Differentiation as a monad

Marie Kerjean
CNRS, LIPN, Université Sorbonne Paris Nord
What’s your favorite monad?
What’s your favorite monad?

A monad over a type $A$:

- It *encapsulate* a certain kind of values: $u_A : A \rightarrow M(A)$.
- It allows *computation* on these values: $\mu_A : M(M(A)) \rightarrow M(A)$

**Examples:**

- Partiality: $M : A \mapsto A + \bot$, $u_A : a \mapsto a$
- Non-determinism: $M : A \mapsto \mathcal{P}(A)$, $u_A : a \mapsto \{a\}$
- Effect: $M : A \mapsto (S \rightarrow (A \times S))$, $u_A : a \mapsto (s \mapsto (a, s))$
The continuation monad

\[ u_A : A \Rightarrow ((A \Rightarrow B) \Rightarrow B) \]
\[ a \mapsto \lambda k.ka \]
The continuation monad, twisted

Linear arrow $\circ$: using exactly once its argument

$$u_A : A \Rightarrow ((A \Rightarrow B) \circ B)$$

$$a \mapsto \lambda k. k a$$
The continuation monad

Linear arrow $\circ$: using exactly once its argument

$$u_A : A \circ ((A \Rightarrow B) \circ B)$$

$$a \mapsto \lambda k. ka$$

Making $k$, a non-linear map, linear
The continuation monad

Linear arrow \( \circ \): using exactly once its argument

\[
u_A : A \rightarrow ((A \Rightarrow B) \rightarrow B)
\]
\[
a \mapsto \lambda k. D_0(k) a
\]

Making \( k \), a non-linear map, linear: differentiation
What’s differentiation?

The differential of a function at a point is its *best linear approximation* at that point.
From linearity to quantitative models

<table>
<thead>
<tr>
<th>Functions</th>
<th>Programs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power series</td>
<td>Resources consumption or Probabilistic sums</td>
</tr>
<tr>
<td>( f = \sum_n f_n )</td>
<td>( p(x) = \sum p_n )</td>
</tr>
</tbody>
</table>
From linearity to quantitative models

**Functions**

Power series
\[ f = \sum_n f_n \]

\( f_n \) is \( n \)-linear

**Programs**

Resources consumption or Probabilistic sums
\[ p(x) = \sum p_n \]

\( p_n \) consumes exactly \( n \)-times its resources.

\( f \) is *Taylor*
\[ f = \sum_n \frac{1}{n!} D_0^{(n)} f \]

Programs can be approximated
\[ (M)S = \sum_n \frac{1}{n!} < M > S^\otimes n \]

- Experimentally, quantitative semantics is what gets you higher-order.
- It leads to new proof techniques on \( \lambda \)-calculus.
- A strong link with intersection types.

*Simona Ronchi della Rocca’s talk tomorrow!*

*Even when trying to avoid it, we stumble back on quantitative constructions [Dabrowski, K. 2018]*
## From linearity to quantitative models

<table>
<thead>
<tr>
<th><strong>Functions</strong></th>
<th><strong>Programs</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Power series</strong></td>
<td><strong>Resources consumption or Probabilistic sums</strong></td>
</tr>
<tr>
<td>( f = \sum_n f_n )</td>
<td>( p(x) = \sum p_n )</td>
</tr>
<tr>
<td>( f_n ) is ( n )-linear</td>
<td>( p_n ) consumes exactly ( n )-times its resources.</td>
</tr>
<tr>
<td>( f = \sum_n \frac{1}{n!} D_0^{(n)} f )</td>
<td>Programs can be approximated</td>
</tr>
<tr>
<td></td>
<td>((M) S = \sum_n \frac{1}{n!} &lt; M &gt; S \otimes^n )</td>
</tr>
</tbody>
</table>

- Experimentally, quantitative semantics is what gets you higher-order.
- It leads to new proof techniques on \( \lambda \)-calculus.
- A strong link with intersection types.

---

Simona Ronchi della Rocca’s talk tomorrow!

Even when trying to avoid it, we stumble back on quantitative constructions [Dabrowski, K. 2018]

---

**Core intuition:** Differentials are enough to compute
The quantitative monad

Theorem [K. Lemay 2023]

The following:

- $M : E \rightarrow \mathcal{C}^\infty(E, \mathbb{K})'$
- $u : v \mapsto (f \mapsto D_0(f)(v))$
- $\mu : \delta_\phi \mapsto \sum \frac{1}{n!} \phi^n$

is a monad in quantitative models of $\lambda$-calculus:

\[ !u; \mu = id \iff f = \sum_n \frac{1}{n!} D_0^{(n)} f \]

The monad laws:

\[ u_M; \mu = id \]
\[ M(u); \mu = id \]
\[ \mu_M; \mu = M(\mu); \mu \]

From functional analysis to functional programming, and back
Surprise test

Is it a function?
Surprise test

Is it a function?

Yes, that’s a linear function $f \in \mathcal{L}(\mathbb{R}, \mathbb{R})$
Surprise test

Is it a function?
Surprise test

Is it a function?

Yes, that’s a smooth function \( f \in C^\infty(\mathbb{R}, \mathbb{R}) \)
Is it a function?
Surprise test

Is it a function?

\[ \delta_x \]

No, that’s:

- [ ] A distribution
- [ ] A generalized function
- [ ] That’s the argument to a program.
1 Introduction
   • Quantitative Semantics

2 Different type of functions
   • Smooth functions
   • Linear functions
   • Distribution theory

3 Analytic and Differential Linear Logic

4 Graded Monads in smooth settings
Programs are interpreted as functions...

<table>
<thead>
<tr>
<th>Programs</th>
<th>Logic</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{fun } (x:A)\rightarrow (t:B) )</td>
<td>Proof of ( A \vdash B )</td>
<td>( f : A \rightarrow B. )</td>
</tr>
<tr>
<td>Types</td>
<td>Formulas</td>
<td>Objects</td>
</tr>
<tr>
<td>Execution</td>
<td>Cut-elimination</td>
<td>Equality</td>
</tr>
</tbody>
</table>

.. but *special* ones.

Programs act on programs \( f : C(A, B) \rightarrow C \)

- (AxO) Domains \( A \) and spaces of functions \( C(A, B) \) are of the same kind.
- (AxF) Programs and function compute on several arguments:

\[
f : A \times B \rightarrow C \equiv f : A \rightarrow C(B, C)\]
Programs are interpreted as functions...

<table>
<thead>
<tr>
<th>Programs</th>
<th>Logic</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>fun (x:A) -&gt; (t:B)</td>
<td>Proof of $A \vdash B$</td>
<td>$f : A \rightarrow B$.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Types</th>
<th>Formulas</th>
<th>Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Execution</td>
<td>Cut-elimination</td>
<td>Equality</td>
</tr>
</tbody>
</table>

.. but *special* ones.

