Thomas Ehrhard's 60 birthday

∂ is for Dialectica

Marie Kerjean

CNRS & LIPN, Université Sorbonne Paris Nord

Work in collaboration with Pierre-Marie Pédrot

Thank you Thomas

... For the opportunity to finally understand **differentiation**.

What's differentiation ?

Thank you Thomas

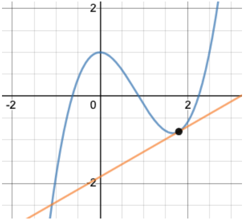
... For the opportunity to finally understand **differentiation**.

What's differentiation ?

Thank you Thomas

... For the opportunity to finally understand **differentiation**.

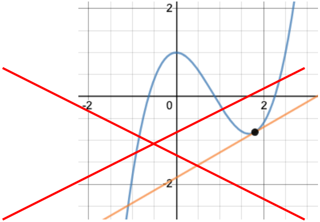
What's differentiation ?



Thank you Thomas

... For the opportunity to finally understand **differentiation**.

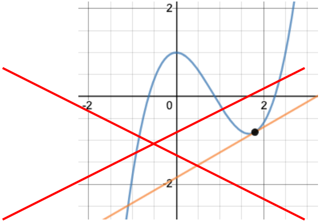
What's differentiation ?

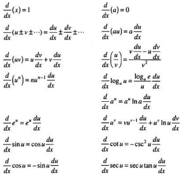


Thank you Thomas

... For the opportunity to finally understand **differentiation**.

What's differentiation ?

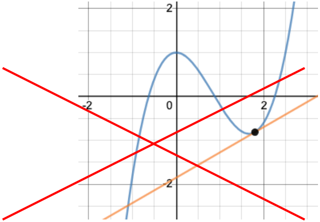


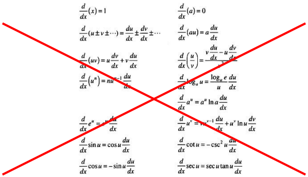


Thank you Thomas

... For the opportunity to finally understand **differentiation**.

What's differentiation ?

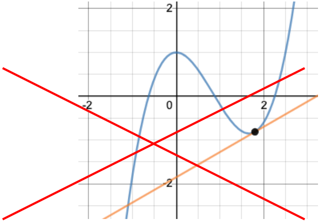


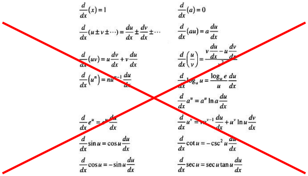


Thank you Thomas

... For the opportunity to finally understand **differentiation**.

What's differentiation ?





That's differentiation !

$$\frac{\partial}{\partial x}((\lambda z.t)u) \cdot s = (\frac{\partial(\lambda z.t)}{\partial x} \cdot s)u + (\mathrm{D}(\lambda z.y) \cdot (\frac{\partial u}{\partial x} \cdot s))u$$

What's this talk is about

Joining Dialectica and Differential λ-calculus
through
Reverse Differentiation

What's this talk is about

Joining Dialectica and Differential Linear Logic
through
Reverse Differentiation

What's this talk is about

Joining Dialectica and Differential Categories
through
Reverse Differentiation

What's this talk is about

Joining Dialectica and Differential λ-calculus
through
Reverse Differentiation

Gödel's Dialectica Transformation

1. $(\mathrm{F} \wedge \mathrm{G})' = (\exists yv)\,(zw)\,[\mathrm{A}\,(y, z, x) \wedge \mathrm{B}\,(v, w, u)]$.
2. $(\mathrm{F} \vee \mathrm{G})' = (\exists yvt)\,(zw)\,[t=0 \wedge \mathrm{A}\,(y, z, x) \cdot \vee \cdot t=1 \wedge \mathrm{B}\,(v, w, u)]$.
3. $[(s)\,\mathrm{F}]' = (\exists \mathrm{Y})\,(sz)\,\mathrm{A}\,(\mathrm{Y}\,(s), z, x)$.
4. $[(\exists s)\,\mathrm{F}]' = (\exists sy)\,(z)\,\mathrm{A}\,(y, z, x)$.
5. $(\mathrm{F} \supset \mathrm{G})' = (\exists \mathrm{VZ})\,(yw)\,[\mathrm{A}\,(y, \mathrm{Z}\,(yw), x) \supset \mathrm{B}\,(\mathrm{V}\,(y), w, u)]$.
6. $(\neg \mathrm{F})' = (\exists \bar{\mathrm{Z}})\,(y)\,\neg \mathrm{A}\,(y, \bar{\mathrm{Z}}\,(y), x)$.

Kurt Gödel (1958). Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes. Dialectica.

