The differential lambda-calculus: From semantics to syntax

Thomas Ehrhard (IRIF - Université Paris Cité, CNRS, Inria)
Laurent Regnier (I2M - Université d’Aix-Marseille, CNRS)

Tallinn - July 10, 2024
What is differentiation?

Approximate functions by linear maps:

\[
\begin{align*}
  f : \mathbb{R}^n &\rightarrow \mathbb{R} & \sim &\quad Df : \mathbb{R}^n &\rightarrow \mathcal{L}(\mathbb{R}^n, \mathbb{R}) \\
  f(x + u) &= f(x) + Df(x).u + o(u)
\end{align*}
\]

Numerical linearization of \( f \)

But what if \( M : A \Rightarrow B \) is a program?

\[
\begin{align*}
  M : A &\Rightarrow B & \sim &\quad DM : A \Rightarrow (A \rightarrow B)
\end{align*}
\]

Linear logical linearization of \( M \)
The lambda-calculus: a syntax for functions

A syntax to denote functions, just as terms of set theory denote sets: this was the original idea of Alonzo Church’s Type Theory

- Given a set of variables $\mathcal{V} = \{x, y, x_1, \ldots\}$
- if $x \in \mathcal{V}$ then $x$ is a term
- is $M$ and $N$ are terms then $MN$ is a term (function $M$ applied to argument $N$)
- if $x \in \mathcal{V}$ and $M$ is a term then $\lambda x M$ is a term (function $x \mapsto M$). The variable $x$ is bound
Only one rule for computing with these terms: $\beta$-reduction

$$(\lambda x \ M)N \rightarrow \ M[N/x]$$

Computing $= \text{rewriting}$

**Fact**

$\lambda$-calculus is a deterministic and Turing complete model of computation
Do $\lambda$-terms denote functions (or morphisms)?

Yes! It suffices to find a cartesian closed category $C$ and, in that category, an object $U$ together with two morphisms

$$e^+ \in C(U \Rightarrow U, U)$$
$$e^- \in C(U, U \Rightarrow U)$$

such that $e^- \circ e^+ = \text{Id}_{U \Rightarrow U}$

- Impossible if $C$ is the category of sets and functions, for cardinality reasons
- Dana Scott (1968): possible in the category of complete lattices and directed sup preserving functions

Many other examples since then: denotational models of the $\lambda$-calculus
An example

\( \textbf{Rel}_! \) is the following category:

- objects of \( \textbf{Rel}_! \): all sets
- \( \textbf{Rel}_!(E, F) = \mathcal{P}(\mathcal{M}_{\text{fin}}(E) \times F) \) where \( \mathcal{M}_{\text{fin}}(E) \) is the set of all finite multisets \([a_1, \ldots, a_n]\) of elements of \( E \)

with

- identity at \( E \): \( \text{Id}_E = \{([a], a) \mid a \in E\} \)
- composition: if \( R \in \textbf{Rel}_!(E, F) \) and \( S \in \textbf{Rel}_!(F, G) \) then

\[
S \circ R = \{(m_1 + \cdots + m_n, c) \mid \exists b_1, \ldots, b_n \in F \}
\]

\[
(((m_i, b_i) \in R)_{i=1}^n \text{ and } ([b_1, \ldots, b_n], c) \in S) \]
Intuition

Elements of $E = \text{“basis”}$ of a vector space (or rather of a free semi-module on the semi-ring $\{0, 1\}$ with $1 + 1 = 1$); multiset $[a_1, \ldots a_k] = \text{monomial } x_{a_1} \cdots x_{a_k}$

Then $R \in \textbf{Rel}(E, F)$ is a powerseries (with parameters indexed by $E$ and output spanned by $F$)

$$\lambda x^\text{Bool}. x \land \neg x = \{([t, t], f), ([f, f], f), ([t, f], f), ([t, f], t)\}$$

$$= (x_t^2 + x_f^2 + x_t x_f). f + x_t x_f. t$$

The definition of identities and composition is compatible with this intuition

This is a CCC:

- the categorical product is disjoint union
- the internal hom of $E$ and $F$ is $(E \Rightarrow F) = \mathcal{M}_{\text{fin}}(E) \times F$
The simplest model of the $\lambda$-calculus

Given a non-empty set $E_0$ (none elements of which are pairs), we can define a monotone family of sets:

1. $U_0 = \emptyset$
2. $U_{n+1} = E_0 \cup (\mathcal{M}_{\text{fin}}(U_n) \times U_n)$

and then

$$U = \bigcup_{n=0}^{\infty} U_n$$

satisfies

$$U = E_0 \cup (U \Rightarrow U)$$
Linearity in $\textbf{Rel}_!$

Even if the category $\textbf{Rel}_!$ is very simple, it has an interesting feature: some morphisms are **linear**

**Definition**

A morphism $R \in \textbf{Rel}_!(E, F) = \mathcal{P}(\mathcal{M}_{\text{fin}}(E) \times F)$ is linear if all the elements of $R$ are of shape $([a], b)$

Identity morphisms are linear and linear morphisms are stable under composition
the subcategory of sets and linear morphisms is (isomorphic) to \( \textbf{Rel} \), the category of sets and relations

This category is monoidal symmetric, that is there is a well behaved tensor product which is simply \( E \otimes F = E \times F \)

\( \textbf{Rel} \) is a well-known model of Linear Logic
We have actually an internal linear hom in \( \text{Rel} \):

\[
\text{Rel}(G \otimes E, F) = \text{Rel}(G, E \rightarrow F)
\]

namely \( E \rightarrow F = E \times F \)

As a powerseries any \( R \in \text{Rel}! (E, F) = \mathcal{P}(\mathcal{M}_{\text{fin}}(E) \times F) \) has a derivative

\[
R' = \{(m, (a, b)) \mid (m + [a], b) \in R\} \in \text{Rel}! (E, E \rightarrow F)
\]

Satisfies all the expected algebraic properties of a derivative (Leibniz, chain rule etc)
In the semantic universe $\text{Rel}_!$ we have at the same time:

- the $\lambda$-calculus
- and differentiation

This suggests to extend the lambda-calculus with differentiation. This is exactly what we did in 2002.
The differential \( \lambda \)-calculus

We introduce a new construction in the \( \lambda \)-calculus:

- if \( M \) and \( N \) are terms, then \( D M \cdot N \) is a term, the differential application of \( M \) to \( N \)

Intuitively: \( M \) denotes a morphism \( R \in \text{Rel}_! (U, U) \), so we have \( R' \in \text{Rel}_! (U, U \rightarrow U) \), and so, swapping the arguments we get

\[
S \in \text{Rel} (U, U \Rightarrow U)
\]

\( D M \cdot N \) denotes the (linear) application of \( S \) to the denotation of \( N \), which \( \in \mathcal{P} (U) \)

This differential application yields an element of \( \mathcal{P} (U \Rightarrow U) = \text{Rel}_! (U, U) \)
Convenient feature of this syntax: derivatives are easy to iterate

\[ D^k M \cdot (N_1, \ldots, N_k) = D(\cdots DM \cdot N_1 \cdots) \cdot N_k \]
The main idea of the differential λ-calculus is to say that

\[ D(\lambda x \ M) \cdot N \]

is a new redex: the differential β-redex

\[ D(\lambda x \ M) \cdot N \rightarrow \lambda x \left( \frac{\partial M}{\partial x} \cdot N \right) \]

where \( \frac{\partial M}{\partial x} \cdot N \) is a kind of “substitution” defined by induction on \( M \)

Intuition: replace exactly one occurrence of \( x \) in \( M \)
In $\frac{\partial M}{\partial x} \cdot N$, the variable $x$ is still free (there are remaining occurrences of $x$ not yet substituted) $\sim$ the $\lambda x$ remaining in the reduct.

In $y \times x$ one cannot say that $x$ has exactly one occurrence because $y$ can be replaced with a function which duplicates or erases its argument.

We need two more operations on terms to define $\frac{\partial M}{\partial x} \cdot N$:

- a constant 0
- and if $M$ and $N$ are terms, a new term $M + N$

**Good news**

We don’t need anything more
The differential lambda-calculus: From semantics to syntax

\[
\frac{\partial x}{\partial x} \cdot N = N \quad \frac{\partial y}{\partial x} \cdot N = 0 \text{ if } y \in \mathcal{V} \setminus \{x\}
\]

\[
\frac{\partial \lambda y M}{\partial x} \cdot N = \lambda y \left( \frac{\partial M}{\partial x} \cdot N \right)
\]

\[
\frac{\partial D P \cdot Q}{\partial x} \cdot N = D \left( \frac{\partial P}{\partial x} \cdot N \right) \cdot Q + D P \cdot \left( \frac{\partial Q}{\partial x} \cdot N \right)
\]

\[
\frac{\partial (P Q)}{\partial x} \cdot N = \left( \frac{\partial P}{\partial x} \cdot N \right) Q + \left( D P \cdot \left( \frac{\partial Q}{\partial x} \cdot N \right) \right) Q
\]
One must also say that all constructs commute with 0 and +, but the argument component of ordinary application.

