Research activities

Thomas Ehrhard
IRIF UMR 8243
CNRS et Université de Paris

Septembre 2021

1 Scientific contributions

I present my scientific activities since the beginning of my PhD in 1986. I also describe some scientific directions for future work and projects, in paragraphs entitled "*Plan for future research*" inserted in this presentation.

1.1 Summary: main scientific contributions

Here is a list of what I currently consider as my main scientific contributions, since the beginning of my career, with some references.

I am currently mainly working on topics related to Coherent Differentiation, see 1.7.5.

- *Hypercoherence semantics of Linear Logic.* A denotational model of classical Linear Logic (LL) where "first-order functions" are sequential in the sense of Milner, Vuillemin and Sazonov, avoiding the use of intensional notions such as Berry and Curien's Sequential Algorithms (and, more generally, deterministic strategies in games). I have also exhibited strong connections between this model and sequential algorithms (extensional collapse) as well as another independent characterization of its morphisms. [BE91b, BE91a, CE94, BE93, Ehr93, BE94, Ehr96, Ehr99, Ehr99].

- *Relational semantics: non-uniform coherence spaces, extensional collapse.* I introduced new coherence spaces which, unlike Girard's, are fully compatible with the relational model of LL. This model yields surprising new interpretations of the exponential and its points can be incoherent with themselves (quite a surprising feature opening new perspectives in semantics). In the same line I proved that the extensional collapse of the relational model is the Scott model of LL by means of a completely new duality. [BEM07, BEM09, ECS10, Ehr12a, BE00, BE01, Ehr04, BEM12, Ehr12b, Ehr20c].

- *Vector space based models of LL: Köthe spaces and finiteness spaces.* I discovered new models of LL where formulas are interpreted as topological vector spaces and proofs as linear and continuous functions. The first uses locally convex spaces (lcs) and the second one, a "purely algebraic" analog of lcs's discovered by Lefschetz in the 1940's. In the associated Kleisli categories, morphisms are generalized power series (analytic functions). This enforces the basic intuition that LL is the logic of (infinite dimensional) linear algebra. [Ehr02, Ehr05, BET12, Ehr10, Ehr18].

- *Differential lambda-calculus and LL.* Inspired by these models I introduced the idea that proofs (programs) can be differentiated wrt. their hypotheses (parameters): such operations are freely available in these models. I showed that the corresponding extension of LL consists in endowing the exponential with rules which are dual to the usual structural rules and that the associated Taylor expansion can be used as the foundation for a new program approximation theory, deeply related with Böhm trees. [EL06, ER06a, EL07, Ehr10, BEM10, CES10, BEM10, EG16, Ehr19, ER03, ER06b, ER08, EL10b, EL10a, Ehr18].

- *Probabilisitic coherence spaces and functional languages.* I developed this model introduced by Girard, refining its definition, extending it to the exponential and recursive types and proving adequacy and full abstraction theorems wrt. various functional probabilistic programming languages (including a version of Levy's call-by-push-value). I extended this model to "continuous types" (such as the real line) using cones and a notion of stable functions between them. This extension has been shown to be conservative by one of my PhD students. [EPT11, EPT14, CEPT17, EPT18b, DE11, EPT18a, ET19, Ehr19, Ehr20a, Ehr20b].

1.2 Introduction: categorical semantics

Proof systems and programming languages are formalisms designed for specific purposes: formalizing mathematical proofs, writing computer programs. As is now well-known, such formalisms are deeply related by a long-standing correspondence: the Curry-Howard isomorphism, discovered in the 1950's and extended in various directions since then. This correspondence is called an *isomorphism* because it works at three levels: logical formulas are related to types, mathematical proofs are related to (functional) programs and cut-elimination of proofs is related to program execution (presented as rewriting).

Among the various extensions of the Curry-Howard isomorphism, the most remarkable is probably the one to *classical logic* by Griffin [Gri90]. Indeed, before this fundamental discovery which relates purely classical principles such as the *Peirce Law* with primitives of functional languages allowing to handle the evaluation context (the most well known being call/cc), the Curry-Howard isomorphism was restricted to intuitionistic logic and deeply associated with its effectiveness features such as the disjunction or the existence property. The classical extension extends this effectiveness to a new interactive setting where these properties do not hold as such.

In spite of this rich entanglement of concepts coming from mathematical logic and computer science, the design of logical or computational systems remains a fairly difficult task because of the many possible choices concerning their syntax and, more importantly, their operational properties. This is the syntax *camembert mou*[1] problem already singled out by Girard in the 1980's. Fortunately however, the Curry-Howard correspondence has a kind of third leg, which can be generically called *categorical semantics*.

Category theory has been introduced by mathematicians who observed that some kinds of algebraic invariants (for instance, groups or modules) can be associated with geometric objects (for instance, topological spaces or manifolds) and that such correspondences can – and actually must – be extended to morphisms. This leads to the the crucial idea that mathematical structures have to be considered together with their morphisms whose global features somehow witness the properties of these objects; this is the key idea of category theory. In that sense, categories must be considered as algebraic objects generalizing monoids.

Category theory also became a very powerful tool for abstracting mathematical constructions or situations appearing in many distinct but analogous settings: (co)limits, adjunctions, monads etc and still plays an important role in this respect, besides its status of a generalized algebra.

In the mid 1960's, Christopher Stratchey stressed the lack of mathematical foundations for the many programming languages which were becoming more and more essential in the current practice of computer scientists; the absence of formal mathematical *semantics* led to serious issues as to the meaning of programs. Here "meaning" has not to be taken in an abstract philosophical sense, but in a very concrete one: what will be the effect of the program when run on an actual computer? It is the meeting of Stratchey and Dana Scott in Oxford in 1969 which gave rise to a new field of theoretical computer science: *denotational semantics* which we can also call categorical semantics since its main idea consists in interpreting types as objects in a category, and programs as morphisms. They were actually considering a particular category, namely that of complete lattices and "Scott continuous" functions (monotone functions preserving directed lubs), a setting strongly suggested by the Rice Shapiro theorem on partial recursive functions.

[1] Soft cheese. This pleasant expression was used by him for qualifying all kinds of more or less artificial syntactic artifact, during his famous DEA lectures in 1986-87 that I've been lucky enough to attend.

This at the same time allowed Dana Scott to settle a long standing problem: find a "functional interpretation"[2] of the pure lambda-calculus introduced by Church in the 1930's and soon recognized as a suitable setting for defining general computable functions (later proved equivalent to Turing machines and all other ways of defining partial computable functions).

Such models of programming languages have several positive outcomes.

- First they answer Stratchey's request[3] of a formal semantics for programs allowing to express their meaning mathematically and to prove properties on them. For instance, the fact that an expression of type ι (the type of integers) will actually be evaluated to a unique integer (independently on the evaluation strategy) can be checked syntactically by proving a Church-Rosser property of the corresponding rewriting system, but also semantically using a suitable model (Scott domains, coherence spaces etc).

- One of the main features of denotational semantics is that it is *modular*, meaning that the semantics of a program can be computed from the semantics of its various parts. This means in particular that when one wants to extend the language with an additional construct, it suffices to find an interpretation in the model for this new construct without changing all the interpretation.

- Models usually contain much more objects and morphisms than those which can be defined using the programming language under consideration. This means that one can find in the model new ideas for extending or refining the syntax one started from. This is typically what happened with the discovery of Linear Logic by Jean-Yves Girard or with my introduction of differential constructs in the lambda-calculus and in linear logic.

- Models are quite agnostic as to the presentation of the language and of its operational semantics, as long as the main requirement that the interpretation is invariant under reduction is preserved. So by providing global abstract settings where many different syntaxes can be embedded, models provide a more canonical grasp to programming languages and logical systems, reducing the *camembert mou* effect.

1.3 Some contributions to semantics

After Gérard Huet's and Pierre-Louis Curien's DEA's lectures[4] in 1985-86 and Girard's and Krivine's DEA's lectures[5] in 1986-87, all insisting on the importance of models and presenting their beauties, I was quickly convinced that this was a perfect research area for a PhD, that I started under the supervision of Pierre-Louis Curien in 1986.

1.3.1 The Calculus of Constructions During a DEA internship in INRIA Rocquencourt in 1986, I had the opportunity to meet Thierry Coquand and Gérard Huet who introduced me to the recently discovered Calculus of Constructions which, at that time, was being implemented in Caml as a proof assistant. In parallel I also learned about Girard's system F and its qualitative domain and coherence space denotational semantics [Gir86], but at that time, inspired by Curien's *categorical combinators*[6], I was more interested by the global categorical structures of models than by their various concrete instances (continuous semantics, stable semantics etc.).

Therefore I developed in my PhD thesis various notions of categorical models for the Calculus of Constructions, some of them based on the notion of fibration (Grothendieck) and relating them with other approaches such as Cartmell's *categories with attributes*. In these settings I proved results relating

[2] By this I mean an interpretation of lambda-terms as functions acting on a suitable non trivial space. Such a quest is most natural because the basic intuition behind lambda-terms is that they represent functions.

[3] Unfortunately he died too early to really see the flourishing of his ideas in the late 1970's and 1980's.

[4] DEA d'informatique fondamentale de Paris VII, the DEA I actually took.

[5] DEA de logique de Paris VII; these were incredibly exciting series of lectures since Girard had just discovered linear logic and Krivine was setting down the bases of his realizability program, with a quite refreshing viewpoint on the lambda-calculus.

[6] Which were used by Gérard Huet's group at INRIA as a kind of assembly language in Caml's implementation: the *categorical abstract machine* (CAM) later replaced by the category-independent ZAM in Xavier Leroy's new implementation.

terms and types with their interpretation, this work is reported in [CE87, Ehr89a, Ehr89b]. In 1988, after my first discussions with Antonio Bucciarelli at ENS, I understood that digging into the fine grain structures of models would lead to deeper insights and we started to work together on *sequentiality*, a notion introduced independently by Jean Vuillemin, Robin Milner and Vladimir Sazonov in the 1970's.

1.3.2 Denotational semantics, adequacy, full abstraction As already mentioned, the first denotational models of programming languages were based on complete lattices (soon replaced by Scott domains which generalize them) and Scott continuous functions.

Scott semantics. In these domains, *compact* (or *isolated*) elements play the role of "finite generators". Scott continuity means that, for getting a finite[7] information from the output of a function, a finite information on the input is sufficient. This accounts quite naturally for the finite nature of computations expressed in particular by the Rice Shapiro theorem.

More specifically, the key ingredient in the objects interpreting types in this kind of model (usually called *domains*) is an order relation whose intended meaning is that the smaller and element is, the less it is defined. Typically, such a domain has a least element usually denoted $\bot$, and which represents the completely undefined value. The simplest example is the *flat domain of integers* $\{\bot, 0, 1, \dots\}$ ordered as follows: $x < y$ iff $x = \bot$ and $y \neq \bot$. In this domain $\bot$ represents the completely undefined integer[8] and all the other elements are fully defined integers.

The abstract programming language PCF. One of the great ideas of Dana Scott in his investigations on the denotational semantics of programming languages, has been to introduce a paradigmatic functional programming language powerful enough for being Turing complete, and simple enough to be fully formally described in a few lines. This language, usually called PCF (for *programming computable functions*) is a simply typed lambda-calculus extended with one ground type (for us, it will always be the type of integers, but the original versions also featured a type of booleans) with basic constructs on this type (successor, predecessor, conditional) as well as a *fixpoint construct* allowing to define recursive programs.

Just like the pure lambda-calculus, PCF is a Turing complete functional language, but one has to be aware that the sources of this Turing completeness for these two languages are quite different. In the pure lambda-calculus, it comes from the complete freedom in applying terms to terms, involving in particular *self-application*[9], whereas in PCF term application must respect the quite restrictive simple type discipline, preventing self-application; there Turing completeness results of course from the direct availability of fixpoint operators.

PCF is given together with a rewriting system which includes β-reduction as well as quite natural rules dealing with integers and fixpoints. This rewriting system is easily seen to be Church-Rosser. Moreover, a simple sub-relation of this basic set of rules defines a *reduction strategy*[10], it is called here *weak head reduction* and has the following main feature: if a closed term M of type ι can be reduced to an integer n using the general reduction, then it can be reduced to n by weak head reduction (with possibly more reduction steps of course). In other words, weak head reduction is a complete strategy for computing values[11] in PCF.

Adequacy and observational equivalence. Such models also satisfy an important property called *adequacy*. It means that, if the denotational interpretation of a closed term of type ι is the integer n, then the reduction of this term leads to the value n. The converse of this statement easily results from

[7] In this setting, finite means compact, or isolated.

[8] Any closed program of type ι which loops forever and never produces any result; we have all seen too many such programs...

[9] The key ingredient in the encoding of fixpoint operators in the pure lambda-calculus.

[10] In the sense that it is deterministic: given a term, there is at most one redex to be reduced, according to this reduction relation.

