

TLLA 2021

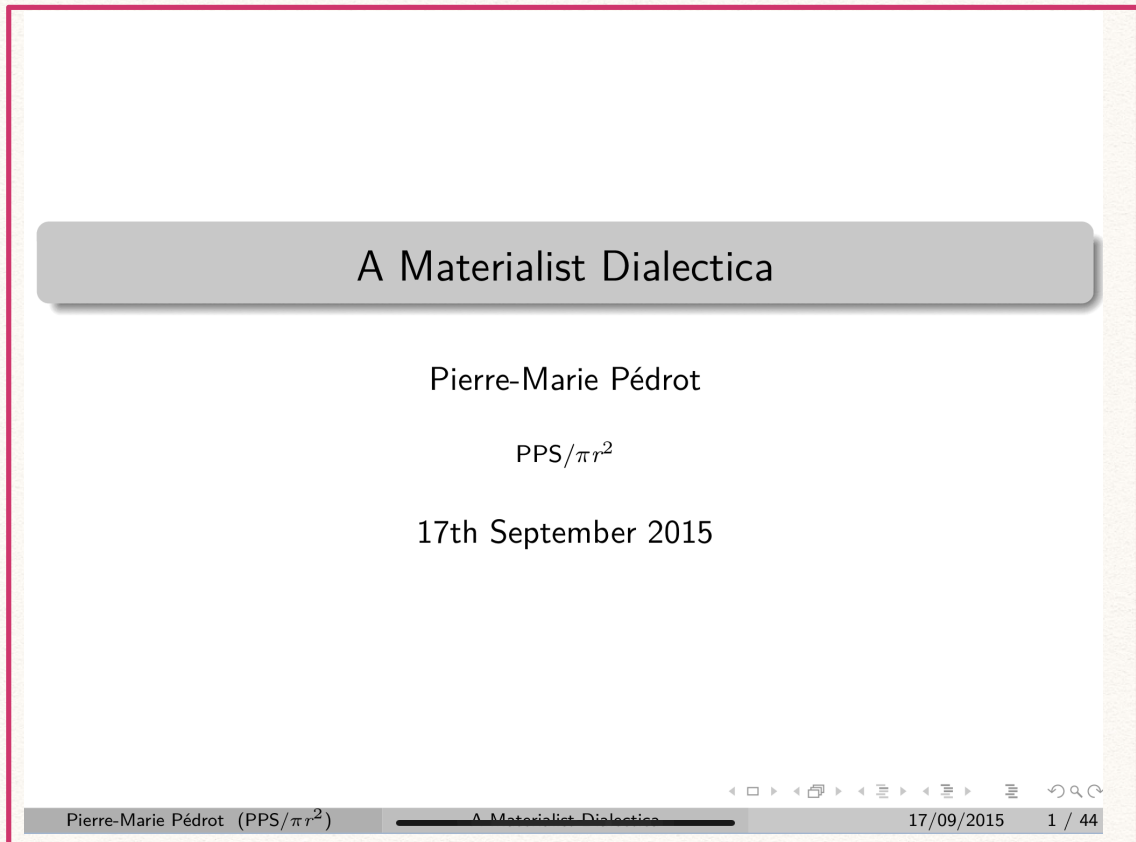
δ is for Dialectica

joint work with
Pierre-Marie
Pédrot

Marie Kerjean
CNRS & LIPN



6 years ago...



Pierre-Marie PhD defense.

6 years ago...

Towards the end of the talk :

Finally!

$$\begin{aligned}x^\bullet &\equiv x \\(\lambda x. t)^\bullet &\equiv \begin{cases} \lambda x. t^\bullet \\ \lambda x \pi. t_x \pi \end{cases} \\(tu)^\bullet &\equiv (\text{fst } t^\bullet) u^\bullet\end{aligned}$$

$$x_x \equiv \lambda \pi. \{\pi\}$$

$$y_x \equiv \lambda \pi. \emptyset$$

$$(\lambda y. t)_x \equiv \lambda(y, \pi). t_x \pi$$

6 years ago...

$$x_x \equiv \lambda\pi. \{\pi\}$$

$$y_x \equiv \lambda\pi. \emptyset$$

$$t^\bullet \equiv \dots$$

$$(\lambda y. t)_x \equiv \lambda (y, \pi) \cdot t_x \pi$$

Marie : «That's differential λ - calculus !
It must be ! »

And then life



2 years ago...

Reverse-Mode AD in a Functional Framework: Lambda the Ultimate Backpropagator

BARAK A. PEARLMUTTER

Hamilton Institute

and

JEFFREY MARK SISKIND

Purdue University

We show that reverse-mode AD (Automatic Differentiation)—a generalized gradient-calculation operator—can be incorporated as a first-class function in an augmented lambda calculus, and therefore into a functional-programming language. Closure is achieved, in that the new operator can be applied to any expression in the augmented language, yielding an expression in that language. This requires the resolution of two major technical issues: (a) how to transform nested lambda expressions, including those with free-variable references, and (b) how to support self application of the AD machinery. AD transformations preserve certain complexity properties, among them that the reverse phase of the reverse-mode AD transformation of a function have the same temporal complexity as the original untransformed function. First-class unrestricted AD operators increase the expressive power available to the numeric programmer, and may have significant practical implications for the construction of numeric software that is robust, modular, concise, correct, and efficient.