Programs act on programs $f : C(A, B) \rightarrow C$

- (AxO) Domains $A$ and spaces of functions $C(A, B)$ are of the same kind.
- (AxF) Programs and function compute on several arguments:

$$f : A \times B \rightarrow C \equiv f : A \rightarrow C(B, C)$$

- Games
- Vector spaces
- Normed spaces
- Topological vector spaces
- Lattices
- Graphs
- Sequences
- Normed spaces
- Topological vector spaces
Interpreting programs by smooth functions

\[ p \colon A \Rightarrow B \quad f \in \mathcal{C}^\infty(A, B) \]

Probabilistic Programming
\[ p \xrightarrow{\alpha} x \]
- Correctness Properties \([D(p)] = D(\mathbb{E}[p])\)
- Completeness Properties \(\forall f, \exists p, [p] = f\)
- **New** programming paradigms \(p = d(q)\)
- **New** mathematical structures \(\mathcal{C}^\infty(E, F)\)

Differentiable Programming
\[ D(p_1; p_2) = D(p_1); D(p_2) \]

Convenient vector spaces
A first interpretation of Higher-Order Smooth Functions

\[(\text{AxF}): \mathcal{C}^\infty(A \times B, C) \simeq \mathcal{C}^\infty(A, \mathcal{C}^\infty(B, C))\]

Frölicher, Kriegl, Michor (1997)
Blute, Ehrhard, Tasson (2012)
<table>
<thead>
<tr>
<th>Programs</th>
<th>Logic</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>fun (x:A) -&gt; (t:B)</td>
<td>Proof of $A \vdash B$</td>
<td>$f : A \rightarrow B$.</td>
</tr>
<tr>
<td>Types</td>
<td>Formulas</td>
<td>Objects</td>
</tr>
<tr>
<td>Execution</td>
<td>Cut-elimination</td>
<td>Equality</td>
</tr>
</tbody>
</table>
Perspective

<table>
<thead>
<tr>
<th>Programs</th>
<th>Logic</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{fun } (x:A) \rightarrow (t:B) )</td>
<td>Proof of ( A \vdash B )</td>
<td>( f : A \rightarrow B ).</td>
</tr>
<tr>
<td>Types</td>
<td>Formulas</td>
<td>Objects</td>
</tr>
<tr>
<td>Execution</td>
<td>Cut-elimination</td>
<td>Equality</td>
</tr>
</tbody>
</table>

- Probabilistic and resources \( \lambda \)-calculi \([\text{NR06]}\)
- Differential Linear Logic \([\text{Gir87]}\)
- Linear Logic \([\text{Gir87]}\)
- \( \lambda \)-calculus
- Min. Logic
- Normal functors
- Vectorial Models
Interpreting programs by Linear Functions

\[[p] \in \mathcal{L}(A, B)\]

(Ax0): If $B$ is a complete or metrizable space, then so is $\mathcal{L}(A, B)$.

Trickier for $A$ though

(AxF):

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

▶ Always true algebraically.
Interpreting programs by Linear Functions

\[ [p] \in \mathcal{L}(A, B) \]

(Ax0): If \( B \) is a complete or metrizable space, then so is \( \mathcal{L}(A, B) \).

Trickier for \( A \) though

(AxF):

\[ \mathcal{L}_B(A \otimes_B B, C) \simeq \mathcal{L}_B(A, \mathcal{L}_B(B, C)) \]

\[ \begin{align*}
\text{▶} & \quad \text{Always true algebraically.} \\
\text{▶} & \quad \text{Topologically, it depends on the set } B \subset \mathcal{P}(A) \text{ of bounded sets on which uniform convergence must be enforced.} \\
\text{▶} & \quad \text{MANY topological tensor products: } \otimes_\beta, \otimes_\sigma, \otimes_\mu, \otimes_\varepsilon. \\
\text{▶} & \quad \text{MANY duals: } E'_B := \mathcal{L}_B(E, \mathbb{R})
\end{align*} \]

\textbf{WE ARE MISSING AN IMPORTANT CRITERIA}
Not Not ... Who’s there?
Not Not ... Who’s there?

\[ ((A \Rightarrow \bot) \Rightarrow \bot) \simeq A \]

\[ C^\infty(C^\infty(A, \mathbb{K}), \mathbb{K}) \simeq A \]

**No one:** not a chance for $A$ smooth enough
Not Not ... Who’s there?

$$((A \Rightarrow \perp) \Rightarrow \perp) \simeq A$$

$$C^\infty(C^\infty(A, K), K) \simeq A$$

_No one:_ not a chance for $A$ smooth enough

$$((A \rightarrow \perp) \rightarrow \perp) \simeq A$$

$$\mathcal{L}(\mathcal{L}(A, K), K) \simeq A$$

_A lot of people!: _Reflexive topological vector spaces.

_We have plenty of examples!_

► Finite dimensional vector spaces
► Hilbert spaces
► Spaces on which an orthogonality relation can be defined …

In general, reflexive spaces enjoy _poor stability properties_.
× higher-order, × tensor product.
Interpreting types by reflexive topological vector spaces

$C^\infty(\mathbb{R}^n, \mathbb{R})$ is not finite dimensional

Nuclear Fréchet spaces are reflexive and complete

Let us take the other way around, through Nuclear, Complete+Metrizable (=Fréchet) spaces.

Coherent Banach spaces, Girard 2004, a norm is too restrictive

a norm is too restrictive
Polarization as a solution to reflexivity

Semantics for polarized MLL: Melliès Chiralities

\[(\mathcal{P}^{op}, \otimes, 1) \perp (\mathcal{N}, \&\&, \perp) \]

Replacing \((\text{AxF})\) with:

\[\mathcal{N}(\uparrow p \otimes n^{\perp L}, m) \simeq \mathcal{N}(\uparrow p, n \&\& m)\]

Interpreting formulas by two categories of topological vector spaces, with a contravariant equivalence interpreting the involutive linear negation
Polarization as a solution to reflexivity

Notation: $E' := \mathcal{L}(E, \mathbb{R})$

Grothendieck, Produits tensoriels topologiques et espaces nucléaires, 1958

Melliès, A micrological study of negation, APAL 2017

Linear implications and reflexivity

Property: \( E \simeq (E'_\beta)' \iff E \text{ barrelled and } E \text{ weakly quasi complete.} \)

Barrelled spaces (Bourbaki): there for Banach-Steinhauss theorem.

Theorem

- Barrelled and weak quasi-complete form a model of polarized calculus (Melliès’ Chiralities).
- Banach-Steinhaus is exactly \((\text{AxF})!\):

\[
\mathcal{N}(\uparrow p \otimes n^\perp_L, m) \simeq \mathcal{N}(\uparrow p, n \triangleright m)
\]
Mixing Linear and Non-Linear Proofs: here comes the fun!
Not not ... Who’s there?