- Validates semi-classical axioms:
  - Markov's principle : $\neg\neg\exists x A \rightarrow \exists x A$ when A is decidable.
- Numerous applications :
  - Soudness results
  - **Proof mining**: applying Dialectica to theorems in analysis extract quantitative information.

"There are infinitely many prime numbers."

$\Downarrow$

"For any m there exists some $m < p \leq \lceil e^{m-\gamma} \rceil$ such that p is prime.

And now for something completely different : Automatic Differentiation

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

E.g. : $z = y + cos(x^2)$ $\quad \begin{array}{ll} x_1 = x_0^2 & x_1' = 2x_0x_0' \\ x_2 = cos(x_1) & x_2' = -x_0'sin(x_0) \\ z = y + x_2 & z' = y' + 2x_2x_2' \end{array}$

Derivative of a sequence of instruction

$$\Downarrow$$

sequence of instruction $\times$ sequence of derivatives

Forward Mode differentiation [Wengert, 1964]
$(x_1, x_1') \to (x_2, x_2') \to (z, z')$.
Reverse Mode differentiation: [Speelpenning, Rall, 1980s]
$x_1 \to x_2 \to z \to z' \to x_2' \to x_1'$ *while keeping formal the unknown derivative*.

I hate graphs

$$\mathrm{D}_u(f \circ g) = D_{g(u)} f \circ D_u(g)$$

- **Forward Mode differentiation :**
  $g(u) \to D_u g \to f(g(u)) \to D_{g(u)} f \to D_{g(u)} f \circ D_u(g)$.
- **Reverse Mode differentiation:**
  $g(u) \to f(g(u)) \to D_{g(u)} f \to D_u(g) \to D_{g(u)} f \circ D_u(g)$

The choice of an algorithm is due to complexity considerations:

- **Forward mode** for $f \circ g : \mathbb{R} \to \mathbb{R}^n$.
- **Reverse mode** for $f \circ g : \mathbb{R}^n \to \mathbb{R}$

$\rightsquigarrow$ *Differentiable programming* is a new research area triggered by the advances of deep learning algorithms on neural networks, it tries to attach two very old domains: lambda-calculus and automatic differentiation, with *correctness* and *modularity* goals in mind.

Functorial Forward AD

$$\mathbf{D}_u(f \circ g) = \mathbf{D}_{g(u)} f \circ \mathbf{D}_u(g)$$

Non-functorial !!!

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

Forward Mode differentiation :

$$g : E \Rightarrow F \rightsquigarrow \overrightarrow{D} g : E \Rightarrow E \multimap F.$$

Functorial forward differentiation :

$$\overrightarrow{D}(g) : \begin{cases} E \times E \to F \times F \\ (a, x) \mapsto (f(a), (\mathrm{D}_a f \cdot x)) \end{cases}$$

Reverse functorial differentiation

Linear implication

$$A^{\perp} \equiv A \multimap \perp \equiv \mathcal{L}(A, \mathbb{R}) \equiv A'$$

Reverse functorial differentiation

Linear implication

$$A^{\perp} \equiv A \multimap \perp \equiv \mathcal{L}(A, \mathbb{R}) \equiv A'$$

- **Reverse Mode differentiation:**
  $g(u) \to f(g(u)) \to D_{g(u)}f \to D_{g(u)}f \circ D_u(g)$

$$D_u(g) : F' \multimap E'; \ell \mapsto \ell \circ D_u g$$

$$g : E \Rightarrow F \rightsquigarrow \overleftarrow{D}g : E \Rightarrow F^{\perp} \multimap E^{\perp}.$$

[Mazza, Pagani, POPL2020]

Reverse functorial differentiation

Linear implication

$$A^{\perp} \equiv A \multimap \perp \equiv \mathcal{L}(A, \mathbb{R}) \equiv A'$$

- **Reverse Mode differentiation:**
  $g(u) \to f(g(u)) \to D_{g(u)} f \to D_{g(u)} f \circ D_u(g)$

$$D_u(g) : F' \multimap E'; \ell \mapsto \ell \circ D_u g$$

$$g : E \Rightarrow F \rightsquigarrow \overleftarrow{D} g : E \Rightarrow F^{\perp} \multimap E^{\perp}.$$

[Mazza, Pagani, POPL2020]

- **Reverse functorial differentiation :**

$$(f, \overleftarrow{D}(f)) : (E \Rightarrow F) \times (E \Rightarrow F^{\perp} \multimap E^{\perp})$$

Outline of the talk

- Reverse differentiation and differentiable programming.
- Dialectica acting on formulas.
- Dialectica acting on λ-terms.
- Factorizing Dialectica through differential linear logic.
- Applications and related work.

A Dialectica Transformation

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

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

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

- Pedrot [2014] A *computational* Dialectica translation preserving β-equivalence, via the introduction of an "abstract multiset constructor" on types on the target.