Categories and Subject Descriptors: D.3.2.a [Programming Languages]: Language Classifications—*Applicative (functional) languages*; G.1.4.b [Numerical Analysis]: Quadrature and Numerical Differentiation—*Automatic differentiation*

General Terms: Experimentation, Languages, Performance

Additional Key Words and Phrases: closures, derivatives, forward-mode AD, higher-order AD, higher-order functional languages, Jacobian, program transformation, reflection

1. INTRODUCTION

When you first learned calculus, you learned how to take the derivatives of some simple expressions. Later you learned the chain rule: the ability to take the derivative of the composition of two functions. The fact that the space of expressions can

Pearlmutter was supported, in part, by Science Foundation Ireland grant 00/PI.1/C067 and the Higher Education Authority of Ireland. Siskind was supported, in part, by NSF grant CCF-0438806. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

Authors' addresses: B. A. Pearlmutter, Hamilton Institute and Department of Computer Science, NUI Maynooth, Co. Kildare, Ireland; email: barak@cs.nuim.ie; J. M. Siskind (contact author), School of Electrical and Computer Engineering, Purdue University, 455 Northwestern Avenue, Room 330, West Lafayette, IN 47907-2035 USA; email: qobi@purdue.edu.

Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.

© 20TBD ACM 0164-0925/20TBD/0500-0001 \$5.00

ACM Transactions on Programming Languages and Systems, Vol. TBD, No. TBD, TBD 20TBD, Pages 1-35.

PMP : « That's nothing but Dialecta ! »

"but ugly because untyped"

It worked !

1. The historical Dialectica.

And the chain rule

2. The categorical Dialectica.

And reverse differentiation

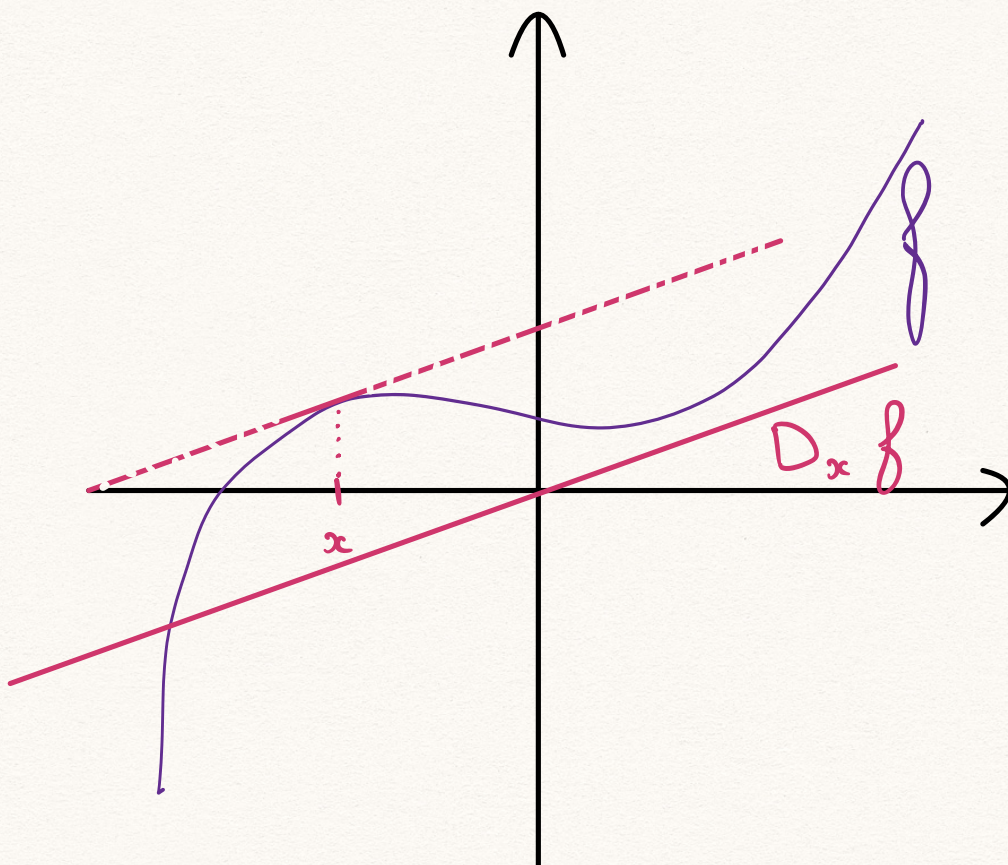
3. The computational Dialectica.

