Derivatives of proofs in linear logic

Daniel Murfet
based on joint work with James Clift

Precis

BHK interpretation: intuitionistic proofs of $A \to B$
give rise to functions $\mathrm{Proofs}(A) \to \mathrm{Proofs}(B)$

- Can these functions be differentiated?
- What would such derivatives be good for?
  1. Efficient (re)computation
  2. Differentiable reasoning
  3. Investigating logic vs physics

According to the Brouwer-Heyting-Kolmogorov interpretation of intuitionistic logic, a proof of A -> B is a construction transforming any proof of A into a proof of B. This construction describes a function from the set of proofs of A to the set of proofs of B. The subject of this talk is the following question: can such functions be differentiated?

At first glance the answer is obviously No, or even worse, that the question itself is ill-posed. To differentiate something means to vary the input slightly and measure the resulting variation in the output. But the input to such a function is a proof of A, and proofs are discrete syntactic things. There is no reasonable sense in which we can vary a proof infinitesimally and have the result still be a proof in the usual sense. Of course, we could change the meaning of the word “proof” and admit, say, all the usual proofs of intuitionistic logic plus some formal things which stand for infinitesimal variations of the “real” proofs. Roughly speaking, Ehrhard and Regnier showed that one can proceed in this manner to enlarge the universe of intuitionistic proofs in such a way that the resulting class of has a natural notion of derivatives (at least for first-order logic).

Then the obvious question is: what good is it to differentiate a proof? I’m not entirely clear on the original motivations of Ehrhard and Regnier, but here are what I consider the three best reasons to care about proof derivatives:

(a) under the Curry-Howard correspondence this proof is a program taking inputs of type A and returning outputs of type B. Suppose the output is calculated for some particular input x, and that it takes a week to complete the calculation. And then, suppose we want to know the output for an input y which is “close to” x (say that y is a 100 digit binary word, differing in one digit from x). If we had computed the derivatives of the program ahead of time, we could imagine using a Taylor expansion to compute the output of the program on y, from its output on x together with its precomputed derivatives. Obviously finding the derivatives imposes some overhead cost, but the “Taylor method” will still be efficient for some use-cases. One form of this, for programs computing real-valued functions, is known as “incremental computation” and is widely used in applied computer science.

Precis

BHK interpretation: intuitionistic proofs of $A \to B$
give rise to functions $\mathrm{Proofs}(A) \to \mathrm{Proofs}(B)$

- Can these functions be differentiated?
- What would such derivatives be good for?

1. Efficient (re)computation
2. Differentiable reasoning
3. Investigating logic vs physics

(a) Suppose there is a phenomena in the world that I'm trying to abstract in logic. For concreteness, say I'm trying to represent a pattern I observe in nature by a sequence of integers. Perhaps at first my idea is in error, in the sense that the pattern I have in my head is not consistent with all the evidence. In what direction should I vary my "idea" in order to bring it into better agreement with the world? This kind of smooth variation of "ideas" is behind the connectionist or neural network approach to machine learning, and forms one widely subscribed model of human reasoning. It would be good to have a more principled way of thinking about this.

(b) There are many interesting speculations out there about connections between information theory, logic and physics. It's hard to know how seriously to take some of them. The most well-understood lie in thermodynamics, as I've discussed in this seminar previously. One very naive observation is the following: physical instantiations of computational processes are generally described by continuous physics at some level, and so the relevant processes admit derivatives. To what extent are these derivatives just "operational junk" which bears no deep significance, and to what extent are they the shadow of something meaningful in logic that we happen to have missed? We can't even ask this question until we have grappled with the notion of derivatives of proofs.

Let me now recontextualise these issues using the backdrop of the Curry-Howard correspondence.

Curry-Howard correspondence

logic	programming	math
formula	type	space
proof	program	function
cut-elimination	execution	—
contraction	copying	coalgebra
?	**?**	calculus

The first two columns in this table are the standard Curry-Howard correspondence. Formulas in logic correspond to types in programming, and so on. What I mean by the third column is the following: there are many semantics of logic in which one assigns spaces to formulas, and functions between spaces to proofs.

On the previous slide I was alluding to the possibility of expanding the notion of syntactic proofs so as to have a proof-theoretic notion of derivative. That's Step 1. But given that we have some semantics of logic in which the denotation of proofs are functions between spaces, Step 2 must be to make sure that these syntactic proof derivatives are consistent with the usual calculus derivatives of the denotations. The mechanism of conciliation between syntactic and semantic derivatives is coalgebras, as I will explain.

Outline

1. History of derivatives in logic
2. Derivatives in the syntax
3. Relation to calculus via coalgebras

(based on arXiv:1701.01285)

History of derivatives in logic

- Leibniz's stepped reckoner (1670s)
- Babbage's difference engine (1830s)
- Circuits and 2nd order differential equations
- Automatic differentiation of real-valued programs
- Ehrhard-Regnier's differential lambda calculus (2003)
- Differential linear logic

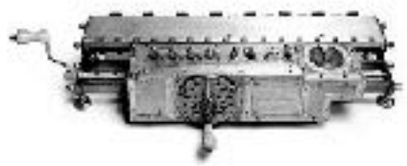
I'd like to give a very brief look at the history of logic and computing devices, highlighting some of the strands that resonate with our theme of derivatives. Perhaps the very beginning of such a history is Leibniz and his fascination with formal languages for reasoning and with devices for carrying out calculations in them. The only device of this kind that he actually built, as far as I'm aware, was the stepped reckoner, which was a mechanical calculator for arithmetical operations. The physical operation of this machine is interesting from the point of view of modern logic and its semantics. This will be the one example in this list that I will examine in detail, so let me not say too much about it now.

