gamma \vdash m_1 \otimes m_2 : M A \\
\Gamma \vdash f : A \Rightarrow M B & \quad \Gamma \vdash m \gg f : M B
\end{align*}$$

$$\begin{align*}
W(A \Rightarrow B) & := \left( W(A) \Rightarrow W(B) \right) \\
\times (C(B) \Rightarrow W(A) \Rightarrow M C(A)) \\
C(A \Rightarrow B) & := W(A) \times C(B)
\end{align*}$$
Pédrot’s Dialectica Transformation

Soundness [Ped14]

If $\Gamma \vdash t : A$ in the source then we have in the target

1. $\mathbb{W}(\Gamma) \vdash t^\bullet : \mathbb{W}(A)$
2. $\mathbb{W}(\Gamma) \vdash t_x : \mathbb{C}(A) \Rightarrow \mathbb{M} \mathbb{C}(X)$ provided $x : X \in \Gamma$.

A global and a local transformation

$x^\bullet := x \quad (\lambda x. t)^\bullet := (\lambda x. t^\bullet, \lambda \pi x. t_x \pi)$

$x_x := \lambda \pi. \{\pi\} \quad (\lambda x. t)_y := \lambda \pi. (\lambda x. t_y) \pi.1 \pi.2$

$x_y := \lambda \pi. \emptyset$ if $x \neq y \quad (t \ u)^\bullet := (t^\bullet.1) \ u^\bullet$

$(t \ u)_y := \lambda \pi. (t_y (u^\bullet, \pi)) \otimes ((t^\bullet.2) \pi u^\bullet \gg u_y)$
Flashback: Differential $\lambda$-calculus [Ehrhard, Regnier 04]

Inspired by denotational models of Linear Logic in vector spaces of sequences, it introduces a differentiation of $\lambda$-terms.

$D(\lambda x.t)$ is the linearization of $\lambda x.t$, it substitute $x$ linearly, and then it remains a term $t'$ where $x$ is free.

Syntax:

\[
\begin{align*}
\Lambda^d : & \quad S, T, U, V ::= 0 \mid s \mid s + T \\
\Lambda^s : & \quad s, t, u, v ::= x \mid \lambda x.s \mid sT \mid Ds \cdot t
\end{align*}
\]

Operational Semantics:

\[
\begin{align*}
(\lambda x.s)T \rightarrow_\beta s[T/x] \\
D(\lambda x.s) \cdot t \rightarrow_\beta D \lambda x.\frac{\partial s}{\partial x} \cdot t
\end{align*}
\]

where $\frac{\partial s}{\partial x} \cdot t$ is the linear substitution of $x$ by $t$ in $s$. 
Linearity in Linear Logic

**Linearity is about resources:** A proof/program is *linear* iff it uses only once its hypotheses/argument.

<table>
<thead>
<tr>
<th>Linear</th>
<th>Non-linear</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A \vdash A \lor B$</td>
<td>$A \vdash A \land A$</td>
</tr>
<tr>
<td>$\lambda f \lambda x. fxx$</td>
<td>$\lambda x. \lambda f. fxx$</td>
</tr>
</tbody>
</table>

Usual Implication

A call-by-name translation

$A \Rightarrow B = ! A \rightarrow B$

$\mathcal{C}^\infty(A, B) \sim \mathcal{L}(!A, B)$
Linearity in Linear Logic

**Linearity is about resources:** A proof/program is *linear* iff it uses only once its hypotheses/argument.

### Linear vs. Non-linear

<table>
<thead>
<tr>
<th>Linear</th>
<th>Non-linear</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A \vdash A \lor B$</td>
<td>$A \vdash A \land A$</td>
</tr>
<tr>
<td>$\lambda f \lambda x. fxx$</td>
<td>$\lambda x. \lambda f. fxx$</td>
</tr>
</tbody>
</table>

**Usual implication**

**A call-by-name translation**

$$A \Rightarrow B = !A \multimap B$$

$$\mathcal{C}^\infty (A, B) \simeq \mathcal{L}(!A, B)$$

**Linear Implication**
Linearity in Linear Logic

**Linearity is about resources:** A proof/program is *linear* iff it uses only once its hypotheses/argument.

<table>
<thead>
<tr>
<th>Linear</th>
<th>Non-linear</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A \vdash A \lor B$</td>
<td>$A \vdash A \land A$</td>
</tr>
<tr>
<td>$\lambda f \lambda x. fxx$</td>
<td>$\lambda x. \lambda f. fxx$</td>
</tr>
</tbody>
</table>