And sums

4. Applications

Dialectica and the chain rule

Differentiation



The chain rule :

$$D_x (g \circ f) = D_{f(x)} g \circ D_x f$$

Dialectica

I. Für $k = 0$ sei $F' = F$.

II. Es sei $F' = (\exists y) (z) A (y, z, x)$ und
 $G' = (\exists v) (w) B (v, w, u)$ bereits definiert,

dann ist *per definitionem* :

1. $(F \wedge G)' = (\exists yv) (zw) [A (y, z, x) \wedge B (v, w, u)]$.
2. $(F \vee G)' = (\exists yvt) (zw) [t=0 \wedge A (y, z, x) \cdot \vee \cdot t=1 \wedge B (v, w, u)]$.
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. $(\neg F)' = (\exists \bar{Z}) (y) \neg A (y, \bar{Z}(y), x)$.

*Kurt Gödel. Über eine bisher noch nicht benützte
Erweiterung des finiten Standpunktes.
Dialectica, 12:280–287, 1958.*

Dialectica


A translation acting on intuitionistic arithmetic:

$$A \rightsquigarrow \exists u, \forall x, \boxed{A_D}[u, x]$$

- Validates Markov's principle $\neg\forall x.\neg A \rightarrow \exists x.A$
- Validates the independance of premises
 $(A \rightarrow \exists x.B) \rightarrow \exists x.(A \rightarrow B)$
- Numerous applications in logic
 - Soundness results
 - Proof mining
- Relies on decidability lemmas

Dialectica

fresh variables


$$(A \rightarrow B)_D [f, g; x, y] := A_D [x, gxy] \rightarrow B_D [fx, y]$$

Justifications :

- Begin with:

$$\exists x \forall y A_D [x, y] \rightarrow \exists v \forall w B_D [v, w]$$

- Apply the least unconstructive prenexation:

$$\forall x \exists r \forall w \exists y (A_D [x, y] \rightarrow B_D [v, w])$$

- Apply axiom of choice:

$$\exists f, g \forall x, w (A_D [x, fwx] \rightarrow B_D [gx, w])$$

A mysterious construction

CHAPTER VI

Gödel's Functional ("Dialectica") Interpretation

Jeremy Avigad

*Department of Philosophy, Carnegie Mellon University
Pittsburgh, PA 15213*

Solomon Feferman

*Departments of Mathematics and Philosophy, Stanford University
Stanford, CA 94305*

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 $\varphi \rightarrow \psi$ 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.

The Chain rule

$$(A \rightarrow B)_D[f, g; u, v] = A_D[u, guv] \rightarrow B_D[fu, v]$$

$$(B \rightarrow C)_D[f', g', u', v'] = B_D[u', g'u'v'] \rightarrow C_D[f'u', v']$$

$$(A \rightarrow C)_D[f'', g'', u'', v''] = A_D[u'', g''u''v''] \rightarrow C_D[f''u'', v'']$$

equations

$$\begin{cases} f'u' = f''u'' \\ v' = v'' \end{cases}$$

$$\begin{cases} u'' = u \\ g''u''v'' = guv \end{cases}$$

$$\begin{cases} fu = u' \\ v = g'u'v' \end{cases}$$

Results

1. $g''u''v'' = gu''(g'(fu'')v'')$ Chain rule

2. $f''u'' = f'(fu)$ Functoriality

Types!

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

	$\mathbb{W}(-)$	$\mathbb{C}(-)$
$A \rightarrow B$	$\left\{ \begin{array}{c} \mathbb{W}(A) \rightarrow \mathbb{W}(B) \\ \times \\ \mathbb{W}(A) \rightarrow \mathbb{C}(B) \rightarrow \mathbb{C}(A) \end{array} \right.$	$\mathbb{W}(A) \times \mathbb{C}(B)$
1	1	1
$A \times B$	$\mathbb{W}(A) \times \mathbb{W}(B)$	$\mathbb{C}(A) + \mathbb{C}(B)$
0	1	1
$A + B$	$\mathbb{W}(A) + \mathbb{W}(B)$	$\mathbb{C}(A) \times \mathbb{C}(B)$

Types!

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

$$\begin{aligned} \mathbb{W}(A \rightarrow B) &= \\ \mathbb{W}(A) &\rightarrow \mathbb{W}(B) \times \\ \mathbb{W}(A) &\rightarrow \mathbb{C}(B) \rightarrow \mathbb{C}(A) \end{aligned}$$

Types!

$$\begin{aligned} \mathbb{W}(A \rightarrow B) &= \underbrace{\mathbb{W}(A) \rightarrow \mathbb{W}(B)}_{\substack{\uparrow \\ x}} \times \underbrace{\mathbb{W}(A) \rightarrow \mathbb{C}(B) \rightarrow \mathbb{C}(A)}_{\substack{\uparrow \\ D_x f}} \end{aligned}$$

Diagram illustrating the type decomposition of $\mathbb{W}(A \rightarrow B)$. The expression is shown as a product of two types. The first type, $\mathbb{W}(A) \rightarrow \mathbb{W}(B)$, is associated with the function f via a red arrow. The second type, $\mathbb{W}(A) \rightarrow \mathbb{C}(B) \rightarrow \mathbb{C}(A)$, is associated with the derivative $D_x f$ via a red arrow. A red bracket under the first $\mathbb{W}(A)$ in the second type is labeled x with an upward arrow. A red bracket under the entire second type is labeled $D_x f$ with an upward arrow.

$$Df_x : \mathbb{C}(B) \rightarrow \mathbb{C}(A)$$

Why is Df contravariant ?