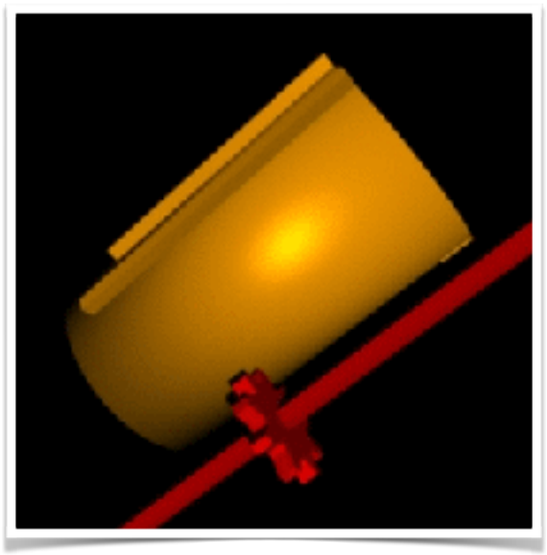
I think most of you will be familiar with Babbage's difference engine, which is usually given as one of the eminent ancestors of today's computers. It was designed to automatically compute values of polynomial functions, for example the truncated Taylor expansion of the logarithm. It did so using Newton's theory of divided differences, which is the way that derivatives are dealt with within abstract algebra.

As students in my Calculus 2 class have just learned, electrical circuits containing only capacitors, inductors and resistors are modelled by second-order ordinary differential equations.

The most well developed link between logic and calculus that I'm aware of is the differential lambda calculus of Ehrhard and Regnier, and the refinement of its simply-typed variant in differential linear logic. I'll go into detail about the latter system later on in this talk.

History: Leibniz's stepped reckoner

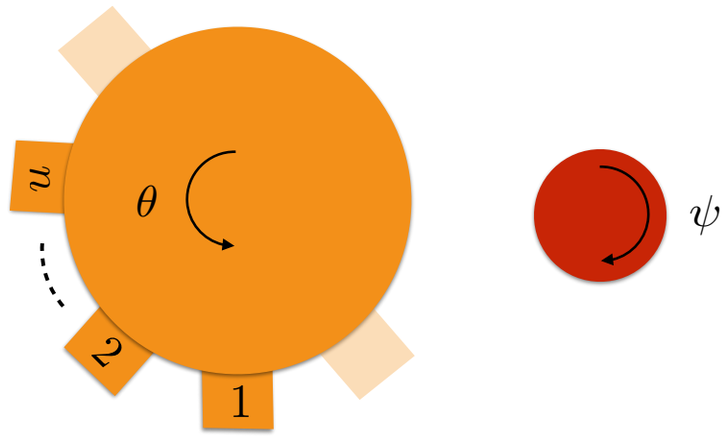




The stepped reckoner was the first true four-function calculator, which formed the basis for many calculator designs for the next two hundred years. Leibniz actually built several of them.

I'm going to explain the key innovation in the mechanical design of this calculator, which is the stepped drum that you see in the animation. I'll then explain how an idealised mathematical model of this mechanism leads us to the denotational semantics of Church numerals.

History: Leibniz's stepped reckoner

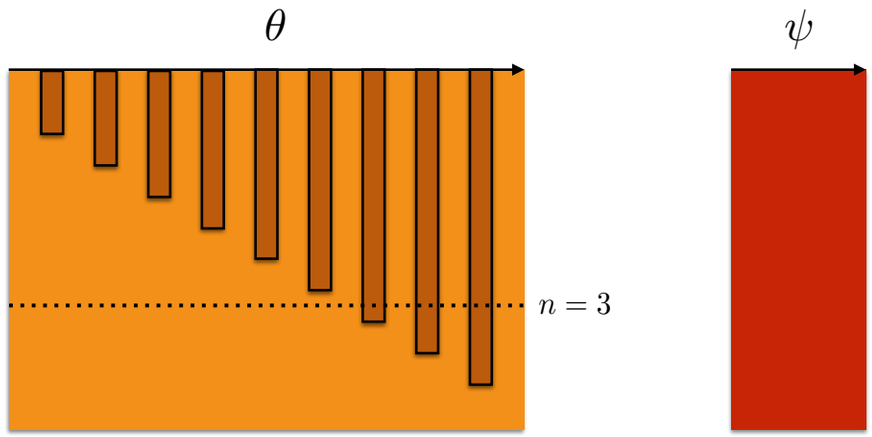


After a full rotation of the **drum**, the **shaft** rotates by nk

$$\Delta\psi = nk\frac{\Delta\theta}{2\pi} \qquad \theta = 0, 2\pi, 4\pi, \ldots$$

Here is a top down view of the orange stepped drum, and the red gear and shaft. There are gears on the red shaft which I am not drawing here, obviously. Let theta and psi denote respectively the angle of rotation of the orange and red components. We assume that the integer n is fixed, that is, that we have selected the position of the red gears along the long axis of the stepped drum, which is the axis running out of the page, such that after a single rotation of the orange drum the red gears have encountered exactly n of the orange gear teeth. The effect is to cause the red shaft to undergo a total rotation of nk \Delta \theta on 2\pi, whenever \Delta \theta is a multiple of 2 \pi. Here k is the quantity of angle induced by a single gear tooth on the orange drum.