Gödel's Dialectica

1. $(\mathrm{F} \wedge \mathrm{G})' = (\exists yv)(zw)[\mathrm{A}(y,z,x) \wedge \mathrm{B}(v,w,u)]$.
2. $(\mathrm{F} \vee \mathrm{G})' = (\exists yvt)(zw)[t=0 \wedge \mathrm{A}(y,z,x) \cdot \vee \cdot t=1 \wedge \mathrm{B}(v,w,u)]$.
3. $[(s)\,\mathrm{F}]' = (\exists \mathrm{Y})(sz)\,\mathrm{A}(\mathrm{Y}(s), z, x)$.
4. $[(\exists s)\,\mathrm{F}]' = (\exists sy)(z)\,\mathrm{A}(y,z,x)$.
5. $(\mathrm{F} \supset \mathrm{G})' = (\exists \mathrm{V}\mathrm{Z})(yw)[\mathrm{A}(y, \mathrm{Z}(yw), x) \supset \mathrm{B}(\mathrm{V}(y), w, u)]$.
6. $(\neg \mathrm{F})' = (\exists \bar{\mathrm{Z}})(y) \neg \mathrm{A}(y, \bar{\mathrm{Z}}(y), x)$.

Kurt Gödel (1958). Über eine bisher noch nicht benützte Erweiterung des finiten Standpunktes. Dialectica.

Gödel's Dialectica

- Validates semi-classical axioms:
  - Markov's principle : $\neg\neg\exists x A \to \exists x A$ when A is decidable.
  - Independant of premises : $(A \to \exists x B) \to (\exists x.(A \to B))$
- Numerous applications :
  - Soudness 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 φ and $\varphi \to \psi$ are combined to yield a proof of ψ, 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.

Jeremy Avigad and Solomon Feferman (1999). Gödel's functional ("Dialectica") interpretation

A peek into Dialectica interpretation of functions

$$(A \to B)_D = \exists fg \forall xy (A_D(x, gxy) \to B_D(fx, y))$$

Usual explanation : least unconstructive prenexation.

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

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:

$$\begin{aligned} u_3 &= u_1 & \qquad \psi_3\, u_3\, v_3 &= \psi_1\, u_1\, v_1 \\ v_3 &= v_2 & \phi_2\, u_2 &= \phi_1\, u_1 \\ u_2 &= \phi_1\, u_1 & v_2 &= \phi_1\, u_1\, v_1 \end{aligned}$$

which can be simplified to:

$$\begin{aligned} \phi_3(u_3) &= \phi_2\,(\phi_1\, u_3) \ \textit{composition of functions} \\ \psi_3 * (u_3\, v_3) &= \psi_2\,(\phi_1\, u_3)\,(\psi_1\, u_3\, v_3) \ \textit{composition of their differentials} \end{aligned}$$

Types !

Programs and variable are typed
by logical formulas which describe their behavior

$$A \rightsquigarrow \exists \overbrace{x : \mathbb{W}(A)}^{\text{witness}}, \forall \underbrace{u : \mathbb{C}(A)}_{\text{opponent}}, A_D[x, u]$$

Witness and counter types :

$$\mathbb{C}(A \Rightarrow B) = \mathbb{C}(A) \times \mathbb{C}(B)$$

$$\mathbb{W}(A \Rightarrow B) = (\mathbb{W}(A) \Rightarrow \mathbb{W}(B)) \times (\mathbb{W}(A) \Rightarrow \mathbb{C}(B) \Rightarrow \mathbb{C}(A))$$

Types !

Programs and variable are typed
by logical formulas which describe their behavior

$$A \rightsquigarrow \exists \overbrace{x : \mathbb{W}(A)}^{\text{global witness}}, \forall \underbrace{u : \mathbb{C}(A)}_{\text{local opponent}}, A_D[x, u]$$

Witness and counter for implication types :

$$\mathbb{C}(A \Rightarrow B) = \mathbb{W}(A) \times \mathbb{C}(B)$$

$$\mathbb{W}(A \Rightarrow B) = \overbrace{(\mathbb{W}(A) \Rightarrow \mathbb{W}(B))}^{\text{function}} \times \left(\mathbb{W}(A) \Rightarrow \underbrace{\mathbb{C}(B) \Rightarrow \mathbb{C}(A)}_{\text{reverse derivative}} \right)$$

Reverse Mode differentiation:

$$\text{Functorial : } (h, \overleftarrow{D} h) : (A \Rightarrow B) \times (A \Rightarrow B^{\perp} \multimap A^{\perp})$$

However:

- Having the same type does not mean you're the same program.
- We (linear logicians) know what program differentiation is.