Dialectica and reverse differentiation

The real inventors of deep learning



Reverse differentiation

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

$$\begin{array}{lll} \text{E.g. : } z = y + \cos(x^2) & x_1 = x_0^2 & x'_1 = 2x_0x'_0 \\ & x_2 = \cos(x_1) & x'_2 = -x'_0 \sin(x_0) \\ & z = y + x_2 & z' = y' + 2x_2x'_2 \end{array}$$

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

Reverse AD, higher-order

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

- Forward mode:

$$(u, 1) \rightarrow (f(u), D_u f) \rightarrow (g(u), D_{f(u)} g)$$

- Reverse mode:

$$u \rightarrow f(u) \rightarrow g(u) \rightarrow D_{f(u)} g \rightarrow D_u f$$

Complexity considerations:

- Forward is faster for

$$g \circ f : \mathbb{R} \rightarrow \mathbb{R}^2 \dots \rightarrow \mathbb{R}^n$$

- Reverse is faster for

$$g \circ f : \mathbb{R}^n \rightarrow \dots \rightarrow \mathbb{R}^2 \rightarrow \mathbb{R}$$

Reverse AD, functorially

Key idea: reverse derivatives are typed by linear negations

Consider $f : \mathbb{R}^n \rightarrow \mathbb{R}$ a function variable. [BMP 20]

$$\overleftarrow{D}(f) : \begin{cases} \mathbb{R}^n \times \mathbb{R}^\perp \rightarrow \mathbb{R} \times \mathbb{R}^{n\perp} \\ (a, x) \mapsto (f(a), (v \mapsto x \cdot (D_a f \cdot v))) \end{cases}$$

$$f : A \Rightarrow B \quad D_a f : A \multimap B$$

$$\overleftarrow{D}_a f : \begin{cases} B^\perp \rightarrow A^\perp \\ \ell \rightarrow \ell \circ D_a f \end{cases}$$

Differentiable programming:

- Brunel & Mazza & Pagani, backpropagation in the simply λ -calculus with linear negation, 2020.
- Elliot, the simple essence of AD, 2018

Reverse AD, functorially

$$\begin{aligned} \mathbb{W}(A \rightarrow B) &= \mathbb{W}(A) \rightarrow \mathbb{W}(B) \times \mathbb{W}(A) \rightarrow \mathbb{C}(B) \rightarrow \mathbb{C}(A) \\ &\quad \text{CoDf} \end{aligned}$$

$$Df := a \mapsto (\ell \mapsto (\ell \circ D_a f))$$

Dialectica categories

$\text{Dial}(\mathcal{C}) :=$

- objects relations $A \xrightarrow{\alpha} U \times X$

- morphisms :

$$A \xrightarrow{\alpha} U \times X \quad \xrightarrow{(f, F)} \quad B \xrightarrow{\beta} V \times Y$$

with $f: U \rightarrow V$ $F: U \times Y \rightarrow X$

such that $u \times F(u, y) \Rightarrow f(u) \beta y$

- composition

show that DC is a category. Given two maps $(f, F): \alpha \rightarrow \beta$ and $(g, G): \beta \rightarrow \gamma$ their composition $(g, G) \circ (f, F)$ is $gf: U \rightarrow W$ in the first coordinate and $G \circ F: U \times Z \rightarrow X$ given by:

$$U \times Z \xrightarrow{f \times Z} U \times U \times Z \xrightarrow{U \times f \times Z} U \times V \times Z \xrightarrow{U \times G} U \times Y \xrightarrow{F} X$$

Dialectica categories:

\mathcal{C} a $*$ -autonomous differential category:

- Model of $\lambda\lambda$
- biproduct
- Deriving transformation

$$f \in \mathcal{C}(!A, B), Df \in \mathcal{C}(A \otimes !A, B)$$

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

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

Thm: One has a functor

$$\mathcal{C}^! \rightarrow \text{Dial}(\mathcal{C})$$

$$A \mapsto (!A, A^\perp)$$

$$f \mapsto (f \circ \mu, \overset{\leftarrow}{D}(f))$$

The computational Dialectica

*more than the
chain rule..*

Not enough.

The chain rule as a functional equation

Shiri Artstein-Avidan ^{a,1}, Hermann König ^{b,*}, Vitali Milman ^{a,2}

^a School of Mathematical Sciences, Tel Aviv University, Ramat Aviv, Tel Aviv 69978, Israel

^b Mathematisches Seminar, Universität Kiel, 24098 Kiel, Germany

Received 18 June 2010; accepted 7 July 2010

Available online 17 July 2010

Communicated by J. Bourgain

Abstract

We consider operators T from $C^1(\mathbb{R})$ to $C(\mathbb{R})$ satisfying the “chain rule”

$$T(f \circ g) = (Tf) \circ g \cdot Tg, \quad f, g \in C^1(\mathbb{R}),$$

and study under which conditions this functional equation admits only the derivative or its powers as solutions. We also consider T operating on other domains like $C^k(\mathbb{R})$ for $k \in \mathbb{N}_0$ or $k = \infty$ and study the more general equation $T(f \circ g) = (Tf) \circ g \cdot Ag$, $f, g \in C^1(\mathbb{R})$ where both T and A map $C^1(\mathbb{R})$ to $C(\mathbb{R})$.