History: Leibniz's stepped reckoner

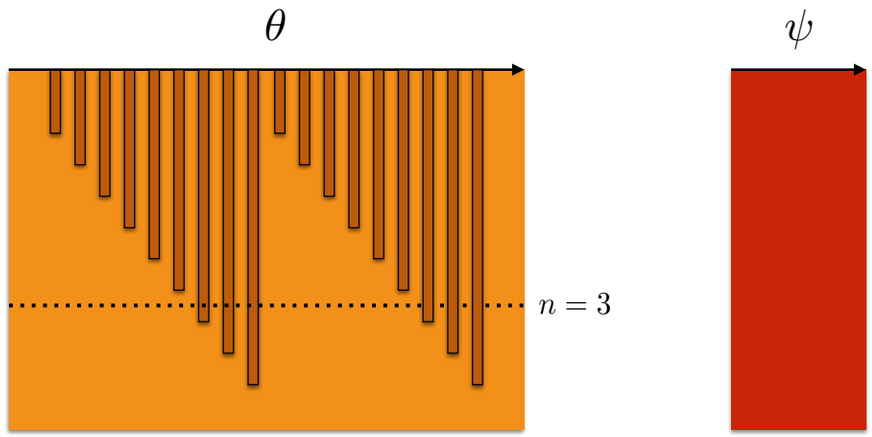


After a full rotation of the **drum**, the **shaft** rotates by nk

$$\Delta\psi = nk\frac{\Delta\theta}{2\pi} \qquad \Delta\theta = 0, 2\pi, 4\pi, \ldots$$

This slide shows another view of the same system. This time the long axis of the orange drum runs vertically on the page, and I'm showing the horizontal axis (the dotted line) along which the red gear runs when n is fixed to 3.

History: Leibniz's stepped reckoner



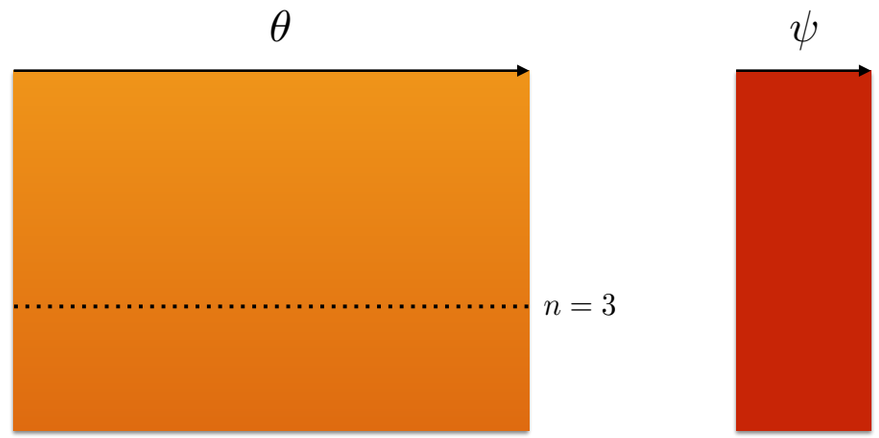
After a full rotation of the **drum**, the **shaft** rotates by nk

(if we halve the rotation caused by each tooth, while doubling the number)

$$\Delta\psi = nk\frac{\Delta\theta}{2\pi} \qquad \Delta\theta = 0, \pi, 2\pi, \ldots$$

At the moment this is not a very continuous model, because an infinitesimal change in the angle \theta does not induce an infinitesimal change in the angle \psi. To refine the model into a continuous one, we can double the number of vertical teeth on the orange drum, while halving the angle of rotation each individual tooth induces on the red shaft. In that case, a full rotation of \Delta \theta = 2\pi will still cause a rotation of nk on the red shaft. But a half rotation of \Delta \theta = \pi will induce a rotation of nk/2. That is, we have arranged by this physical change to make the formula relating \Delta \psi to \Delta \theta true for values of \Delta \theta which are multiples of \pi, not just multiples of 2\pi.

History: Leibniz's stepped reckoner



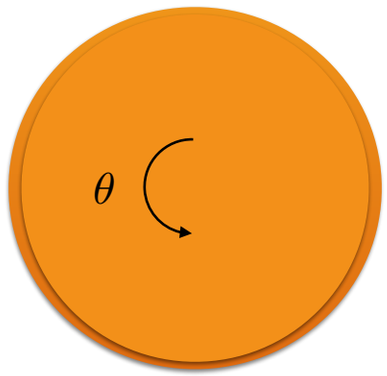
In the limit of infinitely many repetitions of this group of nine teeth

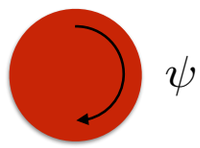
$$d\psi = nk\frac{d\theta}{2\pi} = nk'd\theta \qquad k' = \frac{k}{2\pi} \qquad \boxed{\frac{d\psi}{d\theta} = nk'}$$

Obviously we can continue this, adding more teeth and reducing the induced angle, so as to make the formula true for, in the limit, arbitrary values of \Delta \theta. In the limit, we get an idealised form of the stepped reckoner where d \psi is equal to n/k d\theta on 2\pi. We may rewrite this as d \psi equals nk' d\theta, where k' is k/2\pi. That is, the rate of change of \psi with respect to \theta, is nk'.

One way to physically implement this is to have an orange drum of variable radius.

History: Leibniz's stepped reckoner





$$\frac{d\psi}{d\theta} = n$$

$$\psi = n\theta$$

$$e^{i\psi} = e^{in\theta} = (e^{i\theta})^n$$

Upshot: The stepped reckoner gives a "physical semantics" of the Church numerals matching the denotational semantics in vector spaces

$$\begin{array}{ccc} U(1) & \xrightarrow{(-)^n} & U(1) \\ \downarrow & & \downarrow \\ SO(2) & \xrightarrow{(-)^n} & SO(2) \\ \downarrow & & \downarrow \\ M_2(\mathbb{R}) & \xrightarrow{(-)^n = [\![\underline{n}]\!]} & M_2(\mathbb{R}) \end{array}$$

I'm not sure how seriously to take this example. It might be, for instance, that many other mechanical aspects of the stepped reckoner, as well as more complex systems like the difference engine, are just physical instantiations of algorithms involving Church numerals. It would be interesting to see if, for example, there is a relationship between the way arithmetic operations on Church numerals are encoded into simply-typed lambda calculus, or linear logic, and the physical mechanisms which implement arithmetic in these mechanical devices. If that is the case, this example should be taken seriously.