Usual implication

A call-by-name translation

A *call-by-name* translation

$A \Rightarrow B = \text{!} A \rightsquigarrow B$

$\mathcal{C}^\infty (A, B) \simeq \mathcal{L}(\text{!}A, B)$

**Exponential**

**Smooth Semantics**
The linear substitution ...

... which is not exactly a substitution

\[
\frac{\partial y}{\partial x} \cdot T = \begin{cases} 
T & \text{if } x = y \\
0 & \text{otherwise}
\end{cases}
\]

\[
\frac{\partial}{\partial x} (sU) \cdot T = \left( \frac{\partial s}{\partial x} \cdot T \right) U + \left( Ds \cdot \left( \frac{\partial U}{\partial x} \cdot T \right) \right) U
\]

\[
\frac{\partial}{\partial x} (\lambda y \cdot s) \cdot T = \lambda y \cdot \frac{\partial s}{\partial x} \cdot T
\]

\[
\frac{\partial}{\partial x} (Ds \cdot u) \cdot T = D \left( \frac{\partial s}{\partial x} \cdot T \right) \cdot u + Ds \cdot \left( \frac{\partial u}{\partial x} \cdot T \right)
\]

\[
\frac{\partial 0}{\partial x} \cdot T = 0
\]

\[
\frac{\partial}{\partial x} (s + U) \cdot T = \frac{\partial s}{\partial x} \cdot T + \frac{\partial U}{\partial x} \cdot T
\]

\[
\frac{\partial s}{\partial x} \cdot t \text{ represents } s \text{ where } x \text{ is linearly (i.e. one time) substituted by } t.
\]
7 years ago: "That's Differential $\lambda$-calculus"

\[
\begin{align*}
  x_x & := \lambda \pi. \{\pi\} & x^* & := x \\
  x_y & := \lambda \pi. \emptyset \quad \text{if } x \neq y & (\lambda x. t)^* & := (\lambda x. t^*, \lambda x \pi. t_x \pi) \\
  (\lambda x. t)_y & := \lambda \pi. (\lambda x. t_y) \pi.1 \pi.2 & (t u)^* & := (t^*.1) u^* \\
  (t u)_y & := \lambda \pi. (t_y (u^*, \pi)) \otimes ((t^*.2) u^* \pi \gg u_y)
\end{align*}
\]
7 years ago: "That's Differential λ-calculus"

\[\begin{align*}
x_x &:= \lambda \pi. \{ \pi \} & x^* &:= x \\
x_y &:= \lambda \pi. \emptyset \text{ if } x \neq y & (\lambda x. t)^* &:= (\lambda x. t^*, \lambda x\pi. t_x \pi) \\
(\lambda x. t)_y &:= \lambda \pi. (\lambda x. t_y) \pi.1 \pi.2 & (t u)^* &:= (t^*.1) u^* \\
(t u)_y &:= \lambda \pi. (t_y (u^*, \pi)) \otimes ((t^*.2) u^* \pi \gg u_y)
\end{align*}\]
Tracking differentiation in Dialectica

7 years ago: ”That’s Differential $\lambda$-calculus”

\[
x_x := \lambda \pi. \frac{\partial x}{\partial x} \cdot \pi \quad x^* := x
\]
\[
x_y := \lambda \pi. \frac{\partial x}{\partial y} \cdot \pi \quad \text{if } x \neq y \quad (\lambda x. t)^* := (\lambda x. t^*, \lambda x \pi. t_x \, \pi)
\]
\[
(\lambda x. t)_y := \lambda \pi. (\lambda x. t_y \pi.1 \pi.2 \quad (t \, u)^* := \equiv (\lambda x. (tx)^*) \, u^*
\]
\[
(t \, u)_y := \lambda \pi. (t_y \, (u^*, \pi)) \, \odot (t^* \, 2 \, u^* \pi \gg u_y)
\]

3 years ago: That’s reverse differentiation

- $(\_)^* \cdot 2$ obeys the chain rule, $(\_)^*$ is the functorial differentiation.
- $t_x$ is contravariant in $x$. 
Tracking differentiation in Dialectica