[11] At the end of the day, in PCF, one only computes integers which are the only things one can observe: a function of type *e.g.* $\iota \Rightarrow \iota$ is an infinite device which cannot be observed in a finite time. Observing the term which defines it is not the same thing as observing the function itself, see Section 1.3.2.

the *soundness* of the interpretation (if the term M reduces to the term M', then M and M' have the same interpretation). Adequacy is a kind of normalization property and its proof typically relies on reducibility (aka. realizability) techniques.

It has an important consequence regarding *observational equivalence*, a central concept in the study of programming languages. Simplifying a little bit[12] two closed terms M and M' of the same type σ are observationally equivalent if, whenever given any closed term C of type $\sigma \Rightarrow \iota$, both terms $C\,M$ and $C\,M'$ either diverge or reduce to the same integer n. In other words, M and M' are observationally equivalent if you can use indifferently M or M' as "subroutines" in any context[13].

An immediate consequence of adequacy (and soundness) is that, if two terms of the same type have the same interpretation in the considered model, then they are observationally equivalent. This means that the model can be used as a tool to prove the observational equivalence of programs. Indeed, proving observational equivalence is notoriously difficult as it involves a universal quantification over all possible contexts and, even worse, it is not a modular concept (you cannot prove observational equivalence of programs by reducing this task to substructures of these programs: if you know that M_i is observationally equivalent to M'_i for $i = 1, 2$, you cannot infer *a priori* that $M_1\,M_2$ is observationally equivalent to $M'_1\,M'_2$).

Full abstraction. When the converse of this implication holds, the model is said to be *fully abstract*. This means that the model not only can be used as a tool for proving observational equivalence in the language, but also that it is a complete tool for this purpose. Moreover, the mere existence of a fully abstract model implies that observational equivalence is a modular concept.

The Scott model of PCF is not fully abstract for this language. The reason boils down to the existence of a Scott continuous function por[14] of type $\iota \Rightarrow (\iota \Rightarrow \iota)$ (to be understood as a function taking two integers and returning an integer) which behaves as follows:

- if one of the two arguments is 0, the result is 0, even if the other argument is $\bot$
- if both arguments are 1 the result is 1.

Then it is easy to define two closed terms M and M' of type of type $(\iota \Rightarrow (\iota \Rightarrow \iota)) \Rightarrow \iota$ whose interpretations in the Scott model take different values on por but which can be proved[15] observationally equivalent: the Scott model is not fully abstract for PCF.

One reason for this lack of full abstraction is that the Scott model hosts elements such as por, which is not definable in PCF. The main feature of por is that it is not possible to say which of its two arguments it evaluates first: if it were the first one then we would have $\mathsf{por}(\bot, 0) = \bot$ and if it were the second one we would have $\mathsf{por}(0, \bot) = \bot$ (and we know that it looks at its arguments as otherwise it would be a constant function, but we have $\mathsf{por}(1, 1) = 1$ and $\mathsf{por}(\bot, 0) = 0$).

1.4 Sequentiality and strong stability

In other words por is not a *sequential function*. This notion was introduced independently by Jean Vuillemin, Robin Milner and Vladimir Sazonov in the 1970's. If $\mathsf{N}_\bot$ stands for the flat domain of natural numbers (see 1.3.2) and $k \in \mathbb{N}$ then it is easy to define the notion of sequential function $f : \mathsf{N}_\bot^k \to \mathsf{N}_\bot$: it is a monotone function (wrt. the product order) such that if $f(x) = \bot$ then there are two possibilities:

- either $f(y) = \bot$ for any $y \geq x$ (that is: there is no way to add information to the input so as to obtain an information at the output)
- or there is an $i \in \{1, \ldots, k\}$ such that $x_i = \bot$ and, if $y \geq x$ and $f(y) \neq \bot$, then $y_i \neq \bot$; such an i is called a *sequentiality index* for f at x. This i is an input that is undefined in x and that is necessary to f for yielding an output.

[12] The official definition admits non closed terms and uses the concept of *context* which is slightly more tricky to define; on closed terms the equivalence relations are the same.

[13] It is worthwhile to note that as most of this theory, this concept is insensitive to complexity aspects.

[14] For *parallel or*.

[15] Typically using another adequate model where por does not exist.

1.4.1 Generalizing sequential algorithms. For trying to define a fully abstract model of PCF on a semantical basis[16] Gérard Berry and Pierre-Louis Curien introduced the concept of *sequential algorithm*. Indeed, as it is presented above, the notion of sequential function cannot be generalized so as to define a *cartesian closed category*[17]. As explained in the introduction of [Ehr99], this is due to the fact that a sequential function can have several sequentiality indexes at a given x and sequential algorithms contain explicit choices of indices. These explicit choices allow to define a cartesian closed category which has been later understood as a category of games and deterministic strategies[18].

I started to work on this topic with Bucciarelli in 1989 and we first generalized sequential algorithms, presenting them as pairs of a sequential function and of a "choice function" for sequentiality indices, going in the opposite direction [BE93]. One major idea of this construction was the observation that *indices* (like the k "positions" in the cartesian product $\mathsf{N}_\perp^k$) can be understood as linear functions (in the sense of linear logic, see below) from the considered space (for instance $\mathsf{N}_\perp^k$) to the "Sierpinsky space", that is, the domain with two elements $\perp < \top$. Indeed, the objects of this category, called *sequential structures*, are domains equipped with "abstract cells" which are Sierpinsky-valued linear maps.

1.4.2 The discovery of strong stability. Later on, playing once more with the definition of sequentiality, we understood that it could be expressed in a completely different way, as a preservation property. We observed that, if we say that a subset A of $\mathsf{N}_\perp^k$ is coherent if it is finite, non empty, and if, for any index i, if all the elements x of A satisfy $x_i \neq \perp$ then there is some n such that $x_i = n$ for all $x \in A$, then being sequential from $\mathsf{N}_\perp^k$ to $\mathsf{N}_\perp^l$ is equivalent to preserving coherence and commuting with the glb's of coherent sets.

This was a striking observation for two reasons:

- First, it generalized in some sense Berry's idea of *stability* [Ber78] rediscovered later by Girard [Gir86], a weakening a sequentiality leading to cartesian closeness (in the definition above, replace "coherent" by "upper-bounded", which easily implies coherence in the sense above, but there are sets which are coherent in our sense but not in Berry's).

- Second, unlike the notion of "index" sequentiality is based on, the much more general concept of coherence can be extended to function spaces, leading to a cartesian closed category of *qualitative domains with coherence* and *strongly stable functions*.

This was the object of our articles [BE91b, BE91a, BE94]. I found the situation quite interesting since we had two radically different ways to extend sequentiality to functional types: the "game theoretic" approach of sequential algorithms and the "extensional" approach of strong stability. So I tried to figure out if the two notions are equivalent is some sense.

More precisely I wanted to define a "projection" from sequential algorithms to strongly stable functions whose main effect would be to forget the explicit choice of indices contained in sequential algorithms. This however requires some care, and such a projection cannot be defined for all sequential algorithms[19]. But more importantly, the general notion of qualitative domains with coherence (qDC) 1.4.2 that we introduced in [BE91b] appeared soon to be too general for proving the surjectivity of this projection.

Therefore I progressively added some restrictive conditions on qDC's until one memorable sunday of 1992 where I suddenly understood that all these unrelated conditions boiled down to the extremely simple definition of an admittedly non intuitive new concept: *hypercoherence spaces*.

[16] Milner already proposed a construction of such a model based on PCF syntax.

[17] That is, a category with finite products and where, given two objects X and Y, one can define a space $X \Rightarrow Y$ which represents the space of morphisms from X to Y; such categories are necessary to interpret PCF.

[18] This idea has been further developed in particular in [Nic94, HO00, AJM00] to define fully abstract models, by adding to determinism an additional condition stipulating that a strategy cannot access the whole history of earlier moves.

[19] Because a sequential algorithm of type $(\iota \Rightarrow \iota) \Rightarrow \iota$, for instance, can use "intensional" information about its argument to produce a result in ι, for instance it can make a difference between the completely undefined function (mapped to $\perp$) and the program of type $\iota \Rightarrow \iota$ which tests the value of the argument before looping forever (mapped for instance to 0).

A posteriori, hypercoherence spaces are quite similar to Girard's coherence spaces that led him to the discovery of linear logic [Gir87]. A coherence space is simply a set called the *web*[20] together with a binary reflexive and symmetric relation on this set called *coherence* relation. The elements of the web should be considered as elementary pieces of information, and the binary relation express when two such elementary information are compatible. Then a "compound information" is simply a clique, that is, a subset of the web such that any two elements are related by the coherence relation. Ordered under inclusion, the cliques of a coherence space form a domain with a least element (the empty clique) which is a special case of Berry's notion of dI-domain[21].

It is quite interesting to observe that, before coherence spaces, Girard used *qualitative domains* in his stable denotational semantics of system F, which are more general than coherence spaces: a qD is a web together with a collection of subsets of this web, let's call them *configurations*, closed under directed unions, subsets and containing all singletons, so it is essentially a (potentially infinite dimensional) simplicial complex. Our notion of qDC's was using qD's in a crucial way and we will see that they are still essential with hypercoherences. Of course a coherence space can be seen as a qD with a very special property: to check that a finite subset of the web $\{a_1, \ldots, a_n\}$ is accepted, it suffices to check that all pairs $\{a_i, a_j\}$ are. This fundamentally binary nature of coherence vanishes when moving from stability to strong stability[22].

1.4.3 Intermezzo: Linear Logic. This is a good place to say a few more words about linear logic (LL) which plays a central role in my work since my discovery of hypercoherence spaces in 1992. The reader acquainted with LL can safely skip this section.

Before replacing qD's with coherence spaces Girard observed that a stable function from a qD X to a qD Y can be described by means of its *trace* which is a set of couples (x_0, b) where x_0 is a finite configuration of X and b belongs to the web of Y; the idea is quite simple: x_0 is a minimal configuration where the stable function produces b. The set of all such pairs (x_0, b) of a finite configuration of X and of a point of Y can be equipped with a qD structure so as to define a new qD denoted $X \Rightarrow Y$ whose configurations are in bijective correspondence with the stable functions from X to Y: this is the key step for proving cartesian closeness. There is an interesting asymmetry in this construction between the left-hand side and the right-hand side: on the left one uses finite configurations (which are sets of elements of the web) whereas on the right one uses single elements of the web. This suggests restricting one's attention to the special pairs where the left-hand component is a singleton: through the above correspondence between configurations in $X \Rightarrow Y$ and stable functions, this corresponds to a special class of stable functions which are quite easy to characterize. These are the stable functions which commute with all existing unions of configurations, they are called *linear* (stable) functions. So the operation $X \Rightarrow Y$ on qD's actually splits into two simpler one:

$$X \Rightarrow Y \;=\; !X \multimap Y$$

where $Z \multimap Y$ is a qD whose configurations represent the linear (stable) maps from the qD Z to Y whereas $!X$ is a new construction that Girard called *exponential* for quite good reasons.

Focusing first on $Z \multimap Y$, Girard observed that the special case where $Y = \bot$, the special qD whose web is a singleton (there is only one up to unique iso) is quite interesting: $Z \multimap \bot$ has the same web as Z (up to trivial iso) and if one performes this operation twice, defining $(Z \multimap \bot) \multimap \bot$, one retrieves Z as soon as one requires it to be a coherence space instead of an arbitrary qD: it is precisely for this reason that Girard moved from general qD's to coherence spaces in his presentation of the stable semantics.

[20] It is quite natural to assume that this set is at most countable.

[21] Berry proved that these domains, together with stable functions, form a cartesian closed category which is a model of PCF but he didn't observe that coherence spaces and stable functions form a cartesian closed sub-category sufficient large for defining this model; in other words, all objects considered by Berry were coherence spaces, but he didn't notice!

[22] A typical example is provided by the well known *Gustave's functions* like $g : \mathsf{N}_\bot^3 \to \mathsf{N}_\bot$ which maps $(\bot, 0, 1)$, $(1, \bot, 0)$ and $(0, 1, \bot)$ to 0 and $(0, 0, 0)$ to 1. Taken pairwise, the 3 first bahaviours are perfectly sequential, for instance mapping $(1, \bot, 0)$ and $(0, 1, \bot)$ to 0 and $(0, 0, 0)$ to 1 is a sequential behaviour: test first the first argument and depending on the obtained value decide which input testing next. But these 3 elementary behaviours are not globally coherent, that is, as a whole, g is not a sequential function, although it is stable in Berry's and Girard's sense. This kind of phenomenon can of course be observed for any arities, not only 3.

This suggests a beautiful analogy with linear algebra: coherence spaces are like vector spaces[23] and linear (stable) functions are like linear maps, $\bot$ is like the field (1-dimensional space), $Z \multimap Y$ is like the vector space of linear maps from Z to Y so that $Z \multimap \bot$ is like the dual of Z and the iso between $(Z \multimap \bot) \multimap \bot$ is like the well known canonical iso between the vector space E and its bidual E^{**}. Pushing the analogy forward Girard introduced a tensor product of coherence spaces $X \otimes Y$ as well as a direct product $X \,\&\, Y$, together with their dual operations co-tensor $X \parr Y$ and direct sum $X \oplus Y$[24].