But in the context of this talk, I want it to serve only to illustrate the following point: algorithms in logic are discrete, syntactic objects, but they are represented in the world as manipulations of _continuous, even differentiable_ systems. In the case we've just discussed the algorithm is the numeral n, and we discussed two of its representations: the physical one in the stepped reckoner, and mathematical one in the denotational semantics of linear logic in vector spaces. The continuous quantities in the former case are angles, and in the latter case they are linear operators, and these are related by the fact that rotation by a given angle is a linear operator on vectors in the plane.

If you buy that, then the claim that algorithms have meaningful derivatives doesn't seem as outrageous. It is simply the claim that the derivatives of the _representation_ of the algorithm (as a manipulation of continuous objects) are so completely determined by the algorithm itself that they are themselves representations of something that properly belongs at the level of logic.

With that in mind, we now move on to describe what these "somethings" are, in the logic.

Derivatives in the syntax

- Differential linear logic adds a new deduction rule, which produces the derivative of a proof in a direction specified by a new (linear) hypothesis.

$$\dfrac{\begin{array}{c}\pi\\ \vdots \\ !A \vdash B\end{array}}{!A, A \vdash B}\ \text{diff} \qquad (a_1, a_2) \longrightarrow \lim_{h \to 0} \frac{\pi(a_1 + ha_2) - \pi(a_1)}{h}$$

(this is meaningless)

- In the best formulation **diff** is derived from *codereliction, cocontraction* and *coweakening*.

The idea of extending linear logic with differential operators goes back right to the genesis of linear logic, in Girard's exploration of the quantitative semantics of the lambda calculus, and his paper on normal functors, power series and the lambda calculus in 1988. These ideas were further developed in semantics papers by Ehrhard in the early 2000s, and reached a kind of fruition in a paper by Ehrhard and Regnier in which they developed a variant of the untyped lambda calculus with derivatives, in 2003. This was followed by the development of differential linear logic by the same authors, with important work on the categorical semantics being done by Blute, Cockett and Seely.

Deduction rules for (intuitionistic, first-order) linear logic

$$\text{(Dereliction):}\quad \frac{\Gamma, A, \Delta \vdash B}{\Gamma, !A, \Delta \vdash B}\,\text{der}$$

$$\text{(Contraction):}\quad \frac{\Gamma, !A, !A, \Delta \vdash B}{\Gamma, !A, \Delta \vdash B}\,\text{ctr}$$

$$\text{(Weakening):}\quad \frac{\Gamma, \Delta \vdash B}{\Gamma, !A, \Delta \vdash B}\,\text{weak}$$

$$\text{(Axiom):}\quad \frac{}{A \vdash A} \qquad \text{(Cut):}\quad \frac{\Gamma \vdash A \quad \Delta', A, \Delta \vdash B}{\Delta', \Gamma, \Delta \vdash B}\,\text{cut} \qquad \text{(Promotion):}\quad \frac{!\Gamma \vdash A}{!\Gamma \vdash !A}\,\text{prom}$$

$$\text{(Left } \multimap\text{):}\quad \frac{\Gamma \vdash A \quad \Delta', B, \Delta \vdash C}{\Delta', \Gamma, A \multimap B, \Delta \vdash C} \multimap L \qquad \text{(Left } \otimes\text{):}\quad \frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, A \otimes B, \Delta \vdash C}\,\otimes\text{-}L$$

$$\text{(Right } \multimap\text{):}\quad \frac{A, \Gamma \vdash B}{\Gamma \vdash A \multimap B} \multimap R \qquad \text{(Right } \otimes\text{):}\quad \frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}\,\otimes\text{-}R$$

Deduction rules for (intuitionistic, first-order) linear logic

(Dereliction): $\dfrac{\Gamma, A, \Delta \vdash B}{\Gamma, !A, \Delta \vdash B}\,\text{der}$

(Contraction): $\dfrac{\Gamma, !A, !A, \Delta \vdash B}{\Gamma, !A, \Delta \vdash B}\,\text{ctr}$

(Weakening): $\dfrac{\Gamma, \Delta \vdash B}{\Gamma, !A, \Delta \vdash B}\,\text{weak}$

$\underline{001} : \mathbf{bint}_A = !(A \multimap A) \multimap (!(A \multimap A) \multimap (A \multimap A))$

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \vdash A} \quad \dfrac{\dfrac{}{A \vdash A} \quad \dfrac{}{A \vdash A}}{A, A \multimap A \vdash A}\multimap L}{A, A \multimap A, A \multimap A \vdash A}\multimap L}{A, A \multimap A, A \multimap A, A \multimap A \vdash A}\multimap L}{A \multimap A, A \multimap A, A \multimap A \vdash A \multimap A}\multimap R}{!(A \multimap A), !(A \multimap A), !(A \multimap A) \vdash A \multimap A}3\times \text{der}}{!(A \multimap A), !(A \multimap A) \vdash A \multimap A}\text{ctr}}{\vdash \mathbf{bint}_A}2\times \multimap R$$

(Axiom): $\dfrac{}{A \vdash A}$ (Cut): $\dfrac{\Gamma \vdash A \quad \Delta', A, \Delta \vdash B}{\Delta', \Gamma, \Delta \vdash B}\,\text{cut}$ (Promotion): $\dfrac{!\Gamma \vdash A}{!\Gamma \vdash !A}\,\text{prom}$

(Left $\multimap$): $\dfrac{\Gamma \vdash A \quad \Delta', B, \Delta \vdash C}{\Delta', \Gamma, A \multimap B, \Delta \vdash C}\multimap L$ (Left $\otimes$): $\dfrac{\Gamma, A, B, \Delta \vdash C}{\Gamma, A \otimes B, \Delta \vdash C}\otimes\text{-}L$