\((A \Rightarrow \bot) \rightarrow \bot\)

\(\mathcal{L}(\mathcal{C}_\infty(A, \mathbb{R}), \mathbb{R}) = \mathcal{C}_\infty(A, \mathbb{R})'\)
Not not ... Who’s there ?

\[(A \Rightarrow \bot) \dashv \bot\]

\[\mathcal{L}(\mathcal{C}^\infty(A, \mathbb{R}), \mathbb{R}) = \mathcal{C}^\infty(A, \mathbb{R})'\]

**Semantics**

**Programs**

**Distributions**
\[\phi \in \mathcal{C}^\infty(A, \mathbb{R})'\]

**Context**
\[C : (p : A \rightarrow \bot) \mapsto (\text{value} : \bot)\]

e.g.: \[\delta_x : f \mapsto f(x)\]

\[\llbracket \rrbracket(x) : p \rightarrow p[x]\]
Not not ... Who’s there?

\[(A \Rightarrow \bot) \rightarrow \bot\]

\[L(\mathcal{C}^\infty(A, \mathbb{R}), \mathbb{R}) = \mathcal{C}^\infty(A, \mathbb{R})'\]

Semantics

Distributions

\[\phi \in \mathcal{C}^\infty(A, \mathbb{R})'\]

e.g.: \[\delta_x : f \mapsto f(x)\]

Reflexivity:

\[f(x) = \delta_x(f)\]

Programs

Context

\[C : (p : A \rightarrow \bot) \mapsto (\text{value} : \bot)\]

\[\llbracket\cdot\rrbracket(x) : p \mapsto p[x]\]

\[p(x) = \llbracket\llbracket\cdot\rrbracket(x)\|p\rrbracket\]
Not not ... Who’s there?

\[(A \Rightarrow \bot) \rightarrow \bot\]

\[\mathcal{L}(C^\infty(A, \mathbb{R}), \mathbb{R}) = C^\infty(A, \mathbb{R})'\]

### Semantics

<table>
<thead>
<tr>
<th>Distributions</th>
<th>Programs</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi \in C^\infty(A, \mathbb{R})'$</td>
<td>Context</td>
</tr>
<tr>
<td>e.g.: $\delta_x : f \mapsto f(x)$</td>
<td>$C : (p : A \rightarrow \bot) \mapsto (\text{value} : \bot)$</td>
</tr>
</tbody>
</table>

#### Reflexivity

\[f(x) = \delta_x(f)\]

#### Differentiation

\[D_0(f)(x) = < D_0(-)(x) | f >\]

---

Laurent Schwartz, *Théorie des distributions*, 1950
Distributions: Linear Contexts for Non-Linear Programs

\[ C^\infty(E, F)' \]

- (AxO) for distributions:
  - \( C^\infty(\mathbb{R}^n, \mathbb{R}) \) is always Nuclear Fréchet and \( C^\infty(\mathbb{R}^n, \mathbb{R})' \) is Nuclear DF.
  - If \( F \) is Fréchet, then \( C^\infty(\mathbb{R}^n, F) \) is Fréchet
  - Higher order: a bit of work.

- (AxF) for distributions:
  - For linear maps: \( \mathcal{L}_\beta(\hat{E}, \mathcal{L}_\beta(F, G)) \simeq \mathcal{L}_\beta(E \otimes_\beta F, G) \)
  - For smooth maps: \( C^\infty(E, C^\infty(F, G)) \simeq C^\infty(E \times F, G) \)
  - From one to another:

  **Schwartz’ Kernel Theorem**

  \[ C^\infty(E, \mathbb{K})' \hat{\otimes} C^\infty(F, \mathbb{K})' \simeq C^\infty(E \times F, \mathbb{K})' \]
A monoidal operation on distributions

\[(\phi \in \mathcal{C}^\infty(E, \mathbb{R})' \otimes \psi \in \mathcal{C}^\infty(E, \mathbb{R})') \mapsto ?\]

**Convolution**, the monoidal operation on distributions:

\[\phi \ast \psi := f \mapsto \phi(x \mapsto \psi(y \mapsto f(x + y)))\]

Different from \(\phi + \psi : f \mapsto \phi(f) + \psi(g)\)

**Examples:**

\[\delta_x \ast \delta_y = \delta_{x+y}\]
\[\delta_x \ast D_0(-)(v) = D_x(-)(v)\]
\[D_0(-)(v) \ast D_0(-)(v) = D_0^{(2)}(-)(v)\]

There is no "multiplication" extending from functions to distributions, this is our multiplication!
Quantitative semantics, another look

$$\forall x, \forall v, f(x) = \sum_n \frac{1}{n!} D_0^{(n)} f(x)$$
Quantitative semantics, another look

$$\forall f, \forall x, \forall v, \langle f | \delta_x \rangle = \sum_n \frac{1}{n!} \langle f | D_0^{(n)}(-)(x) \rangle$$
Quantitative semantics, another look

\[ \forall x, \forall v, \delta_x = \sum_n \frac{1}{n!} D_0^{(n)}(-)(x) \]
Quantitative semantics, another look

\[ \forall x, \forall y, \delta_x = \sum_{n} \frac{1}{n!} D_0^{(n)} (-)(x) \]
Quantitative semantics, another look

$$\forall x, \forall u, \delta_x = \sum_{n} \frac{1}{n!} \underbrace{D_0(-)(x) \ast \cdots \ast D_0(-)(x)}_{D_0(-)(x)^n}$$
Quantitative semantics, another look

\[ \forall x, \forall v, \delta_x = \sum_n \frac{1}{n!} D_0(_{-})(x) \ast \cdots \ast D_0(_{-})(x) \]

\[ e^x = \sum_n \frac{1}{n!} x^n \quad id = e^* \circ (D_0(_{-})) \]
Quantitative semantics, another look

\[ \forall x, \forall v, \delta_x = \sum_n \frac{1}{n!} D_0(-)(x) \ast \cdots \ast D_0(-)(x) \]

\[ e^x = \sum_n \frac{1}{n!} x^n \quad id = e^* \circ (D_0(-)) \]

A Quantitative Monad

- A functor \( E \mapsto C^\infty(E, \mathbb{R}) \) acting on a subcategory \( \mathcal{L} \) of topological vector spaces and linear maps.
- Differentiation as a unit: \( u : x \mapsto D_0(-)(x) \)
- The convolutional exponential as a multiplication: \( \mu : \delta_\phi \mapsto \sum_n \frac{1}{n!} \phi^* x^n \)

Monad \( \leadsto \forall f \in \mathcal{L}(A, B) \cong C^\infty(A, B), f \) is Taylor.

Examples: Relational model, Weighted Relational Model, Species, Nuclear Fréchet spaces
It was never about the quantitative semantics of $\lambda$-calculus.