Concerning "!" Girard understood that it was a functor on the category of coherence spaces and linear maps similar to a "symmetric tensor algebra" whose structures and properties allowed to recover the cartesian closed structure of coherence spaces and stable functions[25].

But Girard is a logician and clearly for him "$\Rightarrow$" means (intuitionistic) implication so what he had discovered in coherence spaces was actually a decomposition of implication (the most important connective of logic) into simpler ones: a *linear implication* "$\multimap$" and an exponential modality "!". He understood that, in terms of sequent calculus[26], this linearization of implication corresponds to a drastic restriction on the *structural rules*, forbidding the most important ones: *weakening*[27] and *contraction*[28]. The role of the exponential modality is then to make these structural rules available again, but now as explicit *logical rules* associated with this new unary connective (and its dual), and no more as *implicit* structural rules as in intuitionistic or classical logic.

The effects of this refinement of proof theory were too many to be listed here, let us mention only those which are relevant to the present research summary.

- Conjunction and disjunction are now available in two versions: the *multiplicative* connectives $\otimes$ (tensor) and $\parr$ (co-tensor) and the *additive* connectives $\&$ (direct product) and $\oplus$ (direct sum) that we have already mentioned in coherence space. They differ logically by the way the context of "hypotheses" is handled in their introduction rules[29].

- There is a *linear negation* $A^\bot$ corresponding to the linear duality mentioned above on coherence spaces and which, just like classical negation, is involutive, so that for instance the formulas $A \multimap B$ and $B^\bot \multimap A^\bot$ are equivalent (where $A \multimap B = A^\bot \parr B$ is linear implication, to be compared with the "$A \Rightarrow B = \neg A \vee B$" of classical logic).

- There are 4 new rules associated with the exponential connectives "!" and its dual "?": weakening, contraction, *dereliction* (whose intuitive meaning is clear: if we consider proofs of $A \multimap B$ as linear proofs of $A \Rightarrow B$ and $!A \multimap B$ as "general proofs" of this implication, it expresses that the first class of proofs is contained in the second one, in other terms, it allows to forget linearity, whence the name) and *promotion* which allows to make a proof arbitrarily duplicable and discardable. It is certainly the most complicated rule of LL, being, in some sense, the only one which deals with infinity. Notice that promotion is the only rule of LL for introducing a "!", we shall come back to this point in 1.7.

- It is known that classical propositional logic, presented as a sequent calculus, has a cut elimination property: there is a terminating rewriting system on classical proofs for eliminating cuts. But this rewriting is far from being confluent, quite the contrary: for any two proofs of a formula A, there is a third one which reduces to each of these two proofs! One fundamental observation of Girard is that this is not really due to the involutivity of classical negation, but rather to the free availability of structural rules in Gentzen classical sequent calculus. And indeed, in spite

[23] With webs playing the role of bases.

[24] Contrarily to what we are used to with finite dimensional vector spaces, the co-tensor does not coincide with the tensor, and neither does direct sums coincide with direct products. This distinction tensor/co-tensor and sum/product appears in linear algebra when one moves from finite dimensional vector spaces to infinite dimensional topological vector spaces, so it is not so familiar...

[25] He probably didn't know that it was a comonad, but all the comonadic features of "!" were clear to him as we shall see.

[26] A particularly logically convenient presentation of proofs due to Gentzen.

[27] Which allows not to use an hypothesis in a proof.

[28] Which allows to use an hypothesis several times.

[29] The main feature of the sequent calculus is that it has only introduction rules, apart the *cut rule*, which can be eliminated by cut elimination (Gentzen's Hauptsatz). The cut rule is the sequent calculus version of *modus ponens*.

of its involutive linear negation, the LL sequent calculus enjoys normalization, confluence, has a denotational semantics invariant by cut-elimination (in coherence spaces for instance), all these properties meaning that contrarily to classical logic[30], and like in intuitionisitc logic, LL proofs are computationally meaningful.

- Thanks to its linear decomposition of logical rules, LL admits a graphical proof representation which is far more *parallel*[31] than the sequent calculus and even than the lambda-calculus (or natural deduction) of intuitionistic logic: Girard's *proof nets* and Lafont's *interaction nets*. Some rules however are rather reluctant to this "parallelization", this is the case notably of the promotion rule which requires *boxes* whose purpose is to delimitate the part of the proof which is concerned by the rule (and which will possibly be discarded or duplicated by a cut against a weakening or contraction rule).

- LL gives a clear status to a notion which was hardly visible in intuitionistic and classical logic: that of *polarization*. It introduces a notion of *negative formulas*, for which structural rules are freely available[32], and their dual, the *positive* formulas. From a categorical point of view, this distinction is rather clear: positive formulas are interpreted as objects equipped with a structure of !-coalgebra (remember that ! is a comonad, so it has an associated notion of coalgebra and of coalgebra morphisms, which define the Eilenberg-Moore category of !), and negative formulas are their linear duals. This *polarized logic* generalizes LL and provides a clear understanding of Griffin's computational interpretation of classical logic, of Parigot's lambda-mu calculus etc.

Plan for future research:

- I am participating in a project of Handbook on Linear Logic written collectively within the PRN (former GDRI) Linear Logic. Of course I am more involved in chapters concerning denotational semantics.

1.4.4 Hypercoherence spaces (aka. hypercoherences): a new model of Linear Logic. After this intermezzo, let us come back to sequentiality. A hypercoherence space X is a set $|X|$ (the web) together with a subset $\Gamma(X)$ of non-empty elements of $\mathcal{P}_{\mathrm{fin}}(|X|)$ (the set of all finite subsets of $|X|$) whose elements are said to be *coherent* and which contains all singletons. There are no further requirements, in particular: no bound on the cardinality of the elements of $\Gamma(X)$ and, quite unintuitively, *it is not assumed* that any non-empty subset of a coherent set (an element of $\Gamma(X)$) is coherent (belongs to $\Gamma(X)$). Then a *clique* of X is a subset of $|X|$ whose all finite and non-empty subsets are coherent: this defines a qD structure on $|X|$, which is not a coherence space in general. On these cliques, $\Gamma(X)$ allows also to define a notion of "coherence" (in the sense of qDC's) and hence a notion of strongly stable functions between hypercoherence spaces. The main property of this constructions is that this category of hypercoherence spaces and strongly stable functions is cartesian closed and is actually a model of PCF.

Cherry on the cake, I observed that this model of PCF arises from an underlying model of classical linear logic[33] whose objects are hypercoherence spaces and morphisms are special strongly stable functions – of course called linear –, exactly as the category of coherence spaces arises from the model of classical linear logic of coherence spaces and linear maps. All connectives of LL have in this model an interpretation similar to their interpretation in coherence spaces, with several striking differences, especially for the & connective and for the exponentials[34]. For instance the linear dual of a hypercoherence space X is obtained as follows: keep the same web as that of X and decree that a

[30] Until the already mentioned discovery by Tim Griffin that classical proofs can be given a computational content; from the viewpoint of LL, this amounts to defining an encoding of classical logic within LL and there are basically two possible ones corresponding to two confluent restrictions of Gentzen cut elimination.

[31] In the sense that, whenever this is possible, the redexes which can be reduced in a proof are available immediately and simultaneously, without having to perform artificial computations such as the boring sequent calculus' "commutative cuts".

[32] From a *proof search* viewpoint, these are also the reversible formulas.

[33] Technically, it is the Kleisli category of a "!" resource modality which is a comonad on the linear category.

[34] Which is not a surprise since they are the main building blocks of the types $\iota^k \Rightarrow \iota$ where sequentiality arises.

finite set is coherent in $X^\perp$ if it is finite, non-empty and either a singleton or not coherent in X. It was particularly impressive to see that such a simple hypergraphical construction was somehow hidden at the heart of our complicated constructions of qDC's and, in some sense, of sequential algorithms.

So this hypercoherence space model of linear logic was a complete surprise which convinced me that linear logic (discovered 6 years earlier by Girard, see 1.4.3) is an essential structure in computational phenomena; since then, it has been one of my main methodological guidelines.

1.4.5 Relating strong stability and sequential algorithms. This drastic simplification of strong stability was also the key to the proof of the expected theorems relating sequential algorithms and strongly stable functions. I could prove two main results.

First I could build a cartesian closed category (a model of PCF) whose objects are made of a sequential structure (see 1.4.1), a hypercoherence space, and a projection from the former to the latter which has a crucial *lifting property*. This allowed to provide explicitly a "forgetting functor" from sequential algorithms — actually not from the original category of sequential algorithms, but from a subcategory of "extensional" sequential algorithms because the projection cannot be defined on sequential algorithms such as the one of Footnote 19 — to strongly stable functions, see [Ehr96].

To overcome the restriction to "extensional sequential algorithms" of the above result, I proved that the PCF model of hypercoherence spaces and strongly stable functions is the *extensional collapse* of the sequential algorithm model[35]. The key step in this proof is the following crucial property: for any type σ and any finite clique x in the hypercoherence space interpretation of σ, it is possible to find an integer k, a clique y in the hypercoherence space associated with $(\iota^k \Rightarrow \iota) \Rightarrow \iota$ and a PCF term M of type $((\iota^k \Rightarrow \iota) \Rightarrow \iota) \Rightarrow \sigma$ such that x is equal to the application of M to y (in the model). This "relative definability result" was *per se* quite interesting and its proof crucially used the specific properties of hypercoherence spaces. One obtains the extensional collapse result by observing that sequential algorithms can easily be associated with strongly stable functions of type $(\iota^k \Rightarrow \iota) \Rightarrow \iota$ (plus of course some technicalities). These results are presented in [Ehr99].

1.4.6 Another characterization of strong stability. In the meantime, we started to study another possible definition of higher type sequentiality, with my colleague Loïc Colson. He suggested that, since sequentiality is fundamentally a concept concerning functions of "type 1", that is, of type $\iota^k \Rightarrow \iota$ (for all $k \in \mathbb{N}$), it would be a good idea to define it by induction on types by "almost closed reducibility" as follows: the idea is to define, by induction on σ, what is, for all k, a "sequential function" of type $\iota^k \Rightarrow \sigma$. For $\sigma = \iota$, we already know the answer. For $\sigma = (\tau \Rightarrow \varphi)$, we have to explain what is a sequential function f of type $\iota^k \times \tau \Rightarrow \varphi$. By inductive hypothesis we know what it means to be sequential of type $\iota^l \Rightarrow \tau$ and $\iota^l \Rightarrow \varphi$ for all $l \in \mathbb{N}$. So we can say that f is sequential if, for all $l \in \mathbb{N}$ and all h sequential of type $\iota^l \Rightarrow \tau$, the function $g : \iota^k \times \iota^l \Rightarrow \varphi$ which maps (x, y) to $f(x, h(y))$ is sequential of type $\iota^{k+l} \Rightarrow \varphi$ – this is a minimal requirement because we need our notion of sequentiality to be compositional since we want it to give rise to a denotational model.

Then we could prove that (up to some technicalities related to the fact that we had also to account for Scott continuity) the functions which are sequential in that sense coincide with the strongly stable functions on hypercoherence spaces [CE94].

So strongly stable functions can be characterized in two completely different ways (extensional collapse of sequential algorithms and "heriditarily sequential" functions), apart from strong stability itself, which is certainly one of the most striking properties of this notion.

1.4.7 Trying to recover games from hypercoherence spaces. My last investigations on this topic consisted in trying to express game semantics in terms of hypercoherence spaces. The basic idea stemmed from the observation that the so-called *co-graphs*, which can also be described as coherence spaces with a specific property [36], can be seen as trees whose leaves are the points of the web (that is, vertices of the co-graph). The nodes of such trees are of two alternating kinds which are interchanged by

[35] Actually when writing [Ehr96] I didn't know what an extensional collapse was, I must thank Samson Abramsky to have explained it to me and suggested that [Ehr96] might be an evidence that there is something interesting to say about the extensional collapse of sequential algorithms.

[36] Technically: they have no P_4 as induced sub-graph.

the linear duality of coherence spaces, just as the roles of *Player* and *Opponent* in game denotational models are interchanged by linear duality. So why not use co-graphs for defining a game-theoretic interpretation of linear logic, PCF etc, with a direct connection to hypercoherence spaces?

This idea was reinforced by a second observation: I discovered two classes of hypercoherence spaces, exchanged by linear duality, and whose intersection coincides with the class of co-graphs[37]. These are the *parallel* hypercoherence which are those such that two coherent sets which are not disjoint have a coherent union and the *serial* ones[38], defined by duality which also admit a nice direct characterization. Moreover I discovered an operation which allows to turn any hypercoherence space into a parallel one and I could prove that this *parallel enfolding* has good preservation properties, and in particular preserves seriality, so that performing this operation and then its dual turns any hypercoherence space into a co-graph, that is a game, together with a projection from this co-graph to the hypercoherence space which I still think to be an abstract version of 1.4.5.

However, due in particular to the complicated behaviour of this parallel unfolding wrt. morphism (it is not functorial though it has interesting lifting properties), I couldn't, at that time, go very far beyond these encouraging preliminary results which are summarized in [Ehr00].