The computational Dialectica : a reverse Differential λ-calculus

A computational Dialectica

Making Dialectica act on λ-terms instead of formulas:

An abstract multiset $\mathfrak{M}\,(-)$

$$\frac{}{\Gamma \vdash \varnothing : \mathfrak{M}\, A} \qquad \frac{\Gamma \vdash m_1 : \mathfrak{M}\, A \qquad \Gamma \vdash m_2 : \mathfrak{M}\, A}{\Gamma \vdash m_1 \circledast m_2 : \mathfrak{M}\, A}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \{t\} : \mathfrak{M}\, A} \qquad \frac{\Gamma \vdash m : \mathfrak{M}\, A \qquad \Gamma \vdash f : A \Rightarrow \mathfrak{M}\, B}{\Gamma \vdash m \gg= f : \mathfrak{M}\, B}$$

$$\begin{array}{lcl} \mathbb{W}(A \Rightarrow B) & := & (\mathbb{W}(A) \Rightarrow \mathbb{W}(B)) \\ & & \times(\mathbb{C}(B) \Rightarrow \mathbb{W}(A) \Rightarrow \mathfrak{M}\,\mathbb{C}(A)) \\ \mathbb{C}(A \Rightarrow B) & := & \mathbb{W}(A) \times \mathbb{C}(B) \end{array}$$

Pédrot's Dialectica Transformation

Soundness [Ped14]

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

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

A global and a local transformation

$$
\begin{array}{lcllcl}
x^\bullet & := & x & (\lambda x.\, t)^\bullet & := & (\lambda x.\, t^\bullet, \lambda \pi x.\, t_x\ \pi) \\
x_x & := & \lambda \pi.\, \{\pi\} & (\lambda x.\, t)_y & := & \lambda \pi.\, (\lambda x.\, t_y)\ \pi.1\ \pi.2 \\
x_y & := & \lambda \pi.\, \varnothing \text{ if } x \neq y & (t\ u)^\bullet & := & (t^\bullet.1)\ u^\bullet
\end{array}
$$

$$(t\ u)_y := \lambda \pi.\, (t_y\ (u^\bullet, \pi)) \circledast ((t^\bullet.2)\, \pi\, u^\bullet \gg\!= u_y)$$

Flashback: Differential λ-calculus [Ehrhard, Regnier 04]

Inspired by denotational models of Linear Logic in vector spaces of sequences, it introduces a differentiation of λ-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:

$$\Lambda^d : S, T, U, V ::= 0 \mid s \mid s + T$$
$$\Lambda^s : s, t, u, v ::= x \mid \lambda x.s \mid sT \mid \mathrm{D}s \cdot t$$

Operational Semantics:

$$(\lambda x.s)T \to_\beta s[T/x]$$
$$\mathrm{D}(\lambda x.s) \cdot t \to_{\beta_D} \lambda x.\frac{\partial s}{\partial x} \cdot t$$

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

The linear substitution ...

... which is not exactly a substitution

$$\frac{\partial y}{\partial x} \cdot t = \{ \begin{array}{l} t \; if \; x = y \\ 0 \; otherwise \end{array}$$

$$\frac{\partial}{\partial x}(tu) \cdot s = (\frac{\partial t}{\partial x} \cdot s)u + (\mathrm{D}t \cdot (\frac{\partial u}{\partial x} \cdot s))u$$

$$\frac{\partial}{\partial x}(\lambda y.s) \cdot t = \lambda y.\frac{\partial s}{\partial x} \cdot t$$

$$\frac{\partial}{\partial x}(\mathrm{D}s \cdot u) \cdot t = \mathrm{D}(\frac{\partial s}{\partial x} \cdot t) \cdot u + \mathrm{D}s \cdot (\frac{\partial u}{\partial x} \cdot t)$$

$$\frac{\partial 0}{\partial x} \cdot t = 0$$

$$\frac{\partial}{\partial x}(s + u) \cdot t = \frac{\partial s}{\partial x} \cdot t + \frac{\partial u}{\partial x} \cdot t$$

$\frac{\partial s}{\partial x} \cdot t$ represents s where x is linearly (i.e. one time) substituted by t.

The linear substitution ...

The computational Dialectica

$$\frac{\partial y}{\partial x} \cdot t = \begin{cases} t \; if \; x = y \\ 0 \; otherwise \end{cases} \qquad \frac{\partial}{\partial x}(tu) \cdot s = (\frac{\partial t}{\partial x} \cdot s)u + (\mathrm{D}t \cdot (\frac{\partial u}{\partial x} \cdot s))u$$

$$x_y \cdot \pi = \begin{cases} \pi \; if \; x = y \\ \emptyset \; otherwise \end{cases} \qquad (t\ u)_y := \lambda\pi.\,(t_y\,(u^\bullet, \pi)) \circledast ((t^\bullet.2)\,\pi\,u^\bullet \gg\!= u_y)$$

$$\frac{\partial}{\partial x}(\lambda y.s) \cdot t = \lambda y.\frac{\partial s}{\partial x} \cdot t \qquad \frac{\partial}{\partial x}(\mathrm{D}s \cdot u) \cdot t = \mathrm{D}(\frac{\partial s}{\partial x} \cdot t) \cdot u + \mathrm{D}s \cdot (\frac{\partial u}{\partial x} \cdot t)$$

$$\frac{\partial 0}{\partial x} \cdot t = 0 \qquad \frac{\partial}{\partial x}(s + u) \cdot t = \frac{\partial s}{\partial x} \cdot t + \frac{\partial u}{\partial x} \cdot t$$