(Right $\multimap$): $\dfrac{A, \Gamma \vdash B}{\Gamma \vdash A \multimap B}\multimap R$ (Right $\otimes$): $\dfrac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B}\otimes\text{-}R$

Deduction rules for differential linear logic

(Dereliction): $\dfrac{\Gamma, A, \Delta \vdash B}{\Gamma, !A, \Delta \vdash B}\,\mathrm{der}$ (Codereliction): $\dfrac{\Gamma, !A, \Delta \vdash B}{\Gamma, A, \Delta \vdash B}\,\mathrm{coder}$

(Contraction): $\dfrac{\Gamma, !A, !A, \Delta \vdash B}{\Gamma, !A, \Delta \vdash B}\,\mathrm{ctr}$ (Cocontraction): $\dfrac{\Gamma, !A, \Delta \vdash B}{\Gamma, !A, !A, \Delta \vdash B}\,\mathrm{coctr}$

(Weakening): $\dfrac{\Gamma, \Delta \vdash B}{\Gamma, !A, \Delta \vdash B}\,\mathrm{weak}$ (Coweakening): $\dfrac{\Gamma, !A, \Delta \vdash B}{\Gamma, \Delta \vdash B}\,\mathrm{coweak}$

(Axiom): $\dfrac{}{A \vdash A}$ (Cut): $\dfrac{\Gamma \vdash A \quad \Delta', A, \Delta \vdash B}{\Delta', \Gamma, \Delta \vdash B}\,\mathrm{cut}$ (Promotion): $\dfrac{!\Gamma \vdash A}{!\Gamma \vdash !A}\,\mathrm{prom}$

(Left $\multimap$): $\dfrac{\Gamma \vdash A \quad \Delta', B, \Delta \vdash C}{\Delta', \Gamma, A \multimap B, \Delta \vdash C} \multimap L$ (Left $\otimes$): $\dfrac{\Gamma, A, B, \Delta \vdash C}{\Gamma, A \otimes B, \Delta \vdash C} \otimes\text{-}L$

(Right $\multimap$): $\dfrac{A, \Gamma \vdash B}{\Gamma \vdash A \multimap B} \multimap R$ (Right $\otimes$): $\dfrac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma, \Delta \vdash A \otimes B} \otimes\text{-}R$

Deduction rules for differential linear logic

(Dereliction): $\dfrac{\Gamma, A, \Delta \vdash B}{\Gamma, !A, \Delta \vdash B}\,\mathrm{der}$ (Codereliction): $\dfrac{\Gamma, !A, \Delta \vdash B}{\Gamma, A, \Delta \vdash B}\,\mathrm{coder}$

(Contraction): $\dfrac{\Gamma, !A, !A, \Delta \vdash B}{\Gamma, !A, \Delta \vdash B}\,\mathrm{ctr}$ (Cocontraction): $\dfrac{\Gamma, !A, \Delta \vdash B}{\Gamma, !A, !A, \Delta \vdash B}\,\mathrm{coctr}$

(Weakening): $\dfrac{\Gamma, \Delta \vdash B}{\Gamma, !A, \Delta \vdash B}\,\mathrm{weak}$ (Coweakening): $\dfrac{\Gamma, !A, \Delta \vdash B}{\Gamma, \Delta \vdash B}\,\mathrm{coweak}$

$$\dfrac{\begin{array}{c}\pi\\ \vdots\\ !A \vdash B\end{array}}{!A, A \vdash B}\,\mathrm{diff} \qquad \text{is defined to be} \qquad \dfrac{\dfrac{\begin{array}{c}\pi\\ \vdots\\ !A \vdash B\end{array}}{!A, !A \vdash B}\,\mathrm{coctr}}{!A, A \vdash B}\,\mathrm{coder}$$

Product rule as cut-elimination rule

$$\dfrac{D_A \qquad \dfrac{}{!A \vdash !A}\,ax}{!A, A \vdash !A}\,diff$$

$$\dfrac{D_A \qquad \pi \qquad\qquad \dfrac{!A, !A \vdash B}{!A \vdash B}\,ctr}{!A, A \vdash B}\,cut \quad \rightsquigarrow \quad \dfrac{D_A \qquad \pi \qquad\qquad !A, !A \vdash B}{\dfrac{!A, !A, A \vdash B}{!A, A \vdash B}\,ctr}\,cut \;+\; \dfrac{D_A \qquad \pi \qquad\qquad !A, !A \vdash B}{\dfrac{!A, !A, A \vdash B}{!A, A \vdash B}\,ctr}\,cut$$

Proofs in differential linear logic are formal linear sums of proof trees

$$\mathbf{bint}_A = !(A \multimap A) \multimap (!(A \multimap A) \multimap (A \multimap A)).$$