Differential Linear Logic: from resources to distributions, from discrete to continuous settings
Exponential rules of (Differential) Linear Logic

<table>
<thead>
<tr>
<th>Programs</th>
<th>Logic</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fun (x:A)-&gt; (t:B)</code></td>
<td>Proof of $A \vdash B$</td>
<td>$f : A \rightarrow B$.</td>
</tr>
</tbody>
</table>
Exponential rules of (Differential) Linear Logic

<table>
<thead>
<tr>
<th>Programs</th>
<th>Logic</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>fun (x:A) -&gt; (t:B)</code></td>
<td>Proof of $A \vdash B$</td>
<td>$f : A \rightarrow B$.</td>
</tr>
</tbody>
</table>

Resources calculi | LINEAR LOGIC | Topological vector spaces |

**Exponential connectives:**

\[
\begin{align*}
[!A] &:= C^\infty([A], K)' \\
[?B] &:= C^\infty([B]', K)
\end{align*}
\]

*Linear Logic, Jean-Yves Girard 1987*

*Differential Interaction Nets, Thomas Erhard and Laurent Regnier, 2006*
A decomposition of the implication

\[ A \Rightarrow B \simeq !A \multimap B \]
Linear Logic

A decomposition of the implication

\[ A \Rightarrow B \simeq !A \rightarrow B \]

- Usual non-linear implication

A linear proof is in particular non-linear.

\[ A \vdash B \text{ is linear.} \quad !A \vdash B \text{ is non-linear.} \]

\[ \frac{A \vdash \Gamma}{!A \vdash \Gamma} \quad \text{dereliction} \]

*Slogan:* ! in the hypotheses, speaking of resources.
Linear Logic

A decomposition of the implication

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

- Usual non-linear implication
- Linear implication

A linear proof is in particular non-linear.

\[ A \vdash B \text{ is linear.} \quad !A \vdash B \text{ is non-linear.} \]

\[
\frac{A \vdash \Gamma}{!A \vdash \Gamma} \quad \text{dereliction}
\]

\textit{Slogan:} ! in the hypotheses, speaking of resources.
Linear Logic

A decomposition of the implication

\[ A \Rightarrow B \simeq !A \circ B \]

- Usual non-linear implication
- Linear implication
- Exponential: Usually, the duplicable copies of \( A \).

A linear proof is in particular non-linear.

\( A \vdash B \) is linear. \( !A \vdash B \) is non-linear.

\[ \frac{A \vdash \Gamma}{!A \vdash \Gamma} \text{ dereliction} \]

Slogan: ! in the hypotheses, speaking of resources.
Differential Linear Logic: co-structural rules

\[
\begin{align*}
\ell &: A \vdash B \\
\ell &: !A \vdash B \\
\text{d, dereliction}
\end{align*}
\]

linear \leftrightarrow non-linear.

\[
\begin{align*}
\ell &: A \vdash B \\
\ell &: !A \vdash B \\
\text{d, dereliction}
\end{align*}
\]

\[
\begin{align*}
f &: !A \vdash B \\
\text{D}_0(f) &: A \vdash B \\
\bar{\text{d}}, \text{co-dereliction}
\end{align*}
\]

non-linear \leftrightarrow linear.
Differential Linear Logic: co-structural rules

\[ \frac{\ell : A \vdash B}{\ell : !A \vdash B} \quad \text{linear proof} \]

\[ \frac{\vdash \Delta, v : A}{\vdash \Delta, (f \mapsto D_0(f)(v)) : !A} \quad \text{non-linear proof} \]

\( \ell : A \vdash B \) \quad \text{Linear Logic} \quad \boxed{!A \vdash B} \quad \text{linear} \to \text{non-linear.} \)

\( \vdash \Delta, v : A \) \quad \text{non-linear} \to \text{linear}
Differential Linear Logic: co-structural rules

\[
\ell : A \vdash B \quad \text{linear proof}
\]

\[
\ell : !A \vdash B \quad \text{non-linear proof}
\]

\[
\ell : !A \vdash B \\
\ell : A \vdash B \Rightarrow \quad \text{dereliction}
\]

Cut-elimination:

\[
\Gamma, v : !A \\
\Gamma, !A \\
\ell : A \vdash B \\
\ell : !A \vdash B \\
\Gamma, \Delta \\
\Gamma, \Delta \\
\Gamma, A \quad \Delta, A \Downarrow \\
\Gamma, \Delta \\
cut
\]

\[
\Gamma, v : A \\
\Delta, (f \mapsto D_0(f)(v)) : !A \\
\Gamma, \Delta \\
\Gamma, \Delta \\
cut
\]

linear \leftrightarrow non-linear.
Differential Linear Logic: co-structural rules

linear proof

$A \vdash B$

Non-linear proof

$\ell : A \vdash B$

linear $\leftrightarrow$ non-linear.

Cut-elimination:

$\vdash \Gamma, \nu : A$

$\vdash \Gamma, \nu : A$

$\ell \! : A \vdash B$

$\ell \! : A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$

$\ell \! : \! A \vdash B$
Dereliction and co-dereliction:

\[ A \vdash B \quad \text{DiLL} \quad !A \vdash B \]

**linear proof**
\[
\ell : A \vdash B \\
\ell : !A \vdash B \\
\text{linear} \leftrightarrow \text{non-linear.}
\]

**non-linear proof**
\[
\vdash \Delta, v : A \\
\vdash \Delta, (f \mapsto D_0(f)(v)) : !A \\
\text{non-linear} \leftrightarrow \text{linear}
\]

**Cut-elimination:**
\[
\vdash \Gamma, v : A \\
\vdash \Gamma, D_0(-)(v) : !A \\
\vdash \Gamma, \Delta
\]

\[
\ell : A \vdash B \\
\ell : !A \vdash B \\
\text{d, dereliction}
\]

\[
\ell : \Gamma \vdash \Delta, A_{\perp} \\
\ell \vdash \Delta, A_{\perp} \\
\text{cut}
\]

\[
\vdash \Gamma, v : A \\
\ell \vdash \Delta, A_{\perp} \\
\vdash \Gamma, \Delta, D_0(\ell)(v) = \ell(v) \\
\text{cut}
\]

\[\Rightarrow\]
(Co)-weakening

\[
\frac{c : \vdash \Gamma}{c \cdot!A : \vdash \Gamma} \quad \frac{\vdash \Gamma}{\vdash \Gamma, \delta_0 : !A}
\]

The constant function is non-linear

One can evaluate a function at 0

(Co)-contraction

\[
\frac{x : !A, y : !A \vdash g(x, y) : \Gamma}{x : !A \vdash g(x, x) : \Gamma} \quad \frac{\vdash \Gamma, \phi : !A}{\vdash \Gamma, \Delta, \psi \cdot \phi : !A}
\]

The multiplication of scalar functions

Convolution of distributions
The function $cst_1$ is neutral for scalar multiplication.

The dirac at 0 is neutral for the convolution.

\[
\phi \ast \psi(\ell) = \phi(\ell)\psi(cst_1) + \psi(\ell)\phi(cst_1)
\]

\[
D_0(f \cdot g) = D_0(f) \cdot g(0) + D_0(g) \cdot f(0)
\]
Symmetric cut-elimination procedures

\[ \bar{d}; w = 0 \text{ and } \bar{w}; d = 0 \]
\[ D_0(cst_1) = 0 \text{ and } \ell(0) = 0 \]

\[ (\phi \ast \psi)(cst_1) = \phi(cst_1) \cdot \psi(cst_1) \]
\[ (f \cdot g)(0) = f(0) \cdot g(0) \]

\[ \otimes = \cdot \text{ in } \mathbb{R} \]
Finitary differential Linear Logic

The first version by Erhrard and Regnier in 2006:

\[
\begin{align*}
\Gamma & \vdash \Gamma, \text{cst}_1 : ?A \\
\Gamma & \vdash \Gamma, \delta_0 : !A \\
\Gamma & \vdash \Gamma, f : ?A, g : ?A \\
\Gamma & \vdash \Gamma, f.g : ?A \\
\Gamma & \vdash \Gamma, \ell : ?A \\
\Gamma & \vdash \Gamma, \ell : A \\
\Gamma & \vdash \Gamma, x : A \\
\Gamma & \vdash \Gamma, \text{D}_0(\text{D}_0(x)) : !A
\end{align*}
\]