© 2010 Elsevier Inc. All rights reserved.

Theorem 1. *Let $T : \mathcal{D}(T) := C^1(\mathbb{R}) \rightarrow \text{Im}(T) \subset C(\mathbb{R})$ be an operation satisfying the chain rule*

$$T(f \circ g) = (Tf) \circ g \cdot Tg, \quad f, g \in \mathcal{D}(T). \quad (1)$$

Assume that T is non-degenerate in the sense of (4). Then there exists some $p \geq 0$ and a positive continuous function $H \in C(\mathbb{R})$ such that either

$$\left. \begin{aligned} & \text{or in the case of } p > 0, \\ & Tf = \frac{H \circ f}{H} |f'|^p \\ & Tf = \frac{H \circ f}{H} |f'|^p \operatorname{sgn}(f'). \end{aligned} \right\} \quad (5)$$

More than the chain rule, categorically

Definition 4.2 A *Cartesian (closed) differential category* is a Cartesian (closed) left-additive category having an operator $D(-)$ that maps a morphism $f : A \rightarrow B$ into a morphism $D(f) : A \times A \rightarrow B$ and satisfies the following axioms:

- D1. $D(f + g) = D(f) + D(g)$ and $D(0) = 0$
- D2. $D(f) \circ \langle h + k, v \rangle = D(f) \circ \langle h, v \rangle + D(f) \circ \langle k, v \rangle$ and $D(f) \circ \langle 0, v \rangle = 0$
- D3. $D(\text{Id}) = \pi_1$, $D(\pi_1) = \pi_1 \circ \pi_1$ and $D(\pi_2) = \pi_2 \circ \pi_1$
- D4. $D(\langle f, g \rangle) = \langle D(f), D(g) \rangle$
- D5. $D(f \circ g) = D(f) \circ \langle D(g), g \circ \pi_2 \rangle$
- D6. $D(D(f)) \circ \langle \langle g, 0 \rangle, \langle h, k \rangle \rangle = D(f) \circ \langle g, k \rangle$
- D7. $D(D(f)) \circ \langle \langle 0, h \rangle, \langle g, k \rangle \rangle = D(D(f)) \circ \langle \langle 0, g \rangle, \langle h, k \rangle \rangle$

More than functions

Definition 93 (Type translation). The translation on types is inductively defined in the table below. The $\mathbb{W}(-)$ and $\mathbb{C}(-)$ translations associate to a type of $\lambda^{\times+}$ another type of $\lambda^{\times+}$.

	$\mathbb{W}(-)$	$\mathbb{C}(-)$
$A \rightarrow B$	$\begin{cases} \mathbb{W}(A) \rightarrow \mathbb{W}(B) \\ \times \\ \mathbb{W}(A) \rightarrow \mathbb{C}(B) \rightarrow \mathbb{C}(A) \end{cases}$	$\mathbb{W}(A) \times \mathbb{C}(B)$
1	1	1
$A \times B$	$\mathbb{W}(A) \times \mathbb{W}(B)$	$\mathbb{C}(A) + \mathbb{C}(B)$
0	1	1
$A + B$	$\mathbb{W}(A) + \mathbb{W}(B)$	$\mathbb{C}(A) \times \mathbb{C}(B)$

- $(A + B)_D[w, z] := \text{match } w \text{ with } [u \mapsto A_D[u, \text{fst } z] \mid v \mapsto B_D[v, \text{snd } z]]$
- $(A \times B)_D[w, z] := \text{match } z \text{ with } [x \mapsto A_D[\text{fst } w, x] \mid y \mapsto B_D[\text{snd } w, y]]$
- $(A \rightarrow B)_D[w, z] := A_D[\text{fst } z, \text{snd } w (\text{fst } z) (\text{snd } z)] \rightarrow B_D[\text{fst } w (\text{fst } z), \text{snd } z]$

λ -terms as realizers :

Proposition 51. For all $\lambda^{\times+}$ -term $\vec{x} : \Gamma \vdash t : A$, the translated term $(\vec{x} : \Gamma \vdash t : A)^\bullet$ realizes A , that is:

$$\vec{x} : \mathbb{W}(\Gamma), \pi : \mathbb{C}(A) \mid \cdot \vdash A_D[(\Gamma \vdash t : A)^\bullet, \pi]$$

Dialectica acting on LL

	$\mathbb{W}(-)$	$\mathbb{C}(-)$
1	1	1
$A \otimes B$	$\mathbb{W}(A) \times \mathbb{W}(B)$	$\begin{cases} \mathbb{W}(A) \rightarrow \mathbb{C}(B) \\ \times \\ \mathbb{W}(B) \rightarrow \mathbb{C}(A) \end{cases}$
0	1	1
$A \oplus B$	$\mathbb{W}(A) + \mathbb{W}(B)$	$\mathbb{C}(A) \times \mathbb{C}(B)$
$!A$	$\mathbb{W}(A)$	$\mathbb{W}(A) \rightarrow \mathbb{C}(A)$
A^\perp	$\mathbb{C}(A)$	$\mathbb{W}(A)$

De Paiva's thesis [92]

ILL $\xrightarrow{\text{Dialectica}}$ λ^x -calc

Proposition 52. For any linear type A ,

$$u : \mathbb{W}(A), x : \mathbb{C}(A) \vdash_{\text{wf}} A_D[u, x]$$

As expected, correct proofs of linear logic are mapped to $\lambda^{\times+}$ -terms satisfying the usual orthogonality property.