7 years ago: "That's Differential $\lambda$-calculus"

\[
x_x := \lambda \pi. \frac{\partial x}{\partial x} \cdot \pi \quad x^* := x
\]

\[
x_y := \lambda \pi. \frac{\partial x}{\partial y} \cdot \pi \quad \text{if } x \neq y \quad (\lambda x. t)^* := (\lambda x. t^*, \lambda x. \pi. t_x \ \pi)
\]

\[
(\lambda x. t)_y := \lambda \pi. (\lambda x. t_y) \ \pi.1 \ \pi.2 \quad (t \ u)^* \equiv (\lambda x. (tx)^*) \ u^*
\]

3 years ago: That's reverse differentiation

- $(\_)^*.2$ obeys the chain rule, $(\_)^*$ is the functorial differentiation.
- $t_x$ is contravariant in $x$.

\[\llbracket u \ggg t_x \rrbracket \equiv \lambda z. (\llbracket u \rrbracket (\frac{\partial t}{\partial x} \cdot z))\]

up to the linearity of $\llbracket u \rrbracket$, IRL we make use of two logical relations
Dialectica is reverse differential $\lambda$-calculus

where the linearity of counter terms is not enforced.

Two logical relations : the arrow case

\[
\begin{align*}
    t \sim_{A \to B} T & := \forall u \sim_A U. (t.1 u) \sim_B (T U) \\
    & \quad \land (t.2 u) \bowtie^A_B (\lambda z. (DT \cdot z) U) \\
    t \bowtie^X_{A \to B} T & := \forall u \sim_A U. \lambda \pi. t (u, \pi) \bowtie^X_B (\lambda z. T z U)
\end{align*}
\]

Theorem

If $\Gamma \vdash t : A$ is a simply-typed $\lambda$-term, then

- for all $\vec{r} \sim_{\vec{R}} \vec{R}$, $t^* \{\Gamma \leftarrow \vec{r}\} \sim_A t \{\Gamma \leftarrow \vec{R}\}$,
- and for all $\vec{r} \sim_{\vec{R}}$ and $x : X \in \Gamma$,

\[
t_x \{\Gamma \leftarrow \vec{r}\} \bowtie^X_A \lambda z. \left( \frac{\partial t}{\partial x} \cdot z \right) \{\Gamma \leftarrow \vec{R}\}.
\]
A Linear Logic Refinement
Differential Linear Logic

\[ \vdash \ell : A \multimap B \]
\[ \vdash \ell : !A \multimap B \]
\[ \vdash f : !A \multimap B \]
\[ \vdash D_0 f : A \multimap B \]

A linear proof is in particular non-linear.

From a non-linear proof we can extract a linear proof.

\[ f \in C^\infty(\mathbb{R}, \mathbb{R}) \]

\[ d(f)(0) \]

Differential interaction nets, Ehrhard and Regnier, TCS (2006)
Exponential rules of Differential Linear Logic

\[
\frac{\Gamma \vdash B}{\Gamma, !A \vdash B} \quad \frac{\Gamma, !A, !A \vdash B}{\Gamma, !A \vdash B} \quad \frac{\Gamma, A \vdash B}{\Gamma, !A \vdash B} \\
\frac{\vdash !A}{\quad \bar{w}} \quad \frac{\Gamma \vdash !A \quad \Delta \vdash !A}{\Gamma, \Delta, \vdash !A} \quad \frac{\Gamma \vdash A}{\Gamma \vdash !A} \\
\frac{?\Gamma \vdash A}{\quad \bar{p}} \quad \frac{?\Gamma \vdash !A}{\quad \bar{d}}
\]
Dialectica factorizes through Linear Logic

\[ W(A^\perp) := C(A) \]
\[ W(A \oplus B) := W(A) + W(B) \]
\[ W(!A) := W(A) \]
\[ C(A^\perp) := W(A) \]
\[ C(A \oplus B) := C(A) \times C(B) \]
\[ C(!A) := W(A) \Rightarrow C(A) \]
\[ W(A \otimes B) := W(A) \times W(B) \]
\[ C(A \otimes B) := (W(A) \Rightarrow C(B)) \times (W(B) \Rightarrow C(A)) \]
Dialectica factorizes through Differential Linear Logic