Tracking differentiation in Dialectica

Soundness [Ped14]

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

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

Tracking differentiation in Dialectica

Soundness [Ped14]

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

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

That's reverse differentiation

- $(\_)^\bullet.2$ obeys the chain rule, $(\_)^\bullet$ is the functorial differentiation.
- t_x is contravariant in x, representing a reverse linear substitution.

Tracking differentiation in Dialectica

Soundness [Ped14]

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

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

That's reverse differentiation

- $(\_)^\bullet.2$ obeys the chain rule, $(\_)^\bullet$ is the functorial differentiation.
- t_x is contravariant in x, representing a reverse linear substitution.

Theorem [K. Pédrot 22]

$$[\![u \gg\!= t_x[\Gamma \leftarrow \overrightarrow{r^\bullet}]]\!] \equiv_{\beta,\eta} \lambda z.\,([\![u]\!]\,((\partial x.t[\Gamma \leftarrow \overrightarrow{r}])z))$$

A Linear Logic Refinement

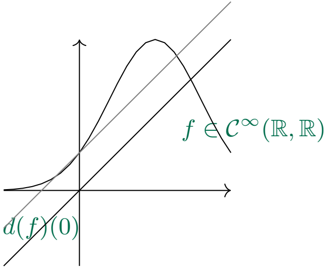
Differential Linear Logic

$$\frac{\vdash \ell : A \multimap B}{\vdash \ell : !A \multimap B}\, d$$

A linear proof
is in particular non-linear.

$$\frac{\vdash f : !A \multimap B}{\vdash D_0 f : A \multimap B}\, \bar{d}$$

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



Differential interaction nets, Ehrhard and Regnier, TCS (2006)

Exponential rules of Differential Linear Logic

$$\frac{\vdash \Gamma}{\vdash \Gamma, cst_1 : ?A}\, w \qquad \frac{\vdash \Gamma, f : ?A, g : ?A}{\vdash \Gamma, f.g : ?A}\, c \qquad \frac{\vdash \Gamma, \ell : A}{\vdash \Gamma, \ell : ?A}\, d$$

$$\frac{\vdash \Gamma}{\vdash \Gamma, \delta_0 : !A}\, \bar{w} \qquad \frac{\vdash \Gamma, \phi : !A \qquad \vdash \Delta, \psi : !A}{\vdash \Gamma, \Delta, \psi * \phi : !A}\, \bar{c} \qquad \frac{\vdash \Gamma, x : A}{\vdash \Gamma, D_0(\_)(x) : !A}\, \bar{d}$$

$$\frac{?\Gamma \vdash x : A}{?\Gamma \vdash \delta_x : !A}\, p$$

Dialectica factorizes through Linear Logic

The call by name arrow

$A \Rightarrow B := !A \multimap B := (!A)^{\perp} \otimes B$

$$
\begin{array}{lcl}
\mathbb{W}(A^{\perp}) & := & \mathbb{C}(A) \\
\mathbb{W}(A \oplus B) & := & \mathbb{W}(A) + \mathbb{W}(B) \\
\mathbb{W}(!A) & := & \mathbb{W}(A)
\end{array}
\qquad
\begin{array}{lcl}
\mathbb{C}(A^{\perp}) & := & \mathbb{W}(A) \\
\mathbb{C}(A \oplus B) & := & \mathbb{C}(A) \times \mathbb{C}(B) \\
\mathbb{C}(!A) & := & \mathbb{W}(A) \Rightarrow \mathbb{C}(A)
\end{array}
$$

$$
\begin{array}{lcl}
\mathbb{W}(A \otimes B) & := & \mathbb{W}(A) \times \mathbb{W}(B) \\
\mathbb{C}(A \otimes B) & := & (\mathbb{W}(A) \Rightarrow \mathbb{C}(B)) \times (\mathbb{W}(B) \Rightarrow \mathbb{C}(A))
\end{array}
$$

$$
\begin{tikzcd}
& \mathrm{LL} \arrow[dr, "\mathbb{W}\ \mathbb{C}"] & \\
\lambda^{+,\times} \arrow[ur, "[\![-]\!]_e"] \arrow[rr, "\mathbb{W}\qquad\mathbb{C}"'] & & \lambda^{+,\times}
\end{tikzcd}
$$

Valeria de Paiva, 1989, A dialectica-like model of linear logic.