Theorem 21 (Linear soundness). From every proof in linear logic of a formula A , one can construct a $\lambda^{\times+}$ -term p of type $\mathbb{W}(A)$ such that

$$\pi : \mathbb{C}(A) \mid \cdot \vdash A_D[p, \pi]$$

But $t \equiv_\beta t' \not\Rightarrow t^\bullet \not\equiv_\beta t'^\bullet$

Computational Dialectica

Pedrot thesis, 2015

Definition 104 (Term translation). Given a λ -term t and a variable x , we mutually define the translations t^\bullet and t_x by induction on t below.

$$\begin{aligned}x^\bullet &:= x \\(\lambda x. t)^\bullet &:= (\lambda x. t^\bullet, \lambda \pi x. t_x \pi) \\(t u)^\bullet &:= \text{fst } t^\bullet u^\bullet \\x_x &:= \lambda \pi. \{\pi\} \\x_y &:= \lambda \pi. \emptyset \\(\lambda y. t)_x &:= \lambda(y, \pi). t_x \pi \\(t u)_x &:= \lambda \pi. (\text{snd } t^\bullet \pi u^\bullet \gg \lambda \rho. u_x \rho) \odot (t_x (u^\bullet, \pi))\end{aligned}$$

Theorem 2 (Soundness [Péd14]). 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 new construction on types allowing sums and replacing decidability constructions

$$\frac{}{\Gamma \vdash \emptyset : \mathfrak{M} A} \quad \frac{\Gamma \vdash m_1 : \mathfrak{M} A \quad \Gamma \vdash m_2 : \mathfrak{M} A}{\Gamma \vdash m_1 \oplus m_2 : \mathfrak{M} A} \\ \frac{\Gamma \vdash t : A}{\Gamma \vdash \{t\} : \mathfrak{M} A} \quad \frac{\Gamma \vdash m : \mathfrak{M} A \quad \Gamma \vdash f : A \Rightarrow \mathfrak{M} B}{\Gamma \vdash m \gg f : \mathfrak{M} B}$$

Differentiation on terms

Two transformations for a higher-order differentiation

Differential λ -calculus, Ehrhard & Regnier

$$\Lambda^s := x \mid (t) u \mid \lambda x \cdot t \mid Dt \cdot u$$

$$D(\lambda x \cdot s) \cdot u \rightarrow_{\beta} \lambda x \cdot \left(\frac{\partial s}{\partial x} \cdot u \right)$$

$$\frac{\partial x}{\partial y} \cdot v = \begin{cases} v \\ 0 \end{cases} \quad \frac{\partial \lambda x \cdot t}{\partial y} \cdot v = \lambda x \cdot \left(\frac{\partial t}{\partial y} \cdot v \right)$$

$$\frac{\partial (t) u}{\partial y} \cdot v = \left(\frac{\partial t}{\partial y} \cdot v \right) u + \left(Dt \cdot \left(\frac{\partial u}{\partial y} \cdot v \right) \right) u$$

Computational Dialectica

Differential λ - calculus , CPS style

$$\frac{\partial x}{\partial y} \cdot v = \begin{cases} v \\ 0 \end{cases} \quad \frac{\partial \lambda x . t}{\partial y} \cdot v = \lambda x . \left(\frac{\partial t}{\partial y} \cdot v \right)$$

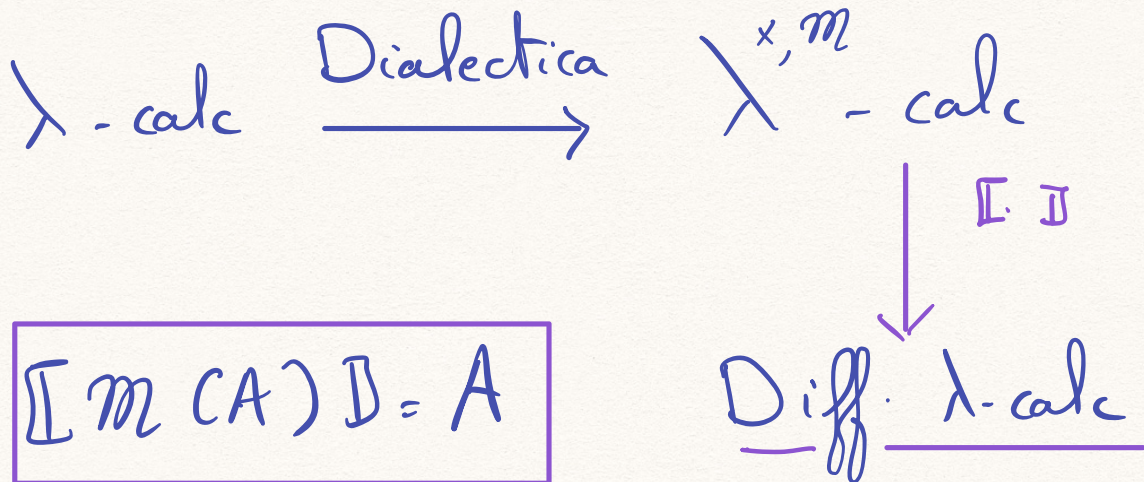
$$\frac{\partial (t) u}{\partial y} \cdot v = \left(\frac{\partial t}{\partial y} \cdot v \right) u + \left(Dt \cdot \left(\frac{\partial u}{\partial y} \cdot v \right) \right) u$$