So we consider terms up to the following equalities:

\[
\begin{align*}
\lambda x \ 0 &= 0 \\
\lambda x (M_1 + M_2) &= \lambda x M_1 + \lambda x M_2 \\
0 N &= 0 \\
(M_1 + M_2) N &= M_1 N + M_2 N \\
D0 \cdot N &= 0 \\
D(M_1 + M_2) \cdot N &= DM_1 \cdot N + DM_2 \cdot N \\
DM \cdot 0 &= 0 \\
DM \cdot (N_1 + N_2) &= DM \cdot N_1 + DM \cdot N_2
\end{align*}
\]

We do not have:

\[
\begin{align*}
M 0 &= 0 \\
M (N_1 + N_2) &= M N_1 + M N_2
\end{align*}
\]
Example

\[ \frac{\partial(y \cdot x)}{\partial x} \cdot N = (Dy \cdot N) \cdot x \]

We also need

Schwarz

\[ D^2 M \cdot (N_1, N_2) = D^2 M \cdot (N_2, N_1) \]
Sums come in even if not invited

\[
\left( D^2 (\lambda x^{\text{Bool}} . x \land \lnot x) \cdot (t, f) \right)(0) \rightarrow^* t + f
\]
If we accept infinite sums and rational coefficients, we can write a Taylor expansion of the application:

$$\sum_{n=0}^{\infty} \frac{1}{n!} (D^n M \cdot (N, \ldots, N))_0$$

instead of $M \cdot N$

We can apply this to all the applications in an (ordinary) $\lambda$-term. Then the only ordinary applications we use are of shape $M \cdot 0$
Differential resource calculus

- If $x \in V$ then $x$ is a term;
- if $x \in V$ and $t$ is a term then $\lambda x \ t$ is a term;
- if $s$ is a term and $T$ is a multiset $[t_1, \ldots, t_n]$ of terms, then $\langle s \rangle \ T$ is a term

$\Delta = \text{the set of all resource terms}

\text{Intuition:}

$$\langle s \rangle \ T = (D^n s \cdot (t_1, \ldots, t_n)) \ 0$$

\text{All the constructions are (multi)linear}

\text{Example of multilinearity}

$$\langle s \rangle \ [t_1 + t'_1, t_2, \ldots, t_n] = \langle s \rangle \ [t_1, \ldots, t_n] + \langle s \rangle \ [t'_1, t_2, \ldots, t_n]$$
In a resource term $s$, all the occurrences of a variable $x$ are linear occurrences (in contrast with the ordinary $\lambda$-calculus)

It make sense to define

$$\text{deg}_x s = \text{number of occurrences of } x \text{ in } s$$

Differential $\beta$-reduction becomes:

$$\langle \lambda x \ s \rangle [t_1, \ldots, t_n] \rightarrow \begin{cases} \sum_{\sigma \in \subseteq_n} s \left[ t_{\sigma(1)}/x_1, \ldots, t_{\sigma(n)}/x_n \right] & \text{if } n = \text{deg}_x s \\ 0 & \text{otherwise} \end{cases}$$

where $x_1, \ldots, x_n$ are the $n$ occurrences of $x$ in $s$
Normalization of resource terms

Contrarily to the $\lambda$-calculus, reduction in the resource calculus always terminates, but can yield any sum (including 0)

Fact

Any resource term $s$ reduces to a unique normal form $\text{NF}(s)$ which is a finite sum of resource terms which have no redexes

Example:

$$\text{NF}(\langle\lambda x \langle x \rangle [x, x] \rangle [y, z, z]) = 2 \langle y \rangle [z, z] + 4 \langle z \rangle [y, z]$$
Then we can define the complete Taylor expansion $M^*$ of a $\lambda$-term $M$ as a (generally infinite) linear combination of resource terms with $\geq 0$ rational coefficients:

$$x^* = x$$

$$(\lambda x \ M)^* = \lambda x \ M^* = \sum_{s \in \Delta} M_s^* \lambda x s$$

$$(M \ N)^* = \sum_{n=0}^{\infty} \frac{1}{n!} \langle M^* \rangle [\underbrace{N^*, \ldots, N^*}_n]$$
If we develop these expressions, we get

\[ M^* = \sum_{s \in \mathcal{T}(M)} \frac{1}{m(s)} s \]