\text{It’s a maths world.}
This is fine.
Higher-Order via promotion

Exponential rules of Linear Logic (Resources)

\[
\frac{\vdash \Gamma}{\vdash \Gamma, \text{cst}_1 : ?A} \quad w \\
\frac{\vdash \Gamma, f : ?A, g : ?A}{\vdash \Gamma, f.g : ?A} \quad \text{c} \\
\frac{\vdash \Gamma}{\vdash \Gamma, \ell : ?A} \quad \text{d} \\
\frac{!\Gamma \vdash x : A}{!\Gamma \vdash \delta_x : A} \quad p
\]

Exponential rules added by Differential Linear Logic (Distributions)

\[
\frac{\vdash \Gamma}{\vdash \Gamma, \delta_0 : !A} \quad \bar{w} \\
\frac{\vdash \Gamma, \phi : !A}{\vdash \Gamma, \Delta, \psi \star \phi : !A} \quad \bar{c} \\
\frac{\vdash \Gamma, x : A}{\vdash \Gamma, D_0(\_)(x) : !A} \quad \bar{d}
\]

The promotion rule \( p : !A \rightarrow !!A \quad \delta_x \mapsto \delta_{\delta_x} : \)

- Makes (!, ⁂) a co-monad : \( p ; d = \text{id} \).
- Is a co-monoidal operation on !A : \( p ; c = c ; p \otimes p \)
- The cut-elimination between \( p \) and \( \bar{d} \) express the chain rule:
  \[
  D_0(g \circ f) = D_{f(0)}g \circ D_0f \\
  \bar{d}; p = \bar{w} \otimes \bar{d}; p \otimes \bar{d}; \bar{c}
  \]
Exponential rules of Linear Logic (Resources or functions)

\[
\begin{align*}
\frac{\vdash \Gamma}{\vdash \Gamma, \text{cst}_1 : ?A} \quad w & \quad \frac{\vdash \Gamma, f : ?A, g : ?A}{\vdash \Gamma, f \cdot g : ?A} \quad c & \quad \frac{\vdash \Gamma, \ell : ?A}{\vdash \Gamma, \ell : ?A} \quad d & \quad \frac{\vdash !\Gamma \vdash x : A}{\vdash !\Gamma \vdash \delta_x : !A} \quad \bar{p}
\end{align*}
\]

Exponential rules added by Differential Linear Logic (Distributions)

\[
\begin{align*}
\frac{\vdash \Gamma}{\vdash \Gamma, \delta_0 : !A} \quad \bar{w} & \quad \frac{\vdash \Gamma, \phi : !A}{\vdash \Gamma, \Delta, \psi \ast \phi : !A} \quad \bar{c} & \quad \frac{\vdash \Gamma, x : A}{\vdash \Gamma, D_0(\_)(x) : !A} \quad \bar{d} & \quad \frac{\vdash ?\Gamma \vdash x : A}{\vdash ?\Gamma \vdash \_ : ?A} \quad \bar{p}
\end{align*}
\]

Digging \( p : !A \to !!A \):

- ▶ \( p ; d = \text{id} \).
- ▶ \( p ; c = c ; p \otimes p \).
- ▶ \( \bar{d}; p = \bar{w} \otimes \bar{d}; p \otimes \bar{d}; \bar{c} \)

Co-digging \( ?\bar{p} : !!A \to !A \):

- ▶ \( \bar{d}; \bar{p} = \text{id} \).
- ▶ \( \bar{c}; \bar{p} = \bar{p} \otimes \bar{p}; \bar{c} \).
- ▶ \( \bar{p}; d = c; \bar{p} \otimes d; w \otimes d \)
**Codigging**

**Exponential rules of Linear Logic (Resources or functions)**

\[
\begin{align*}
\vdash \Gamma & \quad w \\
\vdash \Gamma, \text{cst}_1 : ?A & \\
\vdash \Gamma, f : ?A, g : ?A & \quad c \\
\vdash \Gamma, \ell : ?A & \quad d \\
\vdash !\Gamma & \quad x : A \\
\vdash !\Gamma & \quad \delta_x : !A
\end{align*}
\]

**Exponential rules added by Differential Linear Logic (Distributions)**

\[
\begin{align*}
\vdash \Gamma & \quad \bar{w} \\
\vdash \Gamma, \delta_0 : !A & \\
\vdash \Gamma, \phi : !A & \\
\vdash \Delta, \psi : !A & \quad \bar{c} \\
\vdash \Gamma, \Delta, \psi \ast \phi : !A & \\
\vdash \Gamma, D_0(\_)(x) : !A & \\
\vdash ?\Gamma & \quad x : A \\
\vdash ?\Gamma & \quad \_ : ?A
\end{align*}
\]

**Digging** $p : !A \to !!A$:

- $p; d = \text{id}$.
- $p; c = c; p \otimes p$
- $\bar{d}; p = \bar{w} \otimes \bar{d}; p \otimes \bar{d}; \bar{c}$

**Co-digging** $\bar{p} : !!A \to !A$:

- $\bar{d}; \bar{p} = \text{id}$
- $D_0(g) = \text{id}$.
- $\bar{c}; \bar{p} = \bar{p} \otimes \bar{p}; \bar{c}$
- $g(x + y) = g(x) \ast g(y)$
- $\bar{p}; d = c; \bar{p} \otimes d; w \otimes d$
The missing rule of Differential Linear Logic

Digging \( p : !A \rightarrow !!A \):
\[ \begin{align*}
\triangleright & \quad p; d = \text{id}. \\
\triangleright & \quad p; c = c; p \otimes p \\
\triangleright & \quad \overline{d}; p = \overline{w} \otimes \overline{d}; p \otimes \overline{d}; \overline{c}
\end{align*} \]

Co-digging \( \overline{p} : !!A \rightarrow !A \):
\[ \begin{align*}
\triangleright & \quad \overline{d}; \overline{p} = \text{id} \\
\triangleright & \quad \overline{c}; \overline{p} = \overline{p} \otimes \overline{p}; \overline{c} \\
\triangleright & \quad \overline{p}; d = c; \overline{p} \otimes d; w \otimes d
\end{align*} \]

\[\text{Implicit definition of the exponential:}\]
\[ g = \exp^* : \phi \mapsto \sum_n \frac{1}{n!} \phi^*\]
\[ \overline{p} : \delta \phi \mapsto \sum_n \frac{1}{n!} \phi^{*n} \]
The missing rule of Differential Linear Logic

Digging $p : !A \rightarrow !!A$:
- $p; d = id.$
- $p; c = c; p \otimes p$
- $\overline{d}; p = \overline{w} \otimes \overline{d}; p \otimes \overline{d}; \overline{c}$