$$E = A \multimap A$$

$$\underline{\text{repeat}} : !\mathbf{bint}_A \multimap \mathbf{bint}_A$$

$$
\begin{prooftree}
\AxiomC{}
\UnaryInfC{$!E \vdash !E$}
\AxiomC{}
\UnaryInfC{$!E \vdash !E$}
\AxiomC{}
\UnaryInfC{$!E \vdash !E$}
\AxiomC{}
\UnaryInfC{$!E \vdash !E$}
\AxiomC{$\underline{\text{comp}}^2_A$}
\noLine
\UnaryInfC{$\vdots$}
\noLine
\UnaryInfC{$E, E \vdash E$}
\RightLabel{$\multimap L$}
\BinaryInfC{$!E, !E \multimap E, E \vdash E$}
\RightLabel{$\multimap L$}
\BinaryInfC{$!E, !E, \mathbf{bint}_A, E \vdash E$}
\RightLabel{$\multimap L$}
\BinaryInfC{$!E, !E, !E, \mathbf{bint}_A, !E \multimap E \vdash E$}
\RightLabel{$\multimap L$}
\BinaryInfC{$!E, !E, !E, !E, \mathbf{bint}_A, \mathbf{bint}_A \vdash E$}
\RightLabel{ctr}
\UnaryInfC{$!E, !E, !E, \mathbf{bint}_A, \mathbf{bint}_A \vdash E$}
\RightLabel{ctr}
\UnaryInfC{$!E, !E, \mathbf{bint}_A, \mathbf{bint}_A \vdash E$}
\doubleLine
\RightLabel{$2\times \multimap R$}
\UnaryInfC{$\mathbf{bint}_A, \mathbf{bint}_A \vdash \mathbf{bint}_A$}
\doubleLine
\RightLabel{$2\times$ der}
\UnaryInfC{$!\mathbf{bint}_A, !\mathbf{bint}_A \vdash \mathbf{bint}_A$}
\RightLabel{ctr}
\UnaryInfC{$!\mathbf{bint}_A \vdash \mathbf{bint}_A$}
\end{prooftree}
$$

$$\frac{\underline{\text{repeat}}}{\begin{array}{c}\vdots\\ !\mathbf{bint}_A \vdash \mathbf{bint}_A\end{array}}{\scriptstyle\text{diff}}$$
$$!\mathbf{bint}_A, \mathbf{bint}_A \vdash \mathbf{bint}_A$$

$$(\underline{S}, \underline{T}) \longmapsto \underline{ST} + \underline{TS}$$

$$(S + \varepsilon T)(S + \varepsilon T) = SS + \varepsilon(ST + TS) + \varepsilon^2 TT$$

Relation to calculus via coalgebras

- Following Ehrhard-Regnier we have defined derivatives in the syntax, via new deduction rules and cut-elimination rules.
- Do these syntactic derivatives capture the logical content lying behind the *semantic* derivatives?
- In particular, are they consistent with the role of Church numerals in Leibniz's stepped reckoner?
- **Yes**: because coalgebras

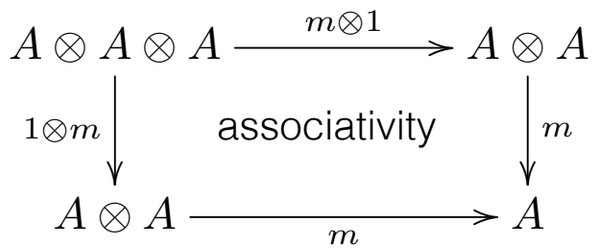
So far I have sketched how to add new deduction rules and cut-elimination rules to intuitionistic first-order linear logic, in such a way that you can differentiate proofs. And I've shown that in at least one example the derivative you get has some reasonable interpretation.

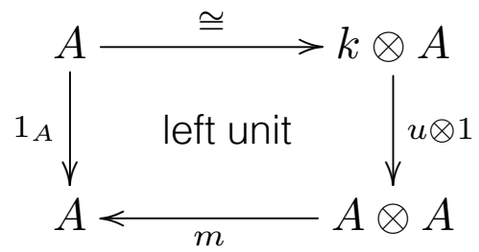
But the question remains: do these syntactic derivatives capture the logical content lying behind the derivatives of the representations of these algorithms? For example, does the derivative of the Church numeral according to these rules match up with the derivatives of the map which sends a matrix to its nth power? If this were the case, at least in this example, the syntactic derivatives really would capture what is going on with the calculus involved in the stepped reckoner.

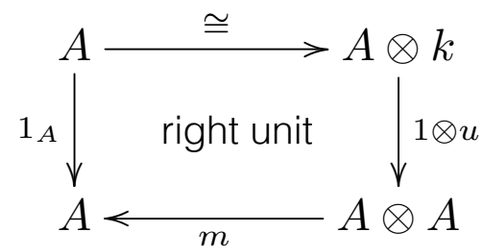
The answer to both this particular case, and the broader questions, is Yes. To cut a long story short, the reason is that the nonlinearities in linear logic are built on the ! connective, which gets interpreted as a cofree coalgebra. Derivatives of polynomial functions can be computed using morphisms out of the coalgebra which is the dual of the dual numbers. So some standard mathematics about coalgebras and their relation to calculus, plus the fact that coalgebras are built into the foundations of linear logic, reconciles the syntactic and semantic derivatives.

Algebras over a field k

multiplication $m : A \otimes A \longrightarrow A$ $\quad u : k \longrightarrow A$ unit







Before I can explain the role of coalgebras in differential linear logic, I have to give a brief introduction to coalgebras and define the cofree coalgebra.

Coalgebras over a field k

comultiplication $\Delta : A \longrightarrow A \otimes A$ $\qquad$ $c : A \longrightarrow k$ counit

$$\begin{array}{ccc} A \otimes A \otimes A & \xleftarrow{\Delta \otimes 1} & A \otimes A \\ {\scriptstyle 1 \otimes \Delta}\uparrow & \text{coassociativity} & \uparrow{\scriptstyle \Delta} \\ A \otimes A & \xleftarrow[\Delta]{} & A \end{array}$$

$$\begin{array}{ccc} A & \xleftarrow{\cong} & k \otimes A \\ {\scriptstyle 1_A}\uparrow & \text{left counit} & \uparrow{\scriptstyle c \otimes 1} \\ A & \xrightarrow[\Delta]{} & A \otimes A \end{array} \qquad \begin{array}{ccc} A & \xleftarrow{\cong} & A \otimes k \\ {\scriptstyle 1_A}\uparrow & \text{right counit} & \uparrow{\scriptstyle 1 \otimes c} \\ A & \xrightarrow[\Delta]{} & A \otimes A \end{array}$$