\[
\begin{align*}
\mathbb{W}(!A) & := !\mathbb{W}(A) & \mathbb{C}(!A) & := !\mathbb{W}(A) \multimap \mathbb{C}(A) \\
\mathbb{W}(A \otimes B) & := \mathbb{W}(A) \otimes \mathbb{W}(B) & \mathbb{C}(A \otimes B) & := (\mathbb{W}(A) \multimap \mathbb{C}(B)) \oplus (\mathbb{W}(B) \multimap \mathbb{C}(A)) \\
\mathbb{W}(A \multimap B) & := (\mathbb{W}(A) \multimap \mathbb{W}(B)) \& (\mathbb{C}(B) \multimap \mathbb{C}(A)) & \mathbb{C}(A \multimap B) & := \mathbb{W}(A) \otimes \mathbb{C}(B)
\end{align*}
\]

If \( \Gamma \vdash A \) in LL, then \( \mathbb{W}(\Gamma) \vdash \mathbb{W}(A) \) in classical DiLL.

\[
\begin{align*}
\Gamma \vdash & A, A \perp & \text{ax} \\
\vdash & A, !A \perp & \bar{d} \\
\vdash & A, !A \perp & \text{ax} \\
\vdash & ?A, !A \perp & \bar{c} \\
\vdash & ?A, A, !A \perp & \pi \\
\Gamma \vdash & ?A, A & \text{cut}
\end{align*}
\]
Dialectica factorizes through Differential Linear Logic

The economical translation

\[
\begin{align*}
[A \Rightarrow B]_e &:= !A \multimap B \\
[A \times B]_e &:= A & B \\
[A + B]_e &:= A \oplus B
\end{align*}
\]

\[\text{ILL} \xrightarrow{W} \text{C} \xrightarrow{\ldots} \text{IDiLL}\]

\[\lambda^{+,\times} \xrightarrow{W} \text{C} \xrightarrow{\ldots} \lambda^{+,\times}\]

IDiLL : Intuitionistic Differential Linear Logic? Oh no ...
Dialectica categories through Differential Categories

Categories representing specific relations

Consider a category $\mathcal{C}$. $\text{Dial}(\mathcal{C})$ is constructed as follows:

- **Objects**: relations $\alpha \subseteq U \times X$, $\beta \subseteq V \times Y$.
- **Maps from $\alpha$ to $\beta$**: $(f : U \to V, F : U \times Y \to X)$
- **Composition**: the chain rule!

Consider

$$(f, F) : \alpha \subseteq (A, X) \to \beta \subseteq (B, Y)$$ and $$(g, G) : \beta \subseteq (B, Y) \to \gamma \subseteq (C, Z)$$

two arrows of the Dialectica category. Then their composition is defined as

$$(g, G) \circ (f, F) := (g \circ f, (a, z) \mapsto G(f(a), F(a, z))).$$
Dialectica categories through Differential Categories

In a $\ast$-autonomous differential category:

$$\partial : \text{Id} \otimes ! \rightarrow !$$

$$\mathcal{L}(B \otimes A, C^\perp) \simeq \mathcal{L}(A, (B \otimes C)^\perp)$$

From $f : !A \rightarrow B$ one constructs:

$$\overleftarrow{D}(f) \in \mathcal{L}(!A \otimes B^\perp, A^\perp).$$

Dialectica categories factorize through differential categories

If $\mathcal{L}$ is a model of $\text{DILL}$ such that $\mathcal{L}_!$ has finite limits:

$$\begin{cases}
\mathcal{L}_! \rightarrow \mathcal{D}(\mathcal{L}_!)
A \mapsto A \times A^\perp
f \mapsto (f, \overleftarrow{D}(f))
\end{cases}$$

To be declined in reverse/cartesian differential categories...
Conclusion and applications
Take home message:

**Dialectica is functorial reverse differentiation**, extracting intensional local content from proofs.

Related work and applications:

- Semantics: Ehrhard’s differentiation without sums.
- Markov’s principle and delimited continuations on positive formulas.
- Proof mining and backpropagation.
Ehrhard’s differentiation without sums