Co-digging $\overline{p} : !!A \rightarrow !A$: $g : !A \Rightarrow !A$
- $\overline{d}; \overline{p} = id$  $D_0(g) = id.$
- $\overline{c}; \overline{p} = \overline{p} \otimes \overline{p}; \overline{c}$  $g(x + y) = g(x) \ast g(y)$
- $\overline{p}; d = c; \overline{p} \otimes d; w \otimes d$

Implicit definition of the exponential:
$$g = exp^* : \phi \mapsto \sum_n \frac{1}{n!} \phi^*  \quad  \overline{p} : \delta_\phi \mapsto \sum_n \frac{1}{n!} \phi_*^n$$

The co-chain rule:
$$\overline{p}(\delta_\phi)(\ell) = \sum_{n \geq 0} \frac{1}{n!} \phi_*^n(\ell) = \sum_{n \geq 1} \frac{n}{n!} \phi(\ell) \cdot \phi_*^{n-1}(cst_1) = \phi(\ell) \cdot \overline{p}(\delta_\phi)(cst_1) = c; \overline{p} \otimes d; w \otimes d$$
The missing rule of Differential Linear Logic

Digging \( p : !A \rightarrow !!A: \)

\[ \begin{align*}
&\quad p; d = \text{id}. \\
&\quad p; c = c; p \otimes p \\
&\quad \overline{d}; p = \overline{w} \otimes \overline{d}; p \otimes \overline{d}; \overline{c}
\end{align*} \]

Co-digging \( \overline{p} : !!A \rightarrow !A: g : !A \Rightarrow !A \)

\[ \begin{align*}
&\quad \overline{d}; \overline{p} = \text{id} \quad D_0(g) = \text{id}. \\
&\quad \overline{c}; \overline{p} = \overline{p} \otimes \overline{p}; \overline{c} \quad g(x + y) = g(x) \ast g(y) \\
&\quad \overline{p}; d = c; \overline{p} \otimes d; w \otimes d \quad ?
\end{align*} \]

Implicit definition of the exponential:

\[ g = \exp^* : \phi \mapsto \sum_n \frac{1}{n!} \phi^* \quad \overline{p} : \delta \phi \mapsto \sum_n \frac{1}{n!} \phi^{*n} \]

The co-chain rule:

\[ \overline{p}(\delta \phi)(\ell) = \sum_{n \geq 0} \frac{1}{n!} \phi^{*n}(\ell) = \sum_{n \geq 1} \frac{n}{n!} \phi(\ell) \cdot \phi^{*(n-1)}(\text{cst}_1) = \phi(\ell) \cdot \overline{p}(\delta \phi)(\text{cst}_1) = c; \overline{p} \otimes d; w \otimes d \]

The monadic rules:

\[ !\overline{d}; \overline{p} = \text{id} \quad \forall \nu, \overline{p}(\delta_{D_0(\omega)}(\nu)) = \delta_{\nu} \quad \forall \nu, \forall f, \sum_n \frac{1}{n!} D_0^{(n)} f(\nu) = f(\nu) \]
A completely uniform logical and categorical structure

Exponential connectives:

\[ [!A] := C^\infty([A], \mathbb{K})' \quad [?A] := C^\infty([A]', \mathbb{K}) \]

Exponential connectives:

\[ \vdash \Gamma \quad w \quad \vdash \Gamma, f : ?A, g : ?A \quad c \]

\[ \vdash \Gamma, \ell : A \quad d \quad \vdash \Gamma, \ell : ?A \]

\[ \vdash \Gamma \quad \overline{w} \quad \vdash \Gamma, \phi : !A \quad \overline{c} \quad \vdash \Gamma, \phi : !A \]

\[ \vdash \Gamma, \Delta, \psi * \phi : !A \]

\[ \vdash \Gamma, x : A \quad \overline{d} \quad \vdash ?\Gamma, x : A \quad \overline{p} \quad \vdash ?\Gamma, e^x[-] : ?A \]

Linear Logic
[Giard1987]

Differential Linear Logic
[Ehrhard&Regnier2006] [Kerjean & Pacaud Lemay 2023]
A reason for this symmetry

Exponential connectives:

\[[!A] := C^\infty([A], \mathbb{K})' \quad [?A] := C^\infty([A]', \mathbb{K})\]

Do you remember the Laplace transformation?

*A continuous version of a power series*

\[\mathcal{L} : \begin{cases} !E & \rightarrow ?E \\ \phi & \mapsto (\ell^E' \mapsto \phi(y^E \mapsto e^{\langle \ell | y \rangle})) \end{cases}\]

\[\mathcal{L}(\overline{w}, \overline{c}, \overline{d}, \overline{p}) = w, c, d, p\]
Let’s make things concrete

- $M : E \to !E := \llbracket ([E] \Rightarrow \bot) \Rightarrow \bot \rrbracket = \mathcal{C}^\infty([E], \mathbb{K})'$
- $u : v \mapsto (f \mapsto D_0(f)(v))$
- $\mu : \delta_\phi \mapsto \sum \frac{1}{n!} \phi^*^n$
Let’s make things concrete

- \( M : E \rightarrow !E := \llbracket (\llbracket E \rrbracket \Rightarrow \bot) \rightarrow \bot \rrbracket = C^\infty(\llbracket E \rrbracket, \mathbb{K})' \)
- \( u : v \mapsto (f \mapsto D_0(f)(v)) \)
- \( \mu : \delta \phi \mapsto \sum \frac{1}{n!} \phi^n \)

The way I make things concrete
Existential questions

Does $\bar{p} = e^*$ even exist?

Bad news: sums need to converge.

At least at every point: $(\forall f, \forall \phi, \sum_n \frac{1}{n!} \phi^n(f)) \in \mathbb{R}$

- In discrete models of computations (e.g: relations over sets), sums are union, not an issue.
- In continuous models of computations, well...
Convolutional exponential VS exponential maps

\[(AxC)\] Proofs/Programs/Functions must compose

\[\overline{p} : \delta_{\phi} \mapsto \sum_{n} \frac{1}{n!} \phi^n\]
Convolutional exponential VS exponential maps

(AxC) Proofs/Programs/Functions must compose

\[ \overline{p} : \delta_{\delta x} \mapsto \sum \frac{1}{n!} \delta_{nx} = \left( f \mapsto \sum \frac{1}{n!} f(nx) \right) \]
Convolutional exponential VS exponential maps

(AxC) Proofs/Programs/Functions must compose

\[ p: \]
\[
\overline{p}(\delta_{\delta x})(x \mapsto e^x) = \sum \frac{1}{n!} e^{nx} = e^x \quad \checkmark
\]
\[
\overline{p}(\delta_{\delta x})(x \mapsto e^{e^x}) = \sum \frac{1}{n!} e^{e^{nx}} \quad \times
\]

This example is due to T. Ehrhard

- \( p \) and \( \overline{p} \) do not mix well.
- We need to quantify over the exponential growth of the functions \( \overline{p} \) is applied to.
Grading Exponentials

\[ A \rightarrow B \]
Linear programs/proofs/functions
using exactly once their \textbf{resource}

\[ !A \rightarrow B \]
Usual programs/proofs/functions

\[ !_n A \rightarrow B \]

- \( n \)-linear functions.
- Programs/proofs using exactly \( n \)-times their resource.
- Quantitative semantics: \( !A = \sum !_n A \).