Examples

polynomial algebra

$$k[x_1, \ldots, x_n]$$

ring of dual numbers

$$k[\varepsilon]/(\varepsilon^2) = k \cdot 1 \oplus k \cdot \varepsilon$$

$$\varepsilon^2 = 0$$

polynomial coalgebra

$$k[x_1, \ldots, x_n]$$

$$\Delta(x^n) = \sum_{i=0}^{n} x^i \otimes x^{n-i}$$

dual of the ring of dual numbers

$$(k[\varepsilon]/(\varepsilon^2))^* = k \cdot 1^* \oplus k \cdot \varepsilon^*$$

$$\Delta(1) = 1 \otimes 1$$

$$\Delta(\varepsilon^*) = 1 \otimes \varepsilon^* + \varepsilon^* \otimes 1$$

For us the two most important examples of coalgebras are the cofree coalgebra on a vector space, which I'll introduce in a moment, and the coalgebra dual to the ring of dual numbers. The ring of dual numbers encodes algebraically the idea of an "infinitesimal" and it is standard in algebraic geometry to think of tangent vectors at a point of a space in terms of functions to the space from the space with "one point and an infinitesimal tangent vector", which is the spectrum of the ring of dual numbers.

Without getting too far into that, I'd like to sketch why algebra morphisms into the ring of dual numbers are synonymous with tangent vectors, since this will be crucial to explaining how to think about derivatives using just coalgebras and morphisms of coalgebras.

Consider a morphism of k-algebras

$$k[x_1, \ldots, x_n] \xrightarrow{\varphi} k[\varepsilon]/(\varepsilon^2) = k \cdot 1 \oplus k \cdot \varepsilon$$

$$\varphi(x_i) = \lambda_i + \mu_i \varepsilon$$

it is straightforward to see that, for any polynomial f,

$$\varphi(f) = f(\lambda_1, \ldots, \lambda_n) + \sum_i \mu_i \frac{\partial f}{\partial x_i}\Big|_{x=\vec{\lambda}} \cdot \varepsilon$$

this gives rise to a bijection of k-algebra morphisms with pairs

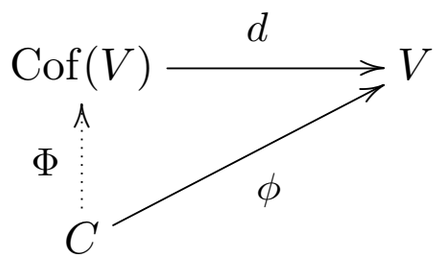
$$\mathrm{Hom}_{k-\mathrm{Alg}}(k[x_1, \ldots, x_n], k[\varepsilon]/(\varepsilon^2)) \xleftrightarrow{1:1} k^n \times k^n$$

$$\varphi \longleftrightarrow (\vec{\lambda}, \vec{\mu}) \quad \textbf{(point, tangent vector)}$$

Universal coalgebra

The *cofree coalgebra* $\mathrm{Cof}(V)$ over a vector space V is a coalgebra together with a linear map $d : \mathrm{Cof}(V) \to V$ which is universal, in the sense that for any coalgebra C and linear $\phi : C \to V$ there unique morphism of coalgebras Φ such that

$$d \circ \Phi = \phi$$



Theorem: $\mathrm{Cof}(V)$ is the space of distributions with finite support on V, i.e. all derivatives of Dirac distributions

Sweedler semantics $[\![-]\!] : \mathrm{LL} \longrightarrow \mathrm{Vect}$

$$[\![A \multimap B]\!] = \mathrm{Hom}_k([\![A]\!], [\![B]\!])$$

$$[\![A \otimes B]\!] = [\![A]\!] \otimes [\![B]\!]$$

$$[\![!A]\!] = \mathrm{Cof}([\![A]\!])$$

dereliction = universal linear map $[\![!A]\!] \longrightarrow [\![A]\!]$

contraction = comultiplication $[\![!A]\!] \longrightarrow [\![!A]\!] \otimes [\![!A]\!]$

weakening = counit $[\![!A]\!] \longrightarrow k$

promotion = lifting of $[\![!A]\!] \longrightarrow [\![B]\!]$ to $[\![!A]\!] \longrightarrow [\![!B]\!]$

Sweedler semantics $[\![-]\!] : \mathrm{LL} \longrightarrow \mathrm{Vect}$

$$[\![A \multimap B]\!] = \mathrm{Hom}_k([\![A]\!], [\![B]\!])$$

$$[\![A \otimes B]\!] = [\![A]\!] \otimes [\![B]\!]$$

$$[\![!A]\!] = \mathrm{Cof}([\![A]\!])$$

The Sweedler semantics is also a semantics of differential linear logic, as follows:

$$\frac{\begin{array}{c}\pi \\ \vdots \\ !A \vdash B\end{array}}{!A, A \vdash B}\ \text{diff}$$

$$\begin{array}{ccccc} [\![!A]\!] \otimes [\![A]\!] & \longrightarrow & [\![!A]\!] & \xrightarrow{[\![\pi]\!]} & [\![B]\!] \\ || & & || & & \\ \mathrm{Cof}([\![A]\!]) \otimes [\![A]\!] & \longrightarrow & \mathrm{Cof}([\![A]\!]) & & \end{array}$$

$$D \otimes \nu \longmapsto \partial_\nu D$$

The important intuition here is that in the Sweedler semantics, to feed an input x to an algorithm \pi, what you actually do is feed the Dirac distribution at the point x into the denotation of \pi. To differentiate \pi in some direction at x you feed into [[\pi]] the derivative of the Dirac distribution in that direction.