Plan for future research:

- Nevertheless, I think that the idea of reformulating game models in terms of co-graphs and serial and parallel hypercoherences is an interesting one: for instance there is a natural definition for the composition of strategies (seen as cliques in co-graphs) in this setting, using parallel unfolding. During an M2 MPRI internship in 2021 I have proved, together with my student Baptiste Chanus, that this composition of strategies not only exists but is alseo unique, under mild hypotheses on the intermediate space. This is a good starting point for such a new approach to game model.
- I also would like to come back to an old idea I was not able to develop 30 years ago, namely that tools from algebraic topology might be useful for analysing sequentiality in hypercoherence spaces. Indeed the qDs associated with hypercoherence spaces are nothing but simplicial complexes which express obstruction to global sequentialization. Typically the Gustave function counterexample can be seen as a set of 3 vertices $\{1,2,3\}$ such that $\{1,2\}$, $\{2,3\}$ and $\{1,3\}$ are edges but $\{1,2,3\}$ is not a face because the 3 atomes are pairwise sequentializable but not globally. I would like to use modern homotopy tools to try analyze geometrically this kind of obstruction to seqentiality.

1.5 Indexed Linear Logic and relational semantics

On the occasion of a long visit of Antonio Bucciarelli in Marseille in 1999, we started to work on *logical relations* which are a central proof tool in the study of operational and denotational properties of programs and proofs. For instance the reducibility method used for proving normalization properties of functional languages can be understood as a logical relation technique. We focused on the denotational aspects of logical relations, and more precisely, on their linear refinement that is: what is the counterpart of logical relations in an LL denotational model?

Rather than considering arbitrary categorical models, we considered the simplest possible one, which at that time was not so popular[39]: the relational model. This model is similar to coherence spaces model, apart that it is stripped from the coherence relation: formulae are simply interpreted as sets, and proofs as relations between this sets. The main difference with the coherence space models is that, for defining $!X$, where X is a set, one takes all finite multisets of elements of X. Taking finite sets as one is first tempted to do when copying the coherence space semantics doesn't work.

We understood however that logical relations on the logical model of LL would lead to too poor a setting for being able to prove interesting results and we managed to give more "flesh" to this interpretation by parameterizing it with a given (though arbitrary) commutative monoid P together with

[37] Up to a natural embedding of coherence spaces into hypercoherence spaces.

[38] This terminology is motivated by the fact that co-graphs are sometimes called *serial-parallel graphs* due to the fact that finite cographs are generated from singletons by the two basic operations of graph disjoint union (the one where no pairs of vertices of the two graphs are related, and the one where all pairs of vertices from the two graphs are related).

[39] I don't really know who should be credited for this model, probably Girard again because this semantics underlies in some sense his *quantitative* interpretation of the lambda-calculus [Gir88].

a subset $\perp$ of P^I (where I is an infinite and countable set of indices) satisfying some natural homogeneity/symmetry properties. Very roughly, this construction follows ideas similar to those developed earlier by Girard in the realm of coherence spaces [Gir99] where the model is parameterized with a commutative comonoid in the category of coherence spaces. But the main novelty of our construction is that it gave rise to a new class of denotational models of LL that we could unexpectedly relate with the standard truth value of LL: *phase semantics* [Gir87]. Moreover, we discovered a new class of coherence spaces, quite different from Girard's original ones.

1.5.1 From denotational semantics to phase semantics. A *phase model* of LL consists of a commutative monoid M together with a subset of M (usually denoted as $\perp$)[40]. Given $F \subseteq M$, one define $F^\perp$ as the set of all $m' \in M$ such that, for all $m \in F$, one has $m\,m' \in \perp$. Then F is a *fact* is $F = F^{\perp\perp}$ and all LL formulae are interpreted as facts: these are the "truth values" of the phase model $(M, \perp)$. A fact is true if it contains the unit of M; one proves that any provable formula is interpreted as a true fact (and there is also a completeness theorem). This kind of definition by "biorthogonal closure" is essential in my work as we shall see.

Without going into technicalities, what we observed is that our new notion of "clique", parametrized now by the pair $(P^I, \perp)$, boils down to a notion of true fact in the phase space $(P^I, \perp)$, if we consider cliques not as a *set* of points of the webs, but as *families* of points indexed by subsets of I: there are infinitely many families enumerating a given non-empty set, all these families must correspond to true facts for the set to be accepted as a clique. This strongly suggested that there was a logic hidden behind these families of points of webs, and this turned out to be true: we designed a new logical system, *indexed linear logic*. With a formula A of indexed linear logic one can associate an underlying formula $\underline{A}$ of (ordinary) linear logic as well, a subset J of I (the domain of A) and a J-indexed family α of points of the interpretation of $\underline{A}$ in the relational model (which is similar to the web of a coherence space). Then this formula of indexed LL can be given a phase semantical interpretation in the phase model $(P^I, \perp)$ which coincides with the fact associated with α by the denotational model. We have designed a very natural sequent calculus for indexed LL such that the provability of a formula implies that the associated fact is true, transforming a *definability* problem into a *provability* problem.

1.5.2 Non uniform coherence spaces. We also studied concrete instances of this new class of models. One of the simplest was based on a 3 elements monoid P and a quite simple $\perp$. We could prove that the objects of the corresponding model can be seen as "coherence space" that is, as set (an object of the relational model) together with two binary symmetric relations which have an empty intersection, to be understood as "strict coherence" and "strict incoherence". The big difference with Girard's coherence spaces is that a point of the web can now be strictly coherent or strictly incoherent with itself. This may seem weird but it is actually absolutely necessary because, when building the web of $!X$ we take *all* finite multisets of elements of the web of X. In particular, contrarily to what happens in Girard's coherent spaces, we also take multisets consisting of incoherent points. If we remember that $!X$ is a crucial ingredient of the function type $X \Rightarrow Y = !X \multimap Y$, this means that the interpretation of a function makes no assumption on the uniformity of the behaviour of its argument whence the term *non-uniform semantics* for this kind of model. A simple example helps understanding the situation.

Consider the following function of type $\iota \Rightarrow \iota$:

$$\lambda x^\iota\, \mathsf{if}(x, \mathsf{if}(x, 2, 0), \mathsf{if}(x, 1, 2))$$

where our conditional is a test to 0 (if the value of the first argument is 0 we take the first branch of the conditional and if it is not 0 we take the second one). When applied to an integer this term can only return 2. But if we were in a non-deterministic world were the argument can "change its mind" between its various usages then we could get the tesults 0 or 1. This possibility is taken into account by the relational model where the interpretation of this terms contains the pair $([n, n], 2)$ for all n, but also all pairs $([0, n+1], 0)$ and $([0, n+1], 1)$ – these pairs do not appear in Girard's coherence space interpretation of this term – where $[a_1, \ldots, a_l]$ is the finite multisets[41] which contains the elements

[40] Actually this notion is suitable for multiplicative and additive LL, exponentials need an additional structure.

[41] Multisets are commutative structures, so that the order in which the argument is tested is not taken into account by the model contrarily to what happens in game semantics.

$a_1, \ldots, a_l$. Since 0 and 1 are incoherent in the interpretation of ι, we need $[0, n+1]$ to be incoherent with itself, due to the definition of coherence in $\multimap$.

These results are presented in [BE00, BE01] and there is also a completeness theorem for the indexed linear logic sequent calculus wrt. the "indexed phase semantics" described above, which is presented in [Ehr04].

More recently, I have applied these ideas to the lambda-calculus, introducing an indexed typing system in which implication is decorated by an indexing function. This system is deeply related with non-idempotent intersection types and provides a logical presentation of these systems[42], these results are presented in [Ehr20c].

Plan for future research:

- With Flavien Breuvert and Federico Olimpieri (LIPN, Université Paris Nord) we have started in 2021 investigating categorical generalizations of the idea of indexed LL to a setting based on *profunctors*, a higher dimensional generalization of the relational semantics of LL. This is part of a larger program, quite active at an international level: *categorification of semantics*. The EPSRC project *MOBILLO* (led by Nicola Gambino at Leeds), of which I'm a project partner, is part of this endeavour and Federico Olimpieri as well as my PhD student Zeinab Galal will start post-docs in Leeds in 2021 in this line of research.

1.6 Quantitative models

The parameterization of the relational model with coefficients taken in a commutative monoid as in 1.5 led me to the idea that, if we consider the objects of the relational models as *chosen bases* of vector spaces[43], then it would be quite natural to associate with each of these sets I a collection of I-indexed families of scalars (taken is some fixed field) closed under algebraic operation so as to define a vector space. However since the objects of the relational model tend to be infinite sets (actually, due to the fact that $!I$ is the collection of all finite multisets of elements of I, this set is infinite as soon as X is non-empty), we need to consider infinite-dimensional vector spaces and it is well known that such spaces must be equipped with a well behaved topology, and linear maps to have some continuity properties, if we want the resulting category to be well-behaved, and in particular if we expect *reflexivity* (the infinite dimensional generalization of the finite dimensional iso $E \simeq E^{**}$ which is the semantic account of LL negation's involutivity).

1.6.1 Köthe spaces. Rather than trying to axiomatize a well suited class of topological vector spaces which would enjoy reflexivity[44], I started from the following idea: taking $\mathbb{R}$ as field, the vectors x of the spaces E and the vectors x' of E' (its topological dual) should be elements x of $\mathbb{R}^I$ and the application of x' (considered as a linear form) to x (considered as a vector) should be given by

$$\langle x, x' \rangle = \sum_{a \in I} x_a \, x'_a \, .$$

However I being usually infinite (but countable), this sum does not make sense in general and since I has no canonical order relation, the only way to give a meaning to it is to assume that it is absolutely convergent, that is that we have $\sum_{a \in I} |x_a \, x'_a| < +\infty$. Given $\mathcal{X} \subseteq \mathbb{R}^I$, it is then natural to set $\mathcal{X}^\perp = \{x' \in \mathbb{R}^I \mid \forall x \in \mathcal{X} \ \sum_{a \in I} |x_a \, x'_a| < +\infty\}$; observe that this set is a vector space (for algebraic operations defined pointwise). Then an object of our model is a pair $(|X|, E_X)$ where $E_X \subseteq \mathbb{R}^{|X|}$ such that $E_X = E_X^{\perp\perp}$. Automatically E_X is a $\mathbb{R}$-vector space, and $X^\perp = (|X|, E_X^\perp)$ is also an object of the model.

Moreover, for each $x' \in E_X^\perp$, one can define the map $p : E_X \to \mathbb{R}_{\geq 0}$ by $p(x) = \sum_{a \in I} |x_a \, x'_a|$ which is a semi-norm so that E_X is a locally convex vector space (lcvs) whose topology is generated by all

[42] The "intersection rule" of intersection types is not a logical rule in the usual sense because it requires that the two premises are obtained by proofs having *the same* underlying λ-term.

[43] Indeed, the operations associated with the various connectives of LL on the objects of the relational model coincide with the operations on bases of vector spaces associated with the corresponding operations on vector spaces.

[44] A goal which is very difficult to reach actually.

the unit balls of these semi-norms, and it is easy to prove that it is Cauchy complete. I realized that this construction was already known in the literature on lcvs's under the name "perfect Köthe sequence space", that's why I called my objects $X = (|X|, E_X)$ Köthe spaces.

Then I observed that this construction is compatible with the interpretation of LL connectives in the relational model: for example, given two Köthe spaces X and Y, one can build a new Köthe space $X \multimap Y$ whose web is $|X| \times |Y|$ and such that $t \in E_{X\multimap Y}$ iff, for all $x \in E_X$ and each $y' \in E_{Y^\perp}$ one has $\sum_{a\in|X|,b\in|Y|} |t_{a,b}x_a y'_b| < +\infty$. Then the map $x \mapsto t\,x = \left(\sum_{a\in|X|} t_{a,b}x_a\right)_{b\in|Y|}$ is a linear and continuous function $E_X \to E_Y$ and conversely any linear continuous function $E_X \to E_Y$ can be represented by a unique such (potentially infinite dimensional) matrix $t \in E_{X\multimap Y}$. As any Köthe space $E_{X\multimap Y}$ is equipped with the topology explained above, and this topology can be characterized intrinsically (uniform convergence on a class of "bounded" sets). Similarly one defines a tensor product, a direct product, a direct sum. Also, for all Köthe space X one can define an exponential $!X$. Then the "matrices" in $E_{!X\multimap Y}$ can be understood as analytic functions $E_X \to E_Y$. So I have built in that way a cartesian closed category (that is, a model of the simply typed lambda-calculus) of Köthe spaces and analytic functions.

These analytic functions can be differentiated, that is, for any analytic $f : E_X \to E_Y$ it is possible to define a new analytic function $f' : E_X \to E_{X\multimap Y}$, the derivative (aka. differential, aka. Jacobian etc) of f which is characterized by the fact that $f(x)+f'(x)\,u$ is the best possible approximation of $f(x)$ in some sense (remember that $f'(x)\,u$ is the application of the continuous linear function $f'(x) \in E_{X\multimap Y}$ to u). This suggests to extend the lambda-calculus, and more generally LL, with a differentiation operation and to study its operational meaning, see 1.7. This model is presented in [Ehr02].