Dialectica factorizes through Differential Linear Logic

Witnesses are functorial reverse derivative

$\mathbb{W}(A \Rightarrow B) = (\mathbb{W}(A) \Rightarrow \mathbb{W}(B)) \times (\mathbb{W}(A) \Rightarrow \mathbb{C}(B) \Rightarrow \mathbb{C}(A))$

$$
\begin{array}{lclclcl}
\mathbb{W}(!A) & := & !\mathbb{W}(A) & \quad & \mathbb{C}(!A) & := & !\mathbb{W}(A) \multimap \mathbb{C}(A) \\
\mathbb{W}(A \otimes B) & := & \mathbb{W}(A) \otimes \mathbb{W}(B) & & & & \\
\mathbb{C}(A \otimes B) & := & (\mathbb{W}(A) \multimap \mathbb{C}(B)) \oplus (\mathbb{W}(B) \multimap \mathbb{C}(A)) & & & & \\
\mathbb{W}(A \multimap B) & := & (\mathbb{W}(A) \multimap \mathbb{W}(B)) \,\&\, (\mathbb{C}(B) \multimap \mathbb{C}(A)) & & & & \\
\mathbb{C}(A \multimap B) & := & \mathbb{W}(A) \otimes \mathbb{C}(B) & & & &
\end{array}
$$

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

$$
\dfrac{\dfrac{\dfrac{}{\vdash A, A^{\perp}}\,\mathrm{ax}}{\vdash A, !A^{\perp}}\,\bar{d} \qquad \dfrac{}{\vdash ?A, !A^{\perp}}\,\mathrm{ax}}{\vdash ?A, A, !A^{\perp}}\,\bar{c} \qquad \dfrac{\pi}{\Gamma \vdash ?A}}{\Gamma \vdash ?A, A}\,\mathrm{cut}
$$

Dialectica factorizes through Differential Linear Logic

The economical translation

$$\begin{aligned}
[\![A \Rightarrow B]\!]_e &:= \,!A \multimap B \\
[\![A \times B]\!]_e &:= A \,\&\, B \\
[\![A + B]\!]_e &:= A \oplus B
\end{aligned}$$

$$\begin{tikzcd}
\mathsf{ILL} \arrow[r, "\mathbb{W} \qquad \mathbb{C}"] & \mathsf{IDiLL} \arrow[d, "\cdots"] \\
\lambda^{+,\times} \arrow[u, "{[\![\text{-}]\!]_e}"] \arrow[r, "\mathbb{W} \qquad \mathbb{C}"'] & \lambda^{+,\times}
\end{tikzcd}$$

IDILL : Intuitionnistic Differential Linear Logic ? Oh no ...

Dialectica is differentiation in categories

What's categorical differentiation ?

To cook a good differential category, one needs :

- A category of regular/continuous/non-linear functions
$$\mathbb{C}(A, B) = !A \multimap B \ .$$
- A category of linear functions, in which differentiation embeds
$$\mathscr{L}(A, B) = A \multimap B.$$
- Something which linearizes :
$$\bar{d} : A \to !A$$
- A notion of <u>duality</u>, if one wants to encode <u>reverse</u>. differentiation.

$\rightsquigarrow$ Basically, one wants a categorical model of DiLL.

Dialectica categories

Categories representing specific relations

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

- Objects : relations $\alpha \subseteq U \times X$, $\beta \subseteq V \times Y$.
- Maps from α to β :

$$(f : U \to V, F : U \times Y \to X)$$

- Composition : the chain rule !

Consider

$$\begin{array}{rlcl} & (f,F): \quad \alpha \subseteq (A,X) & \to & \beta \subseteq (B,Y) \\ \text{and} & (g,G): \quad \beta \subseteq (B,Y) & \to & \gamma \subseteq (C,Z) \end{array}$$

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 $*$-autonomous differential category : from $f : !A \to B$ 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 DiLL such that $\mathcal{L}_!$ has finite limits:

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

We have an obvious forgetful functor:

$$\mathcal{U} : \begin{cases} \mathscr{D}(\mathscr{L}_!) & \to & \mathscr{L}_! \\ \alpha \subseteq A \times X & \mapsto & A \\ (f, F) & \mapsto & f \end{cases}$$

which is left adjoint to $\mathcal{R}$, forming a reflection on $\mathscr{L}_{oc}$.