(point, tangent vector) $V \times V \quad (\lambda, \mu)$ $\qquad V = k^n$

$$\updownarrow \cong$$

$$\mathrm{Hom}_{k-\mathrm{Alg}}(k[x_1, \ldots, x_n], k[\varepsilon]/(\varepsilon^2))$$

$$\updownarrow \cong$$

$$\mathrm{Hom}_k(\mathrm{Sym}(V^*), k[\varepsilon]/(\varepsilon^2))$$

$$\updownarrow \cong$$

$$\mathrm{Hom}_k(V^*, k[\varepsilon]/(\varepsilon^2))$$

$$\updownarrow \cong$$

$$\mathrm{Hom}_k((k[\varepsilon]/(\varepsilon^2))^*, V)$$

$$\updownarrow \cong$$

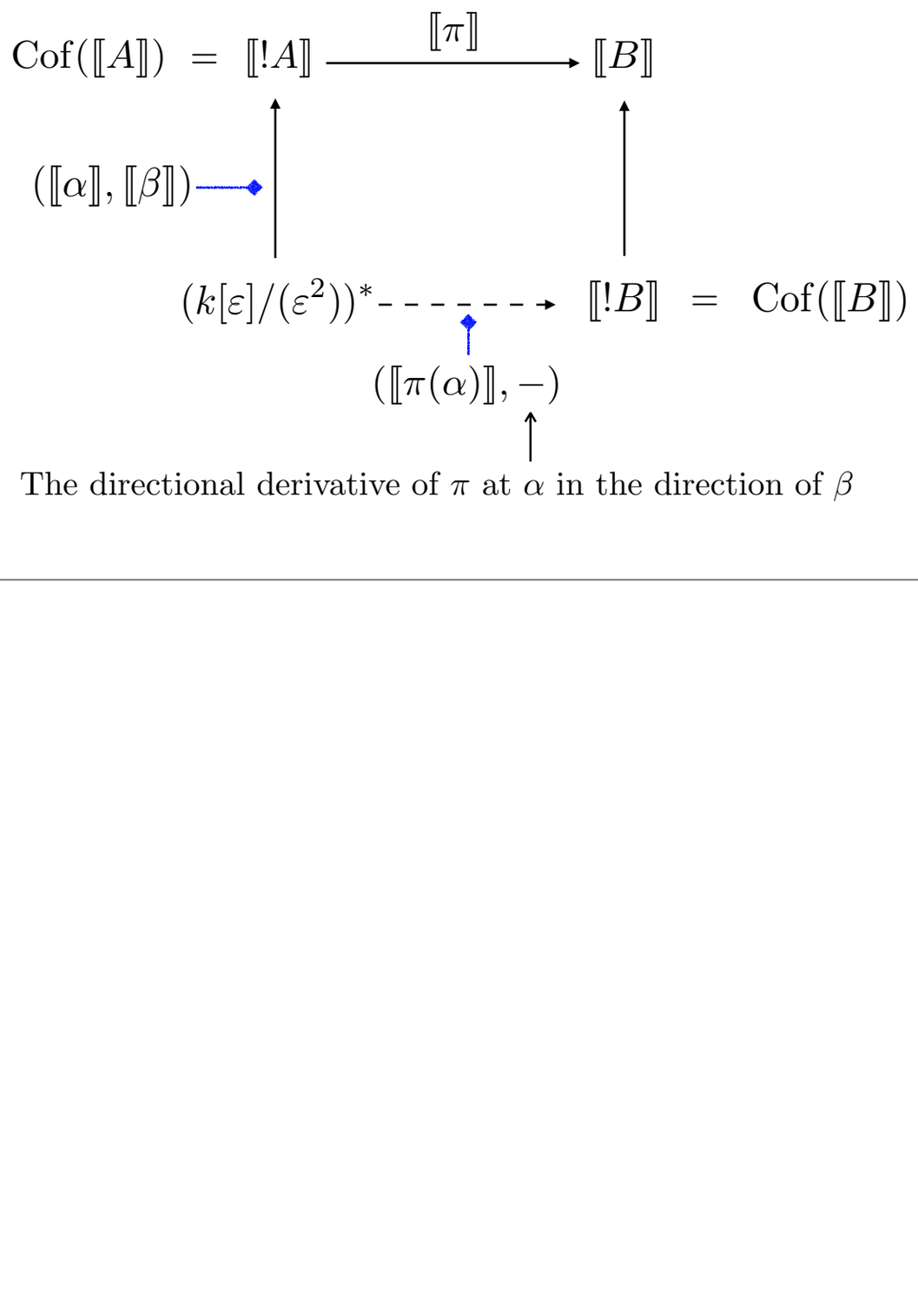
$$\mathrm{Hom}_{k-\mathrm{Coalg}}((k[\varepsilon]/(\varepsilon^2))^*, \mathrm{Cof}(V))$$

$1^* \mapsto \mathrm{Dirac}_\lambda$

$\varepsilon^* \mapsto \partial_\mu \mathrm{Dirac}_\lambda$

How to differentiate a proof denotation

Given $\pi : !A \multimap B$, $\alpha, \beta : A$ so that $[\![\alpha]\!], [\![\beta]\!] \in [\![A]\!]$



Conciliation: syntax vs semantics

- The semantics of (intuitionistic, first-order) linear logic in vector spaces uses cofree coalgebras to model contraction, weakening and dereliction.
- Since the cofree coalgebra is made up of Dirac distributions and their derivatives, this semantics is naturally a model of *differential* linear logic.
- Linear logic secretly wants to be differentiated!

Conclusion/Questions

- Derivatives are natural in (linear) logic.
- Examples like the stepped reckoner suggest the use of calculus in logic is justified. Are there more convincing mechanical examples of this kind?
- The Sweedler semantics is a step in the direction of more interesting algebra and geometry. What is the logical content of distributions with more general support?
- Differential linear logic forms the basis for one approach to integrating symbolic reasoning with neural networks (work in progress with H. Hu).