1.6.2 Finiteness spaces. One of the main features of Girard's coherence space model of LL is that the intersection of a clique and an anti-clique has at most one element, which is a denotational account of the finiteness and determinism of computation. This property is lost in the relational model, and also in the non-uiniform coherence space model 1.5.2. I tried to prove that one could nevertheless keep this intersection finite, as a witness of the finiteness of computations compatible with finite non-determinism[45]. For this I introduced the notion of *finiteness space*, based on an idea similar to that of phase semantics, Köthe spaces etc.: given a set I and a subset $\mathcal{F}$ of $\mathcal{P}(I)$, define $\mathcal{F}^\perp = \{u' \subseteq u \mid u \cap u' \text{ finite}\}$. Then a finiteness space is a pair $X = (|X|, \mathsf{F}(X))$ where $\mathsf{F}(X) \subseteq \mathcal{P}(|X|)$ satisfies $\mathsf{F}(X) = \mathsf{F}(X)^{\perp\perp}$; the elements of $\mathsf{F}(X)$ are called *finitary*. The dual is defined as usual: $X^\perp = (|X|, \mathsf{F}(X)^\perp)$.

This again gives rise to a model of LL, compatible with the relational model. It's most interesting feature is that, for any field $\mathbb{K}$ it induces a model of LL whose objects are $\mathbb{K}$-vector spaces which are actually "topological vector spaces" in a sense which is not the most standard one (as far as I know this notion of tvs was first introduced by Lefschetz in [Lef42]). Indeed, given a finiteness space X we can define a $\mathbb{K}$-vector space $\mathbb{K}\langle X\rangle$ by taking the elements x of $\mathbb{K}^{|X|}$ such that the *domain* of x (the set of all $a \in |X|$ such that $x_a \neq 0$) belongs to $\mathsf{F}(X)$. The fact that $\mathbb{K}\langle X\rangle$ is a vector space results from the closeness of $\mathsf{F}(X)$ under finite unions which itself results from the assumption $\mathsf{F}(X) = \mathsf{F}(X)^{\perp\perp}$ (it is a virtue of this kind of definition to have many nice structural consequences like this one). Now we can use $X^\perp$ to equip $\mathbb{K}\langle X\rangle$ with a topology: given $u' \in \mathsf{F}(X^\perp)$ the set of all $x \in \mathbb{K}\langle X\rangle$ such that $x_a = 0$ for all $a \in u'$ is a linear subspace of $\mathbb{K}\langle X\rangle$ and these subspaces (for all possible u') form a sub-basis of this topology at 0 (and hence everywhere since the topology must be invariant by translation)[46]. This topology on $\mathbb{K}\langle X\rangle$ is Hausdorff and Cauchy complete[47].

The main idea of this construction is that, if $x \in \mathbb{K}\langle X\rangle$ and $x' \in \mathbb{K}\langle X^\perp\rangle$, the sum $\langle x, x'\rangle = \sum_{a\in|X|} x_a x'_a$ has only finitely many non 0 terms since there are only finitely many $a \in |X|$ such

[45] That is, non-deterministic computations where the non-deterministic branchings are finite.

[46] So all our basic neighborhoods of 0 are linear subspaces of $\mathbb{K}\langle X\rangle$ which is quite different from what one is used to in Banach spaces and more generally in locally convex spaces. For instance it is easy to prove that all these subspaces are open and closed, so that the topology is 0-dimensional. Also the field itself, considered as a 1-dimensional space, as well as all finite dimensional spaces of this kind, are equipped with the discrete topology. Of course things become interesting with infinite dimensional spaces.

[47] More precisely, $\mathbb{K}\langle X\rangle$ is not only a topological space, but also a uniform space and is Cauchy complete as such even if it is not always a metric space.

that $x_a \neq 0$ and $x'_a \neq 0$, even if $|X|$ is an infinite set. Then it is possible to prove that $\mathbb{K}\langle X^\perp\rangle$ is linearly isomorphic to the topological dual of $\mathbb{K}\langle X\rangle$ (the space of continuous linear forms on $\mathbb{K}\langle X\rangle$) and it is possible to characterize the topology on this topological dual which makes this linear iso an homeomorphism: it is the topology of uniform convergence on *bounded*[48] linear subspaces.

All the finiteness structures associated with the finiteness space interpretation of the various LL connectives have similar algebraic/topological interpretations. For instance $\mathbb{K}\langle X \multimap Y\rangle$ is linearly homeomorphic to the space of all linear and continuous functions $\mathbb{K}\langle X\rangle \to \mathbb{K}\langle Y\rangle$ equipped with the topology of uniform convergence on bounded linear subspaces. Similarly $!X$ is defined in such a way that $\mathbb{K}\langle !X \multimap Y\rangle$ can be seen as a space of analytic functions $\mathbb{K}\langle X\rangle \to \mathbb{K}\langle Y\rangle$. When X and Y are finite dimensional (that is, when $|X|$ and $|Y|$ are finite sets), such "analytic" functions are all polynomial functions. But when these sets become infinite (and the corresponding finiteness structures non trivial), non polynomial analytic functions arise so that one can probably understand finiteness spaces as a infinite dimensional generalization of polynomials.

As in Köthe spaces 1.6.1 these analytic morphisms can be differentiated; this operation generalizes the formal differentiation of polynomials. These results are presented in [Ehr05, Ehr18] and urged me further to develop a differential extension of lambda-calculus and LL.

Plan for future research:

- I would like to generalize these constructs to more general Lefschetz spaces than those induced by finiteness spaces, and in particular to understand when a Lefschetz space is reflexive. Indeed, it is not clear a priori that the Lefschetz spaces induced by finiteness spaces have all small limits (and more specifically, equalizers) which seem crucial for interpreting dependent types, whereas general Lefschetz spaces clearly have. So they might provide a suitable mathematical setting where it would possible to combine Linear Logic and Dependent Types, a task that I consider as one of the most important in semantics. We have regular discussions with Christine Tasson on this topic.

1.6.3 The relational model and its extensional collapse. When I moved to PPS (Paris) from IML (Marseille) in 2005 I started to work with Antonio Bucciarelli and Giulio Manzonetto on models of the pure lambda-calculus, and more specifically on models based on the relational semantics of LL. We studied a relational analogue of Scott's D_∞ construction and proved some basic constructions thereof [BEM07] and also the semantics of a non-deterministic extension of the lambda-calculus in this model in [BEM12]. This kind of model always satisfy a property called *sensibility* which expresses that any "looping" lambda-term has an empty intersection.

Later on, with Antonio Bucciarelli, Alberto Carraro and Antonino Salibra, we have explored a modification of the interpretation of the "!" modality in the relational model allowing to build non sensible models of the pure lambda-calculus. A multiset of elements of a set I is a function $I \to \mathbb{N}$, and remember that in the relational model, $!X$ is the set of finite multisets of elements of the set X. So we have replaced $\mathbb{N}$ with more general semi-rings containing possibly "infinite" values. We have studied which conditions should satisfy such a semi-ring in order to obtain an exponential which satisfies the requirements of LL (that is, which give rise to a model of LL), arriving to a rather general notion[49] of *multiplicity semi-rings* of which we have found several instances, and show that they give rise to a non-sensible relational interpretation of the pure lambda-calculus, see [ECS10].

The Kleisli category of the usual exponential (based on finite multiplicities) is not well pointed: for instance the two morphisms (relations) $\{([a], b)\}$ and $\{([a, a], b)\}$ from X to Y in this Kleisli category (where $a \in X$ and $b \in Y$) act in the same way on subsets x of X: such a set is mapped to $\{b\}$ if $a \in x$ and to $\emptyset$ otherwise. I have conjectured and then proved that the "extensional collapse" of this model is the Scott model of LL.

Indeed, it was known since the 1990's (by several people: Huth, Krivine, Winskel in particular) that the Scott semantics arises as the Klesili category of a model of LL, very similar to the relational

[48] This notion can be defined intrinsically.

[49] Our conditions have been further generalized by Flavien Breuvart, but an intersting problem remains, which is to understand what is the most general axiomatization of such a structure.

model. The objects of the LL Scott model are preordered sets $S = (|S|, \leq_S)$ (a web plus a transitive and reflexive relation on it). With any such preorder one can associate a (prime algebraic) complete lattice $\mathcal{I}(S)$ whose elements are the downwards-closed subsets of $|S|$, the order relation being $\subseteq$, and it is quite simple to prove that any prime-algebraic complete lattice can be represented in that way. If we take as morphisms from S to T the *linear* functions $\mathcal{I}(S) \to \mathcal{I}(T)$, that is the functions which commute with all unions, which are in bijective correspondence with the initial segments of $S \multimap T = S^{\mathsf{op}} \times T$ (the preorder whose web is $|S| \times |T|$ and $(a,b) \leq_{S\multimap T} (a',b')$ if $a' \leq_S a$ and $b \leq_T b'$). The linear negation of S is simply S^{op} (same web but reversed preorder relation) so it is clearly involutive. The other connectives of LL have interpretations which are the same as in the relational model, as far as the webs are concerned. In particular $!S$ has web $\mathcal{M}_{\mathrm{fin}}(|S|)$, the set of finite multisets of elements of $|S|$, and $m \leq_{!S} m'$ if for all $a \in m$ (meaning that a appears at least once in m) there is $a' \in m'$ such that $a \leq_S s'$. Then it is also easy to exhibit an isomorphism between the lattice of all Scott continuous functions $\mathcal{I}(S) \to \mathcal{I}(T)$ (ordered pointwise) and $\mathcal{I}(!S \multimap T)$.

Using the fact that the web of the interpretation of an LL formula in this Scott model coincides with the interpretation of this formula in the relational model, I have reduced the study of this extensional collapse to the introduction of a new model of LL which is based on a hitherto unknown duality on preorders: given a preorder S and a subset A of $|S|$, we define a subset $A^\perp$ of $|S^\perp| = |S|$ by $A^\perp = \{u' \subseteq |S| \mid \forall u \in A\ u' \cap {\downarrow} u \neq \emptyset \Rightarrow u' \cap u \neq \emptyset\}$ (where ${\downarrow} u$ is the downwards closure of u in S), in other words $u' \in A^\perp$ if u' cannot separate u from its downwards closure, for all $u \in A$. It is essential to observe that this duality is relative to the preorder structure of S, so one should write $A^{\perp(S)}$ rather that $A^\perp$.

An object of this model is a tuple $X = (\underline{X}, \mathsf{D}(X))$ where $\underline{X}$ is a preorder and $\mathsf{D}(X) \subseteq \mathcal{P}(|\underline{X}|)$ satisfies $\mathsf{D}(X)^{\perp\perp} = \mathsf{D}(X)$ (notice that $\mathsf{D}(X)^\perp$ is relative to the preorder $\underline{X}^\perp$ so that the second "$\perp$" is taken relative to the preorder $\underline{X}^\perp = \underline{X}^{\mathsf{op}}$). There is also a notion of morphisms between these objects (a morphism from X to Y is a subset of $|\underline{X}| \times |\underline{Y}|$ satisfying some conditions relative to $\mathsf{D}(X)$ and $\mathsf{D}(Y)$. Again, the corresponding category is a model of LL: one defines of course $X^\perp = (\underline{X}^{\mathsf{op}}, \mathsf{D}(X)^\perp)$ and then $X \otimes Y$, $X \parr Y$ etc and the exponential $!X$. Of course $X \multimap Y = X^\perp \parr Y$ and one can prove that the morphisms $X \to Y$ are exactly the elements of $\mathsf{D}(X \multimap Y)$.

Notice that any object of this model provides an object of the relational model (namely $|\underline{X}|$) and of the Scott model of LL (namely $\underline{X}$) and these two mappings commute with the interpretation of LL connectives in the model. Moreover it provides a partial equivalence relation [50] $\sim_X$ on $\mathcal{P}(|\underline{X}|)$ (points in the relational model): x is equivalent to y if $x, y \in \mathsf{D}(X)$ and ${\downarrow} x = {\downarrow} y$ (these downwards closure being of course taken in the preorder $\underline{X}$) as well as a bijective correspondence between the equivalence classes of $\sim_X$ and the elements of $\mathcal{I}(X)$. My main result on this topic shows that this partial relation "is" the extensional collapse partial equivalence relation of the relational model and therefore that this extensional collapse is isomorphic to the Scott model. Of course giving a precise meaning to these rough statements requires some care. These constructions and results are presented in [Ehr09].

In that way we have exhibited a deep connection between the relational model and the Scott model of LL and revealed its underlying structure as a new duality on preorders. The conceptual benefit of this connection is that the relational model, though infinitary (in the sense that the interpretation of functional types are infinite, due to the use of multisets in the interpretation of "!"), is quite simple and provides in some sense rather explicit information about the syntax of LL — for instance, it has been recently proven that the mapping from normal proof-nets LL to the relational model given by the denotational semantics is *injective* — whereas the Scott model is much less explicit (a lot of information on proofs/programs are lost in the semantics) but is finitary: if we interpret ground types with finite preorders, then the interpretations of all types are finite[51].