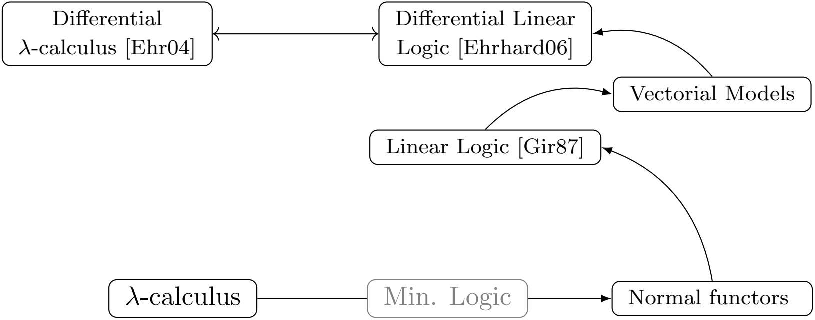
To be declined in reverse/cartesian differential categories...

Recap

Programs	**Logic**	**Semantics**
`fun (x:A)-> (t:B)`	Proof of $A \vdash B$	$f : A \to B$.
Types	Formulas	Objects
Execution	Cut-elimination	Equality

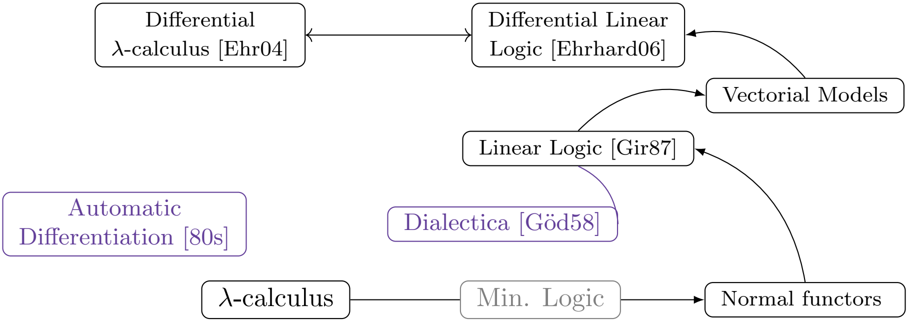
Recap

Programs	**Logic**	**Semantics**
`fun (x:A)-> (t:B)`	Proof of $A \vdash B$	$f : A \to B$.
Types	Formulas	Objects
Execution	Cut-elimination	Equality



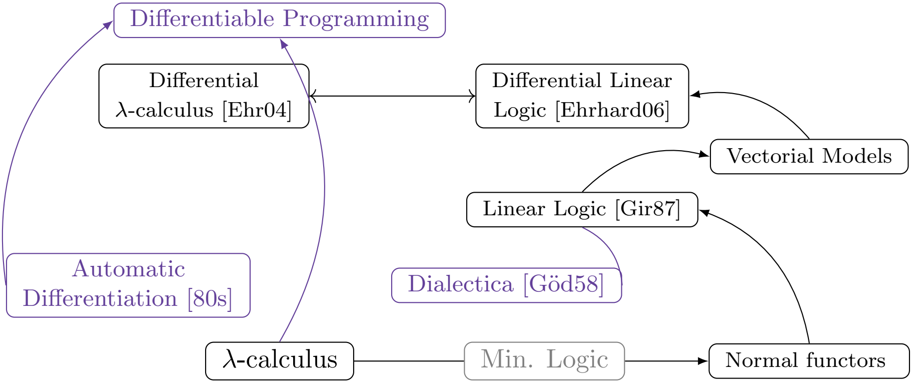
Recap

Programs	**Logic**	**Semantics**
`fun (x:A)-> (t:B)`	Proof of $A \vdash B$	$f : A \to B$.
Types	Formulas	Objects
Execution	Cut-elimination	Equality



Recap

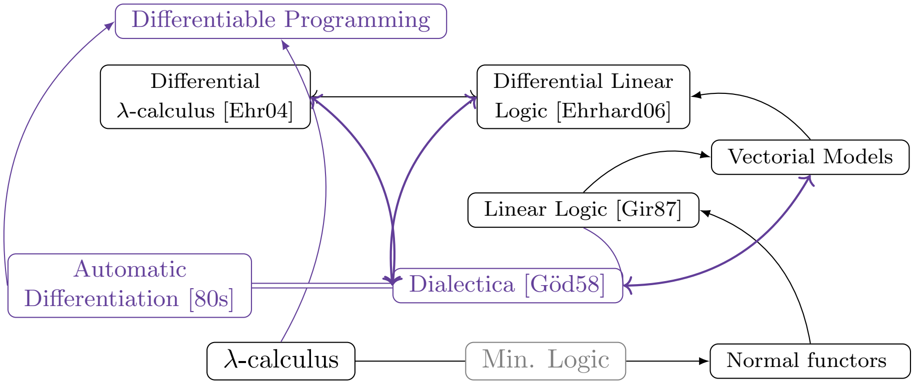
Programs	**Logic**	**Semantics**
`fun (x:A)-> (t:B)`	Proof of $A \vdash B$	$f : A \to B$.
Types	Formulas	Objects
Execution	Cut-elimination	Equality



A good point for logicians : Gödel invented Dialectica 40 years before reverse differentiation was put to light

Recap

Programs	**Logic**	**Semantics**
`fun (x:A)-> (t:B)`	Proof of $A \vdash B$	$f : A \to B$.
Types	Formulas	Objects
Execution	Cut-elimination	Equality



Conclusion and applications

Take home message:

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

A new semantical correspondance between computations and mathematics :
intentional meaning of program is local behaviour of functions.