$$y_x = \begin{cases} \lambda \pi \cdot \{\pi\} \\ \emptyset \end{cases} \quad (\lambda x . t)_y = \lambda (x, \pi) . t_y \pi$$

$$(tu)_x = \lambda \pi \cdot (t_x (u^\bullet, \pi) + (t_2^\bullet \pi u^\bullet >>= u_x))$$

reverse
chain rule

Computational Dialectica



$$\begin{array}{ll}
 \mathbb{I} \emptyset \mathbb{D} = 0 & \mathbb{I} t \oplus u \mathbb{D} = t + u \\
 \mathbb{I} \{t\} \mathbb{D} = t & \mathbb{I} t \gg f \mathbb{D} = (f) t
 \end{array}$$

Thm: $\forall t, \forall u$

$$\mathbb{I} t_x \mathbb{D} u \equiv_{\beta} \frac{\partial t}{\partial x} \cdot u$$

$$\mathbb{I} (\lambda x \cdot t)_2^{\bullet} \mathbb{D} u \equiv Dt \cdot u$$

Dialectica, from LL to DiLL

	$\mathbb{W}(-)$	$\mathbb{C}(-)$
1	1	1
$A \otimes B$	$\mathbb{W}(A) \otimes \mathbb{W}(B)$	$\begin{cases} \mathbb{W}(A) \multimap \mathbb{C}(B) \\ \& \\ \mathbb{W}(B) \multimap \mathbb{C}(A) \end{cases}$
0	0	0
$A \oplus B$	$\mathbb{W}(A) \oplus \mathbb{W}(B)$	$\mathbb{C}(A) \wp \mathbb{C}(B)$
$!A$	$!\mathbb{W}(A)$	$!\mathbb{W}(A) \multimap \mathbb{C}(A)$
A^\perp	$\mathbb{C}(A)$	$\mathbb{W}(A)$

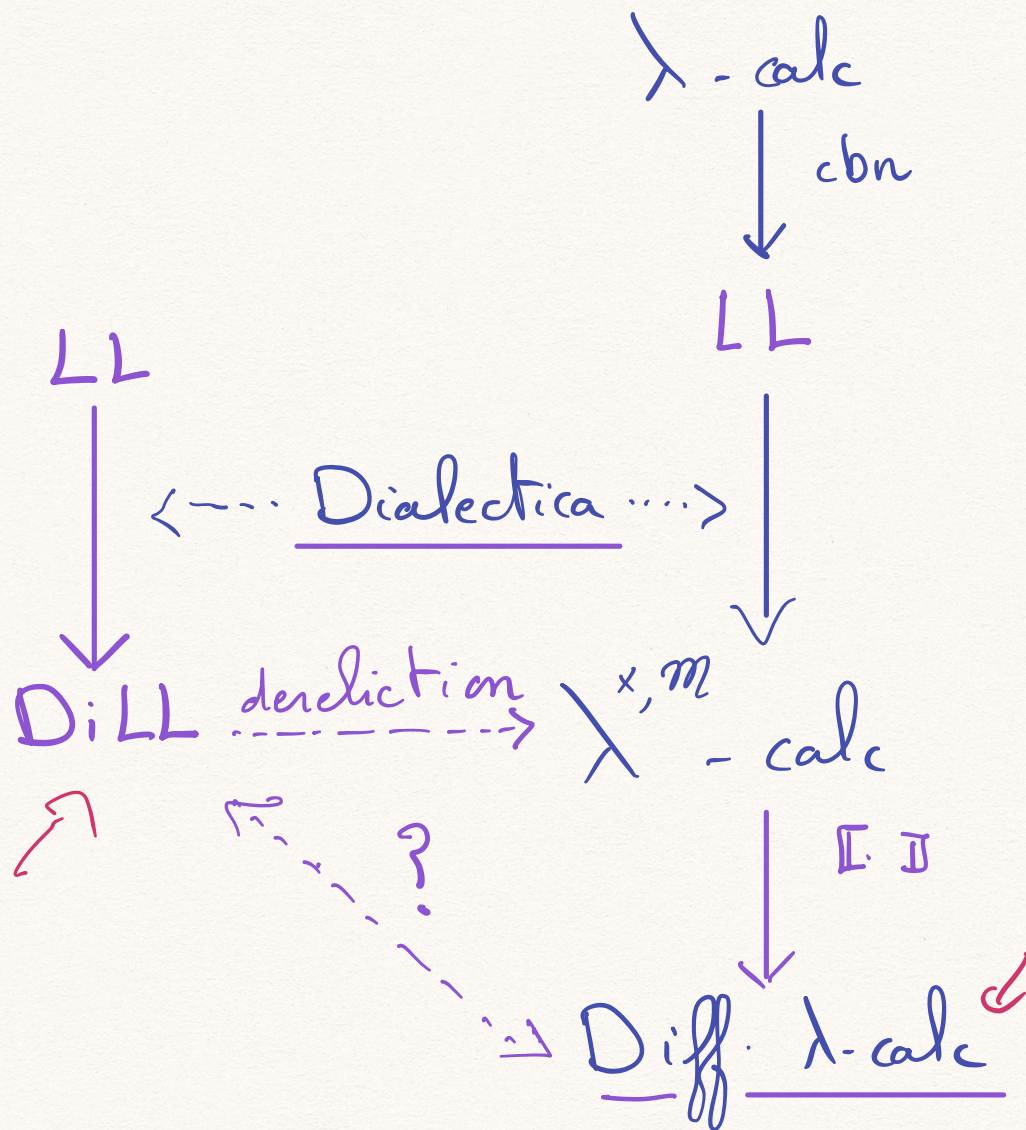
e.g. $\mathbb{W}(!A \multimap B) = !\mathbb{W}(A) \multimap \mathbb{W}(B)$
 $\& (!\mathbb{W}(A) \multimap \mathbb{C}(B) \multimap \mathbb{C}(A))$