1.7 Differential extensions

In the quantitative models 1.6.1 and 1.6.2 we have seen that the morphisms of the associated Kleisli cartesian closed category are analytic functions which can be differentiated. This means that with

[50] A symmetric and transitive, but not necessarily reflexive, relation.

[51] In our presentation of $!S$ in the Scott model we have used finite multisets in order to simplify this connection but one can use finite sets instead without changing the model, which is impossible to do in the relational model.

any morphism $!X \multimap Y$ we can associate its differential of type $!X \multimap (X \multimap Y)$, or equivalently $!X \otimes X \multimap Y$.

1.7.1 The differential lambda-calculus. I have observed that this operation could be transferred to the syntax of the lambda-calculus and we started with Laurent Regnier to study a simply typed *differential lambda-calculus*. The basic idea is to keep the usual types of the lambda-calculus and to introduce a new syntactic construction of "differential application" follows:

$$\frac{\Gamma \vdash M : A \Rightarrow B \qquad \Gamma \vdash N : A}{\Gamma \vdash \mathsf{D}M \cdot N : A \Rightarrow B}$$

where N is the linear parameter of the differential, whereas the a linear parameter (of type A) can be given as argument to $\mathsf{D}M \cdot N$ by an ordinary lambda-calculus application $(\mathsf{D}M \cdot N)\, P$.

One nice feature of this way of presenting things is that it makes iteration of this differential application quite easy: with the same typing assumption on M but with now k terms $N_1, \ldots, N_k$ such that $\Gamma \vdash N_i : A$ for all $i = 1, \ldots, k$, we can define $\mathsf{D}^k M \cdot (N_1, \ldots, N_k)$, for instance $\mathsf{D}^2 M \cdot (N_1, N_2) = \mathsf{D}(\mathsf{D}M \cdot N_1) \cdot N_2$ is a second derivative.

With this new "differential application" construct is associates a new differential β-reduction: given M such that $\Gamma, x : A \vdash M : B$ and N such that $\Gamma \vdash N : A$ the term $\mathsf{D}(\lambda x^A\, M) \cdot N$ reduces to $\frac{\partial M}{\partial x} \cdot N$ exactly as $\left(\lambda x^A\, M\right) N$ reduces to $M\,[N/x]$ (M where N has been substituted for x), where $\frac{\partial M}{\partial x} \cdot N$ can be understood as a "linear substitution", that is a substitution which replaces *exactly one occurrence* of x in M with N. This "substituted term" can be typed as follows

$$\Gamma, x : A \vdash \frac{\partial M}{\partial x} \cdot N : B$$

since it can still contain free variables of x. Exactly as the ordinary substitution $M\,[N/x]$, this linear, or differential, substitution $\frac{\partial M}{\partial x} \cdot N$ is defined by induction on M. The three interesting cases are when M is a variable, a differential application $\mathsf{D}P \cdot Q$ and an ordinary application $(P)\, Q$. We set

$$\begin{aligned}
\frac{\partial y}{\partial x} \cdot N &= \begin{cases} N & \text{if } y = x \\ 0 & \text{if } y \neq x \end{cases} \\
\frac{\partial \mathsf{D}P \cdot Q}{\partial x} \cdot N &= \mathsf{D}\left(\frac{\partial P}{\partial x} \cdot N\right) \cdot Q + \mathsf{D}P \cdot \left(\frac{\partial Q}{\partial x} \cdot N\right) \\
\frac{\partial (P)\, Q}{\partial x} \cdot N &= \left(\frac{\partial P}{\partial x} \cdot N\right) Q + \left(\mathsf{D}P \cdot \left(\frac{\partial Q}{\partial x} \cdot N\right)\right) Q\,.
\end{aligned}$$

This definitions calls several comments. The first equation, in the case M is the variable x is quite clear: we have exactly one occurrence of x and we substitute it with N. The case M is a variable $y \neq x$ has two interpretations. The "operational" one: there are no occurrences of x and hence the process fails (then 0 is just some kind of error). The algebraic one: we take the derivative of a constant (y is a constant wrt. x) and the result is 0.

The operational interpretation of the second equation is that we have two ways of substituting exactly one occurrence of x in $\mathsf{D}P \cdot Q$: either in P or in Q and we return the "non-deterministic" sum of these two options. The algebraic interpretation is simply Leibniz Law of Calculus, since $\mathsf{D}P \cdot Q$ is a bilinear operation in P and Q.

The interpretations of the third equation is similar though slightly more complicated because we cannot be sure *a priori* that an occurrence of x in Q (the argument of the application $(P)\, Q$) is linear in $(P)\, Q$ even if it is linear in Q: this depends on how P uses its argument. So we "force" P to make a linear use of its argument by using a differential application in the second summand. From an algebraic point of view, this is just a combination of Leibniz law and of the chain rule of Calculus.

So we have to add anew term 0 and the possibility of adding terms (endowing them with a commutative monoid structure), these constructs obey the following typing rules:

$$\frac{}{\Gamma \vdash 0 : A} \qquad \frac{\Gamma \vdash M_1 : A \qquad \Gamma \vdash M_2 : A}{\Gamma \vdash M_1 + M_2 : A}$$

It is also quite natural to extend these algebraic operations with the action of a commutative semi-ring so as to turn terms into a semi-module (or a vector space if this semi-ring is a field). We also had to specify which syntactic constructs of our syntax are linear wrt. to this monoid structure: this is very simple, all constructs are, but the right side of ordinary application: we have $(M_1 + M_2)\, N = (M_1)\, N + (M_2)\, N$, but **not** $(M)\,(N_1 + N_2) = (M)\, N_1 + (M)\, N_2$ in general.

One has also to take into account a commutation law for derivatives (the fact that the second differential, or Hessian, is a symmetric bilinear function): with the notations introduced above this boils down to $\mathsf{D}^2 M \cdot (N_1, N_2) = \mathsf{D}^2 M \cdot (N_2, N_1)$.

For all these design choices, we were guided by my vector space LL models, whose Kleisli categories are of course models of this differential lambda-calculus (with the differential operations already mentioned for interpreting differential application and substitution).

In [ER03] we studied the operational properties of this calculus: we proved confluence and strong normalization for a second order typed version of this calculus (a differential System F) under some restrictions on the semi-ring of coefficients. This kind of rewriting on terms equipped with an linear structure is quite interesting and reveals new phenomena which have been in particular studied by my former PhD student Lionel Vaux.

1.7.2 Taylor expansions of lambda-terms. In the same paper [ER03] we also studied a *Taylor expansion* of lambda-term application. We considered the following situation: two ordinary lambda-terms M and N such that $(M)\, N$ is well typed and reduces to a variable (or a constant if we have some in our language). Then we can write a Taylor expansion at 0 for this application in our differential lambda-calculus:

$$\sum_{k=0}^{\infty} \frac{1}{k!} \left(\mathsf{D}^k M \cdot (\overbrace{N, \ldots, N}^{k}) \right) 0$$

and we proved that there is exactly one of the terms of this sum which does not reduce to 0 in our reduction system described above. The corresponding integer k is the number of times N arrives in code position in the execution of $(M)\, N$ in Krivine's machine[52].

In [ER08, ER06a], we continued this line of investigation by considering now a much more general Taylor expansion of ordinary lambda-terms where *all* applications of a term are expanded. By applying standard computation laws (essentially: the multinomial expansion of expressions of the shape $(\sum_{k\in\mathbb{N}} t_k)^n$) we turn in that way any lambda-term into a linear combination of differential terms with coefficients in $\mathbb{Q}^+$ which is infinite as soon a the lambda-term contains an application. The differential terms appearing in this expansion have the specificity that all ordinary applications are applications to 0, that is of the form $(M)\, 0$.

We defined a specific syntax for these differential lambda-terms where all applications are to 0, calling them *resource lambda-terms* for crediting earlier work by Boudol and Kfoury in particular, who introduced related formalisms in their study of the lambda-calculus and of its connections with process algebra (although they had of course no differential background underlying their calculi).

The main feature of our resource lambda-calculus is that application is now of shape $\langle s \rangle\, [t_1, \ldots, t_n]$ where $[t_1, \ldots, t_n]$ is a multiset of resource terms; this represents the nth derivative of s computed at 0 and applied to the tuple $(t_1, \ldots, t_n)$; since this differential is a symmetric n-linear function, we use multisets instead of tuples for collecting arguments. In this setting β-reduction turns $\langle \lambda x\, s \rangle\, [t_1, \ldots, t_n]$ into 0 if the number of occurrences of x in s is $\neq n$ and into the sum of all terms obtained by substituting $t_1, \ldots, t_n$ to the occurrences $x_1, \ldots, x_n$ in s, up to all permutations (so it is a sum of $n!$ terms). We compared the β-reduction of the original lambda-term and the reduction of the resource terms of its Taylor expansion, proving that the computation of the Böhm tree of a term commutes with its Taylor expansion (an abstract way of expressing that Taylor expansion is compatible with β-reduction). We also related this Taylor expansion with Krivine's machine, generalizing our earlier result. For this

[52] A simple abstract machine for computing lambda-terms which implements a refinement of β-reduction called *linear head-reduction* very close to *De Bruijn mini-reduction*. The point of this reduction is that global substitutions are never performed, one only fetches in an environment the term corresponding to a variable each time this is needed.

purpose we introduced a coherence relation on resource terms (the finite cliques of which are the finite sets of resource terms which can simultaneously appear in the Taylor expansion of a lambda-term) and developed a combinatorial analysis of the resource β-reduction.

In the related paper [Ehr10] I defined a finiteness structure (see 1.6.2) on resource terms using their non-deterministic reduction relation and I related it with system F's strong normalization. This line of research has been further investigated by Pagani, Tasson and Vaux. Recently, this approach has been shown to quite effective for proving syntactic properties of the lambda-calculus by Barabarossa and Manzonetto in [BM20] which obtained a Distinguished Paper Award at POPL'20.

1.7.3 Differential LL. The categories of 1.6.1 and 1.6.2 are models of LL and so it was natural to try to understand the new differential rules as LL principles. It has been a quite pleasant surprise to observe that these rules are exactly dual to the LL rules of dereliction, weakening and contraction. This led me to a new logical system, differential linear logic (DiLL) which has exactly the same logical connectives as LL, but new rules for the exponentials that we briefly present and comment. To understand the rules, a good intuition is to see formulas as representing vector spaces.

The first of these rules is *codereliction*

$$\frac{\vdash A, \Gamma}{\vdash !A, \Gamma}$$

which allows in particular to prove $A \multimap !A$ in DiLL and thus turn any proof of $!A \multimap B$ (to be seen as a non necessarily linear function f from A to B) into a proof of $A \multimap B$ (a linear function from A to B) which should be understood as $f'(0)$. It is dual to the standard rule of *dereliction*

$$\frac{\vdash A, \Gamma}{\vdash ?A, \Gamma}$$

where "?" is the dual connective of "!", which allows to prove $!A \multimap A$ and thus to turn a linear proof of $A \multimap B$ (a linear function from A to B) into a proof of $!A \multimap B$ that is, to forget the linearity of f.

The *coweakening* rule is

$$\frac{}{\vdash !A}$$

which allows to turn a proof of $!A \multimap B$ (a function f from A to B) into a proof of B (a vector in B, namely $f(0)$). It is dual to the weakening rule

$$\frac{\vdash \Gamma}{\vdash ?A, \Gamma}$$

which allows to turn a proof of B (a vector v in B) into a proof of $!A \multimap B$ (the function from A to B which is constant of value v).

The *cocontraction* rule is

$$\frac{\vdash !A, \Gamma \qquad \vdash !A, \Delta}{\vdash !A, \Gamma, \Delta}$$

which allows to turn a proof of $!A \multimap B$ (a function f from A to B) into a proof of $!A \otimes !A \multimap B$ (a two parameter function g from $A \times A$ to B, namely $g(u, v) = f(u + v)$). It is dual to the contraction rule

$$\frac{\vdash ?A, ?A, \Gamma}{\vdash ?A, \Gamma}$$

which allows to turn a proof of $!A \otimes !A \multimap B$ (a two parameter function g from $A \times A$ to B) into a proof of $!A \multimap B$ (a function f from A to B, given by $f(u) = g(u, u)$).

There are also rules for dealing with 0 and addition

$$\frac{}{\vdash \Gamma} \qquad \frac{\vdash \Gamma \qquad \vdash \Gamma}{\vdash \Gamma}$$

which are necessary for cut elimination and turn this system into a logically peculiar one since all formulas are provable (by the 0 proof). Indeed, the point of this system is to deal with *partial* proofs, allowing for instance to write approximations of ordinary proofs by means of Taylor expansions.

The symmetry between the structural rules (dereliction, weakening, contraction) and the costructural ones is rather clear in this sequent calculus presentation, but it becomes even more striking in proof-nets or interaction nets, this is why we have adopted in my first paper on this topic with Laurent Regnier [ER06b].