Exponentials indexed by semi-rings \( S \)

\[ \forall s \in S, !_s A \rightarrow B \]

Jean-Yves Girard, Andre Scedrov, Philip J. Scott, 1992
Martin Hoffman, Ugo Dal Lago, 2009

Applications in implicit complexity and differential privacy.
From resources to differential equations

\[
\begin{align*}
\vdash \Gamma & \quad w \\
\vdash \Gamma, \text{cst}_1 : \text{?}A & \\
\vdash \Gamma, \ell : A & \\
\vdash \Gamma, \ell : \text{?}A & \\
\vdash \Gamma, f : \text{?}A, g : \text{?}A & \\
\vdash \Gamma, f \cdot g : \text{?}A & \\
\vdash \Gamma, \phi : \text{!!}A & \\
\vdash \Gamma, \psi : \text{!!}A & \\
\vdash \Gamma, x : \text{?}A & \\
\vdash \Gamma, \Delta, \psi \star \phi : \text{!!}A & \\
\vdash \Gamma, D_0(-)(x) : \text{!!}A & \\
\vdash ?\Gamma, e^{x|-} : \text{?}A & \\
\end{align*}
\]
From resources to differential equations

Grading LL: a story of resources, again

\[ \vdash \Gamma \quad w \]
\[ \vdash \Gamma, \text{cst}_1 : ?_0A \]

\[ \vdash \Gamma, \ell : A \quad d \]
\[ \vdash \Gamma, \ell : ?_1A \]

\[ \vdash \Gamma, f : ?_xA, g : ?_yA \quad c \]
\[ \vdash \Gamma, f.g : ?_{x+y}A \]

\[ \vdash \delta_0 : !A \quad \bar{w} \]
\[ \vdash \phi : !A \quad \bar{c} \]
\[ \vdash \Delta, \psi : !A \]
\[ \vdash \Delta, \psi * \phi : !A \]

\[ \vdash \Gamma, x : A \quad \bar{d} \]
\[ \vdash \Gamma, D_0(\_)(x) : !A \]

\[ ?\Gamma \vdash x : A \quad \bar{p} \]
\[ ?\Gamma \vdash e^{x\mathcal{L}} : ?A \]
From resources to differential equations

Grading LL: a story of resources, again

\[
\frac{\vdash \Gamma}{\vdash \Gamma, \text{cst}_1 : ?A} \quad \frac{\vdash \Gamma, f : ?A, g : ?A}{\vdash \Gamma, f . g : ?A} \quad \frac{\vdash \Gamma}{\vdash \Gamma, \ell : A} \quad \frac{\vdash \Gamma, f : ?A, g : ?A}{\vdash \Gamma, f . g : ?A}
\]

\[
\frac{\vdash \Gamma, \ell : A}{\vdash \Gamma, \ell : ?A} \quad \frac{\vdash \Gamma}{\vdash \Gamma, \ell : ?A}
\]

Grading DiLL: well, not so clear

\[
\frac{\vdash \delta_0 : !?A}{\vdash \delta_0 : !?A} \quad \frac{\vdash \Gamma, \phi : !?A}{\vdash \Gamma, \Delta, \psi \ast \phi : !?A} \quad \frac{\vdash \Delta, \psi : !?A}{\vdash \Delta, \psi : !?A}
\]

\[
\frac{\vdash \Gamma}{\vdash \Gamma, \ell : ?A} \quad \frac{\vdash \Gamma}{\vdash \Gamma, \ell : ?A}
\]

\[
\frac{\vdash \Gamma, x : A}{\vdash \Gamma, D_0(\_)(x) : !?A} \quad \frac{\vdash \Gamma, x : A}{\vdash \Gamma, x : A} \quad \frac{\vdash \Gamma, x : A}{\vdash \Gamma, x : A}
\]

\[
\frac{\vdash \delta_0 : !?A}{\vdash \delta_0 : !?A} \quad \frac{\vdash \Gamma, \phi : !?A}{\vdash \Gamma, \Delta, \psi \ast \phi : !?A} \quad \frac{\vdash \Delta, \psi : !?A}{\vdash \Delta, \psi : !?A}
\]

\[
\frac{\vdash \Gamma}{\vdash \Gamma, \ell : ?A} \quad \frac{\vdash \Gamma}{\vdash \Gamma, \ell : ?A}
\]

\[
\frac{\vdash \Gamma, x : A}{\vdash \Gamma, x : A} \quad \frac{\vdash \Gamma, x : A}{\vdash \Gamma, x : A}
\]

\[
\frac{\vdash \delta_0 : !?A}{\vdash \delta_0 : !?A} \quad \frac{\vdash \Gamma, \phi : !?A}{\vdash \Gamma, \Delta, \psi \ast \phi : !?A} \quad \frac{\vdash \Delta, \psi : !?A}{\vdash \Delta, \psi : !?A}
\]

\[
\frac{\vdash \Gamma}{\vdash \Gamma, \ell : ?A} \quad \frac{\vdash \Gamma}{\vdash \Gamma, \ell : ?A}
\]

\[
\frac{\vdash \Gamma, x : A}{\vdash \Gamma, x : A} \quad \frac{\vdash \Gamma, x : A}{\vdash \Gamma, x : A}
\]
Joining Quantitative, Graded and Differential Linear Logic

\((\text{AxC})\) Proofs/Programs/Functions must compose

\[
\mu : \delta x \mapsto \sum \frac{1}{n!} \delta_n x = \left( f \mapsto \sum \frac{1}{n!} f(nx) \right)
\]

\[
\mu(\delta x)(x \mapsto e^x) = \sum \frac{1}{n!} e^{nx} = e^{ex} n \checkmark
\]

\[
\mu(\delta x)(x \mapsto e^{ex}) = \sum \frac{1}{n!} e^{enx} \times
\]

The quantitative monad does not apply to all functions, but only to those whose convergence is exponentially bound, according to some Young function \(\theta\).

\[
\theta, m(F) := \{ f : F' \to \mathbb{C}, \forall m, \exists K, \forall z, |f(z)| \leq Ke^{\theta(m||z||)} \}.
\]

\[
\overline{p} : !\theta_1 (!!\theta_2(E)) \to !(\theta_1 e^{\theta_2*})^*(E)
\]

\[\text{Gannoun, Hachaichi, Ouerdiane, et Rezgui, Un théorème de dualité entre espaces de fonctions holomorphes à croissance exponentielles, 1999}\]
Les fonctions parlent aux fonctions

\[ \!_{\theta,m}(F) := \{ f : F \to \mathbb{C}, \forall m, \exists K, \forall z, |f(z)| \leq Ke^{\theta(m||z||)} \}'. \]

issue: \( ||z|| : F \) needs to be normed.