where

\[
\begin{align*}
\mathcal{T}(x) &= \{ x \} \\
\mathcal{T}(\lambda x \ M) &= \{ \lambda x s \mid s \in \mathcal{T}(M) \} \\
\mathcal{T}(M \ N) &= \{ \langle s \rangle [t_1, \ldots, t_n] \mid n \in \mathbb{N}, s \in \mathcal{T}(M) \\
 & \quad \text{and } t_1, \ldots, t_n \in \mathcal{T}(N) \}
\end{align*}
\]

and \( m(s) \in \mathbb{N} \setminus \{0\} \) depends only on \( s \)
A theorem

If $M \downarrow x$, then there is exactly one $s \in \mathcal{T}(M)$ such that $\text{NF}(s) \neq 0$ and this $s$ satisfies

$$\text{NF}(s) = m(s) \times$$

This $s$ is the “trace” of execution of $M$ in the Krivine machine, a realistic model of $\lambda$-term execution

$s$ is a “decoration” of $M$ with the multiplicities expressing how many times the various subterms are used

Example: $M = (\lambda y \ y \ (y \ x)) \ \lambda z \ z$

Then $s = \langle \lambda y \langle y \rangle \ [\langle y \rangle \ [x]] \rangle \ [\lambda z \ z, \ \lambda z \ z]$ and $m(s) = 2$

Remark: This theorem may be generalized
Linear logic in a nutshell

A ressource aware typing discipline of programs:

\( M : A \to B \) means \( M \) uses its input (typed by \( A \)) once and only once to produce its output (typed by \( B \))

**Examples**

\[
\begin{align*}
\lambda x^{\text{Bool}} & . \ x \land \neg x : \text{Bool} \Rightarrow \text{Bool} \\
\text{App}_\ell &= \lambda x^A . \ \lambda f^{A \to B} . \ f \ x : A \to (A \to B) \to B \\
\text{App} &= \lambda x^A . \ \lambda f^{A \Rightarrow B} . \ f \ x : A \Rightarrow (A \Rightarrow B) \to B
\end{align*}
\]
Exponentials

Exponential modalities for typing non linear programs:

\[ A \Rightarrow B = !A \multimap B \]

(syntactic version of \( \text{Rel}_! (E, F) \) seen above)

Embedding linear programs into general ones

\[ !A \multimap A \text{ (dereliction)} \]

And coping with erasing and duplication:

\[ !A \multimap 1 \text{ erasing (weakening)} \]
\[ !A \multimap !A \otimes !A \text{ duplication (contraction)} \]
Differential linear logic (DiLL)

Codereliction

\[ A \leadsto !A \]

Differentiation at 0:
\[ F : A \Rightarrow B \leadsto \lambda x^A . (DF \cdot x) 0 : A \leadsto B \]

Coweakening

\[ 1 \leadsto !A \]

Evaluation at 0:
\[ F : A \Rightarrow B \leadsto F 0 : B \]

Cocontraction

\[ !A \otimes !A \leadsto !A \]

Evaluation on a sum:
\[ F : A \Rightarrow B \leadsto \lambda x^A . \lambda y^A . F (x + y) : A \Rightarrow A \Rightarrow B \]
Differential $\lambda$-Calculus and Differential Linear Logic, 20 Years Later (a conference at CIRM - Marseille 2024)

- Pierre Clairambault (CNRS, Aix-Marseille Université) Quantitative semantics in game models
- Ugo Dal Lago (University of Bologna) Reasoning Operationally about Probabilistic Higher-Order Programs
- Thomas Ehrhard (CNRS, Paris Cité) Coherent differentiation
- Zeinab Galal (Bologna) Stable species
- Nicola Gambino (University of Manchester) Generalized species
- Brenda Johnson (Union College) Differential Categories from Functor Calculus
- Marie Kerjean (CNRS, Sorbonne Paris Nord) Introduction to DiLL
Di\textlambda LL 2024, continued

- Delia Kesner (Paris Cité) Non-idempotent intersection types
- Giulio Manzonetto (Paris Cité) Taylor expansion and Böhm trees
- Guy McCusker (Bath) Weighted models
- Jean-Simon Pacaud Lemay (Macquarie University) Differential categories
- Michele Pagani (ENS de Lyon) Automatic differentiation
- Luc Pellissier (Paris-Est Créteil) Taylor expansion in proof nets
- Christine Tasson (ISAE-Supaero) Probabilistic coherence spaces