In ordinary LL, as explained in 1.4.3, the purpose of the "!" connective is to make a proof freely discardable and duplicable by means of the promotion rule which is the only one which introduces the "!" connectives. In DiLL we have new rules for introducing this connective, namely coweakening and codereliction which have not the same "infinite" power as promotion, this changes the meaning of the "!" connective which can be understood now as expressing that some resources become available for communication: codereliction allows to make one copy available, coweakening, 0 copy, and cocontraction allows to add bunches of resources. The resource consumer on the other hand will use dereliction when it needs exactly one resource, weakening when it needs 0 resource and contraction when it "branches two consumers of the same type on the same resource channel"[53].

So for instance when one branches (or, in logical terms, cuts) a codereliction on a contraction we get a sum of two proofs, one where the codereliction is connected to the first consumer and the second consumer is cut on a coweakening, and dually – algebraically this is Leibniz law –. There is a completely similar reduction when one connects a dereliction and a cocontraction: the consumer which needs exactly one resource has to choose between the two providers combined by the cocontraction – algebraically this is just the fact that a linear function commutes with sums –. The sums which appear in the reduction (we already encountered them in 1.7.1) can naturally be seen as *non deterministic choices* between two possible behaviours. So DiLL can be considered as an intrinsically non deterministic logical system extending LL.

Notice also that the symmetry between structural and costructural rules extends to the reduction (cut elimination) rules. There are also cut elimination rules dealing with costructural rules and promotion, one of them is an LL version of the already mentioned chain rule.

1.7.4 DiLL and process calculi. In a joint work with Olivier Laurent [EL10b, EL10a] we have developed these operations intuitions and these non deterministic features of DiLL by encoding two process algebras in this system. The main idea of these encodings is to use a combination of contraction and cocontraction for representing parallel composition and dereliction and codereliction for representing prefixes so that the interaction between these prefixes and parallel composition leads to non-deterministic sums.

A side product of this research is the observation that this implementation of parallel composition supports more sophisticated topologies than the simple one where any actor can communicate with any other actor. Using this idea we developed with Jiang Ying (from the ISCAS, Beijing) a new process algebra which extends tree automata to an interactive setting, just like Milner's CCS extends ordinary automata to an interactive setting [EJ15, JLE19]. With Jean Krivine, we have recently developed a new presentation of this idea, in a setting more compatible with the formalization adopted in the process algebra community [EJK19]. The main feature of these new process algebras is that parallel composition can be an arbitrary graph at the vertices of which processes are located, the edges indicating which communications are possible. This graph structure evolves during execution.

1.7.5 Coherent differentiation. Motivated by observations presented in [Ehr19, Ehr21b], see 1.8.6, and by the preparation of my invited talk at the BIRS meeting *Tangent Categories and their Applications* (June 2021), I have developped in May-June 2021 a new categorical setting where the ideas of the differential λ-calculus and of DiLL can be refined and become compatible with *deterministic* and with *probabilistic* computations. In other terms, in this new setting, the addition of terms of the same type is not always possible anymore. Only some specific additions are allowed, sufficent to make available the summations required by the Leibniz rule (see Section 1.7.1). This new line of ideas is presented

[53] In lambda-calculus, this is just identifying two variables of the same type.

in [Ehr21a], accessible online.

The basic idea is to consider a category $\mathcal{L}$, to be understood as a "linear category" where each homset has a 0 element but, contrarily to the models of 1.6.1 and 1.6.2, addition of morphisms is not always well defined. Typically, $\mathcal{L}$ is a denotational model of LL, a Seely category such as the category of coherence or hypercoherence spaces and linear morphisms, or the category of probabilistic coherence spaces and linear morphisms 1.8.1. Instead of the usual requirement that $\mathcal{L}$ is additive (satisfied in the models of 1.6.1 and 1.6.2 and meaning that morphisms can freely be added), we assume that $\mathcal{L}$ is equipped with a tuple $(\mathsf{S}, \pi_0, \pi_1, \mathsf{s})$ where $\mathsf{S} : \mathcal{L} \to \mathcal{L}$ is a functor and $\pi_0, \pi_1, \mathsf{s} : \mathsf{S} \to \mathsf{Id}$ are natural transformation. Intuitively, X being an object of $\mathcal{L}$, $\mathsf{S}X$ is the object of summable pairs of elements of X, π_0 and π_1 are the obvious projections and s is the addition operation, allowed only for summable pairs. We exhibited a few simple axioms for this structure which imply in particular that addition becomes a partially defined associative and commutative operation. When these axioms are satsified, we say that $(\mathcal{L}, \mathsf{S}, \pi_0, \pi_1, \mathsf{s})$ is a *summable category*[54].

Just as in tangent categories, the functor S inherits a monad structure which is technically crucial for expressing the linearity of differentiation wrt. the partially defined addition of morphisms. What about differentiation itself? It is simply axiomatized as a natural transformation $\partial_X : !\mathsf{S}X \to \mathsf{S}!X$ (here we assume that $\mathcal{L}$ has an "exponential functor" as any model of LL) which, technically speaking, is a *distributive law* allowing to lift the functor S to a functor $\mathsf{D} : \mathcal{L}_! \to \mathcal{L}_!$ on the Kleisli category of $\mathcal{L}$, that is, the category of "non-linear morphisms" which, as we have seen in many examples now, is typically cartesian closed and provides a model of the λ-calculus. On objects, D acts just as S: $\mathsf{D}X$ is the space of summable pairs (x, u) of elements of X. On morphisms, D behaves exactly as the "tangent bundle functor" T of tangent categories, that is, a morphism $f \in \mathcal{L}_!(X, Y)$ seen as a non-linear[55] function $X \to Y$, is mapped to $\mathsf{D}f : \mathsf{S}X \to \mathsf{S}Y$ which maps the summable pair (x, u) to $(f(x), f'(x) \cdot u)$ and indeed it is reasonable to consider that $f(x) + f'(x).u$ always makes sense since it is the beginning of the Taylor expansion of $f(x + u)$ at x. Here $f'(x) \cdot u$ represents the application to u of the differential of f computed at x, which is a linear map (aka. Jacobian of f at x).

In [Ehr21a] I use Girard's coherence spaces and their non-uniform generalization 1.5.2 as a running example of non additive category where these refined differential constructs make sense. I plan to devote a full paper to the case of probabilistic coherence spaces, to which the new approach applies (actually it was the main application I had in mind when starting to develop these general ideas). In [Ehr21a] I also outline a syntax for a differential λ-calculus compatible with this new categorical setting, and thus *not featuring* the rule according to which $M_1 + M_2$ is typable of type A as soon ase M_1 and M_2 are. Based on these ideas I've designed a differential PCF featuring general recursion and fully compatible with this new setting of coherent differentiation. This is the object of a paper I'm currently writing (Aug-Sept 2021).

One of the main open questions with the differential extensions of LL and of the λ-calculus of 1.7.1 and 1.7.3 was to understand their operational meaning and a major obstacle was the essentially non-deterministic character of these formalisms. I see the introduction of *coherent differentiation* as a major step towards a convincing answer to this question since now we are able to extend standard, deterministic or probabilistic[56], functional programming languages with differential constructs without breaking their main operational properties. This does not mean that we fully understand the operational meaning of these constructs but at least we are now able to study them in a reasonably standard operational setting (typically, using abstract machines *à la* Krivine).

Plan for future research:

[54] This structure is similar to that of a *tangent category* which is a category $\mathcal{X}$ (whose object should be thought of as manifolds) equiped with a functor $T : \mathcal{X} \to \mathcal{X}$ to be thought of as the *tangent bundle* functor: intuitively, it maps a manifold X to its tangent bundle which is a manifold TX whose objects are the pairs (x, u) where x is a point of X and u a tangent vector of X at x. In that case there is also a $\pi_0 : TX \to X$ natural transformation, but our two other morphisms π_1 and s do not exist for the tangent bundle functor. Moreover, tangent vectors at a given point can freely be added, a feature that we definitely want to get rid of.

[55] But very regular, typically analytic as in probabilistic coherence spaces 1.8.1

[56] Probabilistic computations are much closer to deterministic computations than to non-deterministic ones. This particularly obvious from a denotational point of view.

- My approach to program differentiation is not specific to "ordinary differentiation" wrt. real or complex numbers (in contrast with the languages used for Automated Differentiation applied typically to gradient backpropagation algorithms in Machine Learning where the type float is crucial). On the contrary the kind of differentiation I consider is *completely generic*: for instance we can compute the derivative of a program from binary trees labeled by integers to lists of boolean or even of an higher order program, taking programs as arguments. Even if more complicated than our initial non-deterministic theory, the new *coherent differentiation* completely preserves this genericity of differentiation, turning it into a new set of constructs with which basically any functional programming language can be extended. Now the question is: what do these new constructs do actually compute? I conjecture that they might be related to *incremental computing* where differential ideas are also used, though admittedly without the strong semantical and logical background our approach relies on.

- In my paper [Ehr18] I introduced a categorical way of understanding the computation of anti-derivatives (primitives) in categorical models of DiLL: this simply boiled down to the existence of a natural morphism $I_X : !X \to !X$ with specific features. This kind of structure seems perfectly compatible with coherent differentiation, and since, in this new setting, all "Scott least fixpoint operators" $(X \Rightarrow X) \to X$ are typically available, it becomes possible to solve differential equations as least fixpoint of "integral equations" invoking now Scott insead of Lipschitz. The meaning of such programs defined as recursive functions whose definition involves the I_X "primitive" construct is of course very intriguing!

- The coherent differentiation setting is compatible with classical LL (all the models of coherent differentiation we know are also models of LL) but it is an exciting open problem to find elementary logical constructs allowing to represent its basic operations. Indeed, the coherent differential PCF we are currently developping is essentially a syntactification of the basic categorical operations used for defining models of coherent differentiation, and the suitable logical ingredients might be quite different from these categorical constructs. This is exactly what happens with LL itself: there, the logical presentation of the exponential (in sequent calculus or proof-nets) relies on the *structural rules* whereas the categorical presentation relies on the *Seely isomorphisms*. Finding the suitable "logical quarks"[57] of coherent differentiation might be a difficult task which should still be based on the basic intuition that the differential constructs are dual to the structural rules.

- Of course the whole theory of Taylor expansions of programs should be revisited in this new setting.

1.8 Probabilistic coherence spaces and probabilistic functional languages

In the early 2000's I developed with Vincent Danos the theory of *probabilistic coherence spaces*, a notion introduced a few months before by Girard and based on my idea of Köthe spaces 1.6.1.

1.8.1 Probabilistic coherence spaces. The definition of these objects follows the pattern that we have already met several times[58]: a probabilistic coherence space is a pair $X = (|X|, \mathsf{P}X)$ where $|X|$ is a set (the web) and $\mathsf{P}X \subseteq (\mathbb{R}_{\geq 0})^{|X|}$ satisfies $\mathsf{P}X = \mathsf{P}X^{\perp\perp}$ where, if $\mathcal{P} \subseteq (\mathbb{R}_{\geq 0})^I$, one sets $\mathcal{P}^\perp = \{x' \in (\mathbb{R}_{\geq 0})^I \mid \forall x \in \mathcal{P}\ \langle x, x'\rangle = \sum_{i\in I} x_i x'_i \leq 1\}$. One also assumes two more conditions to hold for avoiding the appearance of infinite real numbers[59]. As usual we set $X^\perp = (|X|, \mathsf{P}X^\perp)$. The intuition is that $x \in \mathsf{P}X$ is a probabilistic data of type X and $x' \in \mathsf{P}X^\perp$ is an observation. Then $\langle x, x'\rangle$ is the probability of success of this observation on x. The fact that this probability is not necessarily 1 means that the model accepts partial computations (just like Scott semantics, coherence spaces etc).

[57] I borrow this nice expression to Hugo Herbelin

[58] This is often called "double glueing" in the litterature, but I am not convinced that this abstraction provides any real benefit.

[59] Specifically, for any $a \in |X|$ we assume that the set $\{x_a \mid x \in \mathsf{P}X\}$ is upper bounded and not reduced to $\{0\}$.

A typical example is N where $|\mathsf{N}| = \mathbb{N}$ and PN is the set of all $x \in (\mathbb{R}_{\geq 0})^{\mathbb{N}}$ such that $\sum_{n\in\mathbb{N}} x_n \leq 1$, that is the set of all sub-probability distributions on natural numbers: intuitively $x \in \mathsf{P}\mathbb{N}$ represents a probabilistic program of type ι, x_n is the probability to obtain the result n and $1 - \sum_{n\in\mathbb{N}} x_n$ is the probability that this computation diverges. But not all probabilistic coherence spaces can be interpreted in that simple way, the most obvious example being $\mathsf{N}^\perp$ which has also $\mathbb{N}$ as web, and $x' \in \mathsf{PN}^\perp$ iff $x'_n \leq 1$ for all n; clearly such an x' cannot be seen as a probability distribution in general (for instance we can have $\sum_{n\in\mathbb{N}} x'_n = \infty$).