- One single norm is too restrictive: we want to quantify over the convergence of the derivative of each function.
- The power of Fréchet spaces comes from their descriptions as a countable limit of Banach spaces:

\[ F' := \lim_{p} N'_p \]

\[ \!_{\theta}F := \lim_{m,p}(?_{m,\theta}F_p)' \]
Les fonctions parlent aux fonctions
We have a quantitative and graded monad of Nuclear Dual of Fréchet spaces.

\[ !\theta F_p : \{ f : F_p \to \mathbb{C}, \forall m, \exists K, \forall z, |f(z)| \leq K e^{\theta(m||z||)} \} \]

\[ !\theta F := (\lim_{m,p} (?_{m,\theta F})')' \]

The space of Young functions is a semi-ring with a new duality operation \((\_)^*\).

\[ \Theta : \{ \theta, +, (\_ \times e^-), (\_)^* \} \]
## Recap

<table>
<thead>
<tr>
<th>Programs</th>
<th>Logic</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{fun (x:A) -&gt; (t:B)} )</td>
<td>Proof of ( A \vdash B )</td>
<td>( f : A \rightarrow B ).</td>
</tr>
<tr>
<td>Types</td>
<td>Formulas</td>
<td>Objects</td>
</tr>
<tr>
<td>Execution</td>
<td>Cut-elimination</td>
<td>Equality</td>
</tr>
</tbody>
</table>
Recap

<table>
<thead>
<tr>
<th>Programs</th>
<th>Logic</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{fun } (x:A) \to (t:B) )</td>
<td>Proof of  ( A \vdash B )</td>
<td>( f : A \to B ).</td>
</tr>
<tr>
<td>Types</td>
<td>Formulas</td>
<td>Objects</td>
</tr>
<tr>
<td>Execution</td>
<td>Cut-elimination</td>
<td>Equality</td>
</tr>
</tbody>
</table>

- **Types**
- **Formulas**
- **Objects**

- **Execution**
- **Cut-elimination**
- **Equality**

- **Monadic reformulation of resource calculi?**
- **Resource and probabilistic \( \lambda \)-calculus [Ehr04]**
- **Differential Linear Logic [Ehrhard06]**
- **Vectorial Models**
- **Linear Logic [Gir87]**
- **Smooth and graded semantics**
- **Min. Logic**
- **Normal functors**
## Recap

### Programs

<table>
<thead>
<tr>
<th>Types</th>
<th>Logic</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>fun (x:A) -&gt; t:B</td>
<td>Proof of A ⊢ B</td>
<td>f : A → B</td>
</tr>
</tbody>
</table>

### Execution
- Cut-elimination
- Equality

---

### Differentiable Programming

- Monadic reformulation of resource calculi?
- Analytic Linear Logic
- Smooth and graded semantics

---

### Linear Relations

- Resource and probabilistic λ-calculus [Ehr04]
- Differential Linear Logic [Ehrhard06]
- Linear Logic [Gir87]
- Vectorial Models
- λ-calculus
- Min. Logic
- Normal functors
Recap

<table>
<thead>
<tr>
<th>Programs</th>
<th>Logic</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{fun} (x:A) \rightarrow (t:B) )</td>
<td>Proof of ( A \vdash B )</td>
<td>( f : A \rightarrow B ).</td>
</tr>
</tbody>
</table>

Types Formulas Objects
Execution Cut-elimination Equality

Differentiable Programming

Monadic reformulation of resource calculi?

Resource and probabilistic \( \lambda \)-calculus [Ehr04]

Analytic Linear Logic

Smooth and graded semantics

Differential Linear Logic [Ehrhard06]

Vectorial Models

Linear Logic [Gir87]

\( \lambda \)-calculus

Min. Logic

Normal functors
Recap

<table>
<thead>
<tr>
<th>Programs</th>
<th>Logic</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{fun} \ (x:A) \rightarrow (t:B) )</td>
<td>Proof of ( A \vdash B )</td>
<td>( f : A \rightarrow B ).</td>
</tr>
</tbody>
</table>

- Types
- Formulas
- Execution
- Cut-elimination
- Objects
- Equality

Differentiable Programming

- Monadic reformulation of resource calculi?
- Resource and probabilistic \( \lambda \)-calculus [Ehr04]

Analytic Linear Logic

- Automatic Differentiation [80s]
- \( \lambda \)-calculus

Differential Operators

- Smooth and graded semantics
- Vectorial Models
- Linear Logic [Gir87]
- Differential Linear Logic [Ehrhard06]
- Dialectica [Göd58]
- Min. Logic
- Normal functors
Perspectives
Quantitative Semantics: Approximating functions by Polynomials

- Taylor: Orthogonal Basis = \{X^n\}
  - recurrence: \(T_{n+1} = XT_n\)
  - composition \(T_n \circ T_m = T_{nm}\)

- Other Bases? Chebychev?
  - recurrence: \(T_{n+2} = 2T_{n+1} - T_n\)
  - composition \(T_n \circ T_m = T_{nm}\)

- Characterization of approximation and orthogonality?
Grading with partial differential operators

Grading by Linear Partial Differential Equations with constant coef.

\[ \boxed{\begin{align*}
[!_{\mathcal{D}}A] & := D((C^\infty([A], \mathbb{K}')) \quad [?_{\mathcal{D}}A] := D^{-1}(C^\infty([A'], \mathbb{K})) \\
parameters \ of \ the \ equations & \quad solutions \ of \ the \ equations
\end{align*}} \]

\[ \begin{align*}
D(\_ & ) = f \quad \phi \circ D = \_ \\
\end{align*} \]

**Monoid:** \((\mathcal{D}, \circ, \text{Id})\)

Breuvart, K. Mirwasser, 2023
Grading with partial differential operators

Grading by Linear Partial Differential Equations with constant coef.

\[ ![D_A] := D((C^\infty([A], \mathbb{K}))') \quad ![?D_A] := D^{-1}(C^\infty([A'], \mathbb{K})) \]

parameters of the equations

solutions of the equations

For \( D \) an LPDOcc: \((\phi \circ D) * \psi = (\phi * \psi) \circ D\) \quad \( D(E_D * f) = f \)

Monoid: \((D, \circ, Id)\)

Breuvart, K. Mirwasser, 2023
The computational content of differentiation.

The codereliction of differential proof nets: In terms of polarity in linear logic [23], the ∀→-free constraint characterizes the formulas of intuitionistic logic that can be built only from positive connectives (⊕, ⊗, 0, 1, !) and the why-not connective (“?”). In this framework, Markov’s principle expresses that from such a ∀→-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.

This can also be witnessed by identifying the computational content of Dialectica as a CPS style differential λ-calculus.[PMP,K 22]
Open questions

- \( \overline{p} \) and \( p \) do not interact well: cut-elimination?

- More intricate differential operators semi-rings? Higher-order methods? Can we embed approximate resolution methods in the sequent calculus?

- Can we express resolution methods in differential \( \lambda \)-calculi?

- Can we make the categorical semantics of differentiation closer to the one of type theory?
Conclusion

Take away

- The semantics of $\lambda$-calculus is not as much about discrete structures than about approximating continuous ones.

- The notion of linear type $\_ \to \_\_\$ has been influential in functional programming. Let’s now make use of the distribution type $\_\_\$, which internalizes external transformations on programs.

- Functional analysis and functional programming might enrich each other: the former gives the latter new concepts, the latter gives the former new structures.

Thank you for listening!