Content. We base our approach on a concept of summable pair that we axiomatize as a general categorical notion in Section 2: a summable category is a category \( \mathcal{L} \) with 0-morphisms\(^1\) together with a functor \( S : \mathcal{L} \to \mathcal{L} \) equipped with three natural transformations from \( SX \) to \( X \): two projections and a sum operation. The first projection also exists in the “tangent bundle” functor of a tangent category but the two other morphisms do not. Such a summability structure induces a monad structure on \( S \) (a similar phenomenon occurs in tangent categories). In Section 3 we consider the case where the category is a cartesian SMC equipped with a resource comonad !\( -\) in the sense of LL where we present differentiation as a distributive law between the monad \( S \) and the comonad !\( -\). This allows to extend \( S \) to a strong monad \( D \) on the Kleisli category \( \mathcal{L}_1 \) which implements differentiation of non-linear maps. In Section 4 we study the case where the functor \( S \) can be defined using a more basic structure of \( \mathcal{L} \) based on the object \( 1 \& 1 \) where \( \& \) is the cartesian product and \( 1 \) is the unit of \( \otimes \): this is actually what happens in

Thomas Ehrhard. Coherent differentiation. 2021
Dialectica is differentiation ... 

... We knew it already!

The cordereliction of differential proof nets: In terms of polarity in linear logic [23], the \( \forall - \rightarrow \)-free constraint characterizes the formulas of intuitionistic logic that can be built only from positive connectives \( (\oplus, \otimes, 0, 1, !) \) and the why-not connective ("?"). In this framework, Markov’s principle expresses that from such a \( \forall - \rightarrow \)-free formula \( A \) (e.g. \( \exists x (?A(x) \otimes ?B(x)) \)) where the presence of "?" indicates that the proof possibly used weakening (\texttt{efq} or \texttt{throw}) or contraction (\texttt{catch}), a linear proof of \( A \) purged from the occurrences of its "?" connective can be extracted (meaning for the example above a proof of \( \exists x (A(x) \otimes B(x)) \)). Interestingly, the removal of the "?", i.e. the steps from \( ?P \) to \( P \), correspond to applying the cordereliction rule of differential proof nets [24].

**Differentiation** : \((?P = (P \rightarrow \bot) \Rightarrow \bot) \rightarrow ((P \rightarrow \bot) \rightarrow \bot) \equiv P\)

Hugo Herbelin, “An intuitionistic logic that proves Markov’s principle”, LICS ’10.
Markov’s principle is proved by allowing catch and throw operations on hereditary positive formulas.

**Figure 3.** Proof of MP
Proof Mining

Extracting quantitative information from proofs.

Effective moduli from ineffective uniqueness proofs. An unwinding of de La Vallée Poussin’s proof for Chebycheff approximation\textsuperscript{*}

Ulrich Kohlenbach

Fachbereich Mathematik, J.W. Goethe Universität
Robert Mayer Str. 6 10, 6000 Frankfurt am Main, FRG

Abstract

We consider uniqueness theorems in classical analysis having the form

\[(+) \forall u \in U, v_1, v_2 \in V_u (G(u, v_1) = 0 = G(u, v_2) \rightarrow v_1 = v_2),\]

where \(U, V\) are complete separable metric spaces, \(V_u\) is compact in \(V\) and \(G : U \times V \rightarrow \mathbb{R}\) is a constructive function.

If \((+)\) is proved by arithmetical means from analytical assumptions

\[(+++) \forall x \in X \exists y \in Y_x \forall z \in Z (F(x, y, z) = 0)\]

only (where \(X, Y, Z\) are complete separable metric spaces, \(Y_x \subset Y\) is compact and \(F : X \times Y \times Z \rightarrow \mathbb{R}\) constructive), then we can extract from the proof of \((++) \rightarrow (+)\) an effective modulus of uniqueness, i.e.

\[(+++) \forall u \in U, v_1, v_2 \in V_u, k \in \mathbb{N} (|G(u, v_1)|, |G(u, v_2)| \leq 2^{-\phi u k} \rightarrow d_V(v_1, v_2) \leq 2^{-k}).\]

Differentiate the function \((\epsilon \rightarrow \eta)\) in :

\[\forall u, v_1 v_2, \forall \epsilon > 0, \exists \eta > 0, \|G(u, v_1) - G(u, v_2)\| < \eta \rightarrow d_V(v_1, v_2) < \epsilon\|