Thm

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

Applications

A language typed by DiLL



A language typed by DiLL and expressing automatic differentiation ?

A language typed by DiLL

- CBN + Dialectica \Rightarrow reverse diff
- CBV + Dialectica \Rightarrow ?

- Differentiation can be made efficient by distinguishing linearity :

$$D(l \circ f) = D(l) \circ D(f) \quad \checkmark$$

- Let's give a computational content to exponential rules.

$$\begin{aligned} u, v &:= x \mid t^\perp \mid u * v \mid \emptyset \mid u \otimes v \mid 1 \mid \delta_u \mid D_u(t) \mid \downarrow t \\ t, s &:= u^\perp \mid t \cdot s \mid w_1 : N \mid \lambda x. t \mid dx. t \mid \uparrow u \end{aligned}$$

$$\begin{aligned} (\lambda x. t) \delta_u &\rightarrow t[u/x] \\ (\lambda x. t) D_w u &\rightarrow \dots \end{aligned}$$

Proof mining

extracting quantitative info 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

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

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.

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

From a proof

$$\forall \varepsilon, \exists \eta, |G_u(a, b)| < \eta \Rightarrow |a - b| < \varepsilon$$

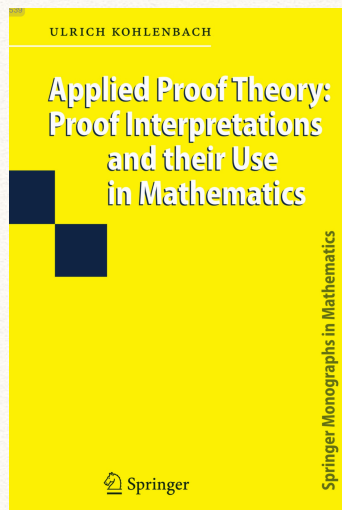
one gets

$$\forall u \exists \Phi \forall k \quad |G_u(a, b)| < 2^{-\Phi_{uk}}$$

$$\Rightarrow |a - b| \leq 2^{-k}$$

Proof mining

- Proof-theory gives you the existence of a quantitative version of a proof.
- One cuts a proof in reasonably sized lemmas.
- Extract quantitative info from each lemma, and compose them.
- Best effective module of uniqueness are linear



Input:

- differentiating ε
- What is the computational version of proof-mining?

Related work :

- Everything on differential λ -calculus and Differential Linear Logic.

and :

A Functional Functional Interpretation

Pierre-Marie Pédro

LICS 14

Laboratoire PPS, CNRS, UMR 7126, Univ Paris Diderot,
Sorbonne Paris Cité, PiR2, INRIA Paris Rocquencourt, F-75205
Paris, France
pierre-marie.pedrot@inria.fr

Backpropagation in the Simply Typed Lambda-calculus with Linear Negation

ALOÏS BRUNEL, Deepomatic, France

POPL '20

DAMIANO MAZZA, CNRS, UMR 7030, LIPN, Université Paris 13, Sorbonne Paris Cité, France

MICHELE PAGANI, IRIF UMR 8243, Université de Paris, CNRS, France

Gödel's functional interpretation and the concept of learning

Thomas Powell

LICS '18

University of Innsbruck
thomas.powell@uibk.ac.at

Conclusion:

- Dialectica categories are a generalization of differential categories
- The computational content of Dialectica + cbn is higher-order reverse mode differentiation
- Dialectica refines to a transformation from LL to DiLL

Question:

Why does differentiation realizes IPP, and Markov's principle when reverse ?