Program	Proof	Function
Quantitative	Resources	Linearity
Control	**Classical Principles**	**Differentiation**
	Intentional	**Local**
	Extentional	**Global**

Related work and applications:

- Markov's principle and delimited continuations on positive formulas.
- Proof mining and backpropagation.

Dialectica is differentiation ...

... We knew it already !

The codereliction 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. $? \oplus_x (?A(x) \otimes ?B(x))$) where the presence of "?" indicates that the proof possibly used weakening (`efq` or `throw`) or contraction (`catch`), a linear proof of A purged from the occurrences of its "?" connective can be extracted (meaning for the example above a proof of $\oplus_x (A(x) \otimes B(x))$). Interestingly, the removal of the "?", i.e. the steps from $?P$ to P, correspond to applying the codereliction rule of differential proof nets [24].

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

Hugo Herbelin, "An intuitionistic logic that proves Markov's principle", LICS '10 .

Differentiation and delimited continuations

Herbelin Lics'10

Markov's principle is proved by allowing `catch` and `throw` operations on hereditary positive formulas.

$$
\begin{prooftree}
\AxiomC{}
\RightLabel{AXIOM}
\UnaryInfC{$a : \neg\neg T \vdash_{\alpha:T} a : \neg\neg T$}
\AxiomC{}
\RightLabel{AXIOM}
\UnaryInfC{$b : T \vdash_{\alpha:T} b : T$}
\RightLabel{THROW}
\UnaryInfC{$b : T \vdash_{\alpha:T} \mathtt{throw}_\alpha\, \mathtt{b} : \bot$}
\RightLabel{$\rightarrow_I$}
\UnaryInfC{$\vdash_{\alpha:T} \lambda b.\mathtt{throw}_\alpha\, \mathtt{b} : \neg \mathtt{T}$}
\RightLabel{$\rightarrow_E$}
\BinaryInfC{$a : \neg\neg T \vdash_{\alpha:T} a\,(\lambda b.\mathtt{throw}_\alpha\, \mathtt{b}) : \bot$}
\RightLabel{$\bot_E$}
\UnaryInfC{$a : \neg\neg T \vdash_{\alpha:T} \mathtt{efq}\ \mathtt{a}\,(\lambda \mathtt{b}.\mathtt{throw}_\alpha\, \mathtt{b}) : \mathtt{T}$}
\RightLabel{CATCH}
\UnaryInfC{$a : \neg\neg T \vdash \mathtt{catch}_\alpha\ \mathtt{efq}\ \mathtt{a}\,(\lambda \mathtt{b}.\mathtt{throw}_\alpha\, \mathtt{b}) : \mathtt{T}$}
\RightLabel{$\rightarrow_I$}
\UnaryInfC{$\vdash \lambda a.\mathtt{catch}_\alpha\ \mathtt{efq}\ \mathtt{a}\,(\lambda \mathtt{b}.\mathtt{throw}_\alpha\, \mathtt{b}) : \neg\neg \mathtt{T} \rightarrow \mathtt{T}$}
\end{prooftree}
$$

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*

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

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

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

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

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.

$$(+++)\ \forall u \in U, v_1, v_2 \in V_u, k \in \mathbb{N} \big(|G(u, v_1)|, |G(u, v_2)| \le 2^{-\Phi u k} \rightarrow d_V(v_1, v_2) \le 2^{-k}\big).$$

Proof Mining

Extracting quantitative information from proofs.

$$\forall u, v_1 v_2, \mathrm{Pol}(u, v_1) = \mathrm{Pol}(u, v_2) \to v_1 = v_2$$

$$\Downarrow$$

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

$$\Downarrow$$

$$\exists \phi, \forall u, k, v_1 v_2, \|G(u, v_1) - G(u, v_2)\| < 2^{-\phi(u,k)} \to d_V(v_1, v_2) < 2^{-k}.$$

Proof Mining

Markov's principle and the independence of premises are necessary for most of **mathematical analysis proofs** :

Proof mining allows to refine these proofs by taking away these principles as guaranteed by (some variant of) Dialectica's transformation.

Conjecture

Does it differentiate the function $(\epsilon \to \eta)$ in :

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

?

Is proof mining (based on) reverse differentiation applied to proofs?

What else can we explain by differentiation ?