In [DE11] we developed the semantics of LL in this setting. For example, given two probabilistic coherence spaces X and Y we defined the probabilistic coherence space $X \multimap Y$ whose web is $|X| \times |Y|$ and $t \in \mathsf{P}(X \multimap Y)$ if for all $x \in \mathsf{P}X$ and $y' \in \mathsf{P}Y^\perp$, one has $\sum_{a\in|X|,b\in|Y|} t_{a,b}x_a y'_b \leq 1$. In other words, for any $x \in \mathsf{P}X$ the vector $t\,x = \left(\sum_{a\in|X|} t_{a,b}x_a\right)_{b\in|Y|}$ (application of the matrix t to the vector x) belongs to $\mathsf{P}Y$. When $X = Y = \mathsf{N}$, these matrices are exactly the $\mathbb{N} \times \mathbb{N}$ sub-stochastic matrices. These matrices $\in \mathsf{P}(X \multimap Y)$ are the morphisms of our categorical model of LL. As Girard we defined a tensor product $\otimes$, a co-tensor $\parr$, a direct product $\&$ and a direct sum $\oplus$. But we also introduced an exponential $!X$ whose Kleisli category is cartesian closed as usually, and again all these definitions are based on the relational model: the web of $!X$ is the set of all finite multisets of elements of $|X|$. Given such a multiset m and an $x \in \mathsf{P}X$, one defined $x^m = \prod_{a\in|X|} x_a^{m(a)} \in \mathbb{R}_{\geq 0}$ (this product is finite since m is a finite multiset). Then we set $x^! = (x^m)_{m\in|!X|} \in (\mathbb{R}_{\geq 0})^{|!X|}$ and $\mathsf{P}(!X)$ is the biorthogonal closure of the set of all $x^!$ for $x \in \mathsf{P}X$ (the definition of $!X$ in Köthe spaces and finiteness spaces follows exactly the same pattern).

It follows from this definition that a morphism $X \to Y$ in the Klesili category, which is an element of $\mathsf{P}(!X \multimap Y)$, is a (generalized) power series with non-negative coefficients. Indeed given $t \in \mathsf{P}(!X \multimap Y)$ we can define a function $\widehat{t} : \mathsf{P}X \to \mathsf{P}Y$ by $\widehat{t}(x) = \left(\sum_{m\in|!X|} t_{m,b}x^m\right)_{b\in|Y|}$ and this correspondence $t \mapsto \widehat{t}$ is injective.

For instance, if $X = Y = 1$ (the probabilistic coherence space such that $|1|$ is a singleton and $\mathsf{P}1 = [0,1]$ up to trivial iso), then $|!1| = \mathbb{N}$ and an element of $\mathsf{P}(!1 \multimap 1)$ is simply (up to trivial iso) a sequence $(t_n)_{n\in\mathbb{N}}$ such that $\sum_{n\in\mathbb{N}} t_n \leq 1$. The corresponding function is $\widehat{t} : [0,1] \to [0,1]$ given by $\widehat{t}(x) = \sum_{n\in\mathbb{N}} t_n x^n$.

The corresponding Kleisli category, that is, the category of probabilistic coherence spaces with these analytic functions as morphisms – and composition is the ordinary composition of functions – is cartesian closed. Moreover $\mathsf{P}X$ can always be seen as a complete domain, elements being ordered pointwise. By some standard categorical constructions this implies that for any object X there is an analytic morphism $(!X \multimap X) \to X$ which maps a morphism t to its least fixpoint[60].

Therefore it is possible to interpret a probabilistic extension of PCF in this model, that is a PCF 1.3.2 extended for instance with a constants $\mathsf{coin}(p)$ which reduces to 0 with probability p and to 1 with probability $1 - p$ (one constant for each $p \in \mathbb{Q} \cap [0,1]$ for keeping the language countable). One has to be careful when defining the operational semantics of the language (weak head reduction, see 1.3.2) which can be presented as a Markov chain whose states are the terms of the language: terms rewrite to probability distributions. We developed in [DE11] the denotational interpretation of this language in this cartesian closed category and proved an adequacy theorem expressing that the denotational interpretation $[\![M]\!] \in \mathsf{PN}$ of a closed term M of type ι is such that $[\![M]\!]_n$ is equal to the probability of M to reduce to n.

In the same paper [DE11] we showed that all recursive types (lists, trees, streams, but also models of the pure lambda-calculus and many other complicated types) can be interpreted very simply in this model. Last, we outlined a simple connection between probabilistic coherence spaces and Banach spaces. In [EPT11] we studied the induced probabilistic semantics of the pure lambda-calculus.

Plan for future research:

- Probabilistic coherence spaces, equipped perhaps with a notion of *totality* allowing to sort probability distributions (and their generalizations) out of partial ones, seem to provide a natural

[60] A kind of categorical miracle because the fact that this operation is analytic is not at all obvious *a priori*.

setting for understanding connections between Bayesian inference (Bayesian inversion, Bayesian networks etc) and LL and more specifically, proof-nets. During the LMFI M2 internship of Robin Lemaire (May-August 2021), supervised by myself, Claudia Faggian and Michele Pagani, we made major progresses in this direction, establishing a simple connection between Bayesian Networks and Proof Nets fully compatible with the **Pcoh** semantics of LL and we understand now basic Bayesian networks algorithms as proof manipulations related to MLL cut-elimination.

1.8.2 Full abstraction in probabilistic coherence spaces. With Michele Pagani and Christine Tasson we understood that the presence of probabilities in the syntax greatly increases the separation power of contexts, thus allowing to prove a full abstraction theorem for our interpretation of probabilistic PCF in probabilistic coherence spaces. Without going into technicalities, the proof is based on the fact that, roughly speaking, if two analytic functions coincide on a non empty *open subset* of their domain, they have the same representation as a power series. We published first this full abstraction result in [EPT14] and then we realized that our probabilistic PCF was not a realistic language from the viewpoint of programming.

Indeed this language was *purely call-by-name*, thus in an application $(M)\,N$ where N is a closed term of type ι (which reduces to integers with a sub-probability distribution which coincides with $[\![N]\!]$ as we have seen), each time M uses N, she will sample another integer according to this probability distribution: it seems impossible to sample an integer and *then* use several times the result of this sampling. This makes probabilistic programming quite difficult and probably impossible...

Fortunately this serious difficulty can be overcome very cleanly by observing that N (the probabilistic coherence space interpreting the type ι of integers) has a canonical structure of !-*coalgebra*. This means that there is a canonical morphism $\mathsf{N} \multimap !\mathsf{N}$ (warning: this has nothing to do with the codereliction of DiLL!) which allows to turn any morphism $!\mathsf{N} \multimap X$ into a morphism of type $\mathsf{N} \multimap X$. This operation is essential for interpreting a "let" construct $\mathsf{let}\ x\ \mathsf{be}\ N\ \mathsf{in}\ R$ where N must be of type ι which allows to sample an integer along the distribution defined by N and providing the result of this sampling M as a value for x, that M can use as many times as she wants.

In [EPT18a] we introduced a version of probabilistic PCF extended with a "let" construct restricted to integers as explained above and proved adequacy and full abstraction for this much more realistic programming language which can be understood as a call-by-name language where the terms of ground type ι (and only them) can be used in a call-by-value fashion.

1.8.3 Generalizing to call-by-push-value. The formulas of LL which have a canonical !-coalgebra structure are the *positive* ones, and so it seems natural to consider them as data types (in contrast with functional types which typically are not positive), generalizing the considerations above on ι. All formula of the form $!A$ is positive, and the connectives $\otimes$ and $\oplus$ preserve positivity: this is the key idea of polarized linear logic (see for instance [LR03]).

This means that it is possible to define a notion of *value* for these positive types (values are terms whose interpretation preserve the !-coalgebra structure) which are freely discardable and duplicable. It is possible to define a functional language implementing this idea. It has two kinds of types: positive ones $\varphi, \psi \ldots$ (to be considered as data types) and general ones $\sigma, \tau \ldots$. Any positive type is a general type, and if φ is a positive type and σ is a general type then $\varphi \multimap \sigma$ is a general type. If σ is a general type then $!\sigma$ is a positive type. If φ and ψ are positive types then $\varphi \otimes \psi$ and $\varphi \oplus \psi$ are positive types. It is also easy to add recursive positive types. These types are associated to terms of a λ-calculus where one defines a syntactic notion of value which is crucial in the definition of the operational semantics. This language is extremely close to Paul Levy's *call-by-push-value* and has a clear interpretation in models of LL based on the principles explained above (positive types are interpreted as !-coalgebras, and general types as general objects of the model), this is the object of [Ehr16]. One of its main features is that it allows to combine freely call-by-name and call-by-value.

Then with Christine Tasson we studied in [ET16, ET19] a probabilistic extension of this language, that we interpreted in the model of probabilistic coherence spaces and for which we could again proof adequacy and full abstraction. Since this language is much richer than PCF and in particular features all recursive types, the proofs are significantly more involved than in [EPT18a].

1.8.4 The exponential is free. With Raphaëlle Crubillé, Michele Pagani and Christine Tasson, we proved in [CEPT17] that the exponential "!" presented in 1.8.1 which means that for any object X the object $!X$ is the free commutative comonoid generated by X, giving canonical status to this exponential. At the beginning of this work we were actually trying to build a free exponential using a construction of Melliès, Tabareau and Tasson hoping to define a new exponential on probabilistic coherence spaces. We still do not know if other exponentials are available in this model.

1.8.5 Stable and measurable functions. Probabilistic coherence spaces seem to have intrinsic limitations which make them difficult to use for interpreting functional languages which handle probability distributions on measurable spaces like the real line (in the spirit of Church or Anglican). Since this is an essential feature in current probabilistic programming we have tried to extend our model in this direction, with Michele Pagani and Christine Tasson. Our starting point has been that any probabilistic coherence space can be seen as a *cone* in a sense made explicit by Selinger. A cone is a cancellative $\mathbb{R}_{\geq 0}$-semimodule equipped with a norm which is monotone in the sense that $\|x\| \leq \|x+y\|$. It is also complete in the sense that any increasing sequence (ω-chain) in the closed unit ball of this norm has a sup which belongs to this ball. Here we use the *algebraic* partial order on the cone, defined by $x_1 \leq x_2$ if there is y such that $x_2 = x_1 + y$. By cancellativity when this y exists it is unique and we denote it by $x_2 - x_1$. Then if X is a probabilistic coherence space we get such a cone by taking all $x \in (\mathbb{R}_{\geq 0})^{|X|}$ such that $x \in \alpha \mathsf{P}X$ for some $\alpha \in \mathbb{R}_{\geq 0}$ and for $\|x\|$ the glb of all these αs.

But we have another very simple class of examples which were not available in probabilistic coherence spaces: for any measurable space $\mathcal{X}$ the set of all finite measures on $\mathcal{X}$ (measures μ such that $\mu(\mathcal{X})$ is finite), with algebraic operations defined in the obvious way and $\|\mu\| = \mu(\mathcal{X})$ is a cone.

Our main result, presented in [EPT18b] is that these cones form a cartesian closed category for a certain class of functions that we called *stable functions*. Given two cones P and Q a *stable function from P to Q* is a function f from the unit ball of P to Q which is bounded in Q and satisfies an infinite bunch of inequalities:

$$\begin{aligned}
&f(x) \leq f(x+u)\\
&f(x+u_1)+f(x+u_2) \leq f(x+u_1+u_2)+f(x)\\
&f(x+u_1+u_2)+f(x+u_2+u_3)+f(x+u_1+u_3)+f(x)\\
&\qquad\qquad \leq f(x+u_1+u_2+u_3)+f(x+u_1)+f(x+u_2)+f(x+u_3)\\
&\vdots
\end{aligned}$$

(as soon as the sums which appear in the arguments of f make sense, that is, belong to the unit ball of P). The first condition says that f is monotone wrt. the algebraic order. The second condition expresses that the — therefore well-defined — function $x \mapsto f(x+u)-f(x)$ is monotone in x. The third one expresses that the — therefore well-defined — function $x \mapsto f(x+u_1+u_2)+f(x)-f(x+u_1)-f(x+u_2) = (f(x+u_1+u_2)-f(x+u_1))-(f(x+u_2)-f(x))$ is monotone in x etc. One also assumes that f commutes with the lubs of ω-chains (Scott continuity).

This gives rise to a cartesian closed category[61] which is a model of PCF where we can have a ground type for each measurable space, using the cones defined above from any measurable space. For allowing to interpret the sampling primitive of our language (as explained before, this is basically a "let" construct, but now for all measurable spaces), we need an additional structure on our cones which allow to express that a stable function is "measurable" in some sense. This is done using a fairly standard definition in the spirit of presheaves. Btw. this explains why we restrict completeness to ω-chains (instead of arbitrary directed sets) because we need at some point to use the monotone convergence theorem.

Following a suggestion of mine inspired by a theorem of Bernstein on totally monotone functions, Raphaëlle Crubillé managed to prove during her PhD that, when restricted to the cones generated by probabilistic coherence spaces, these stable functions coincide with the analytic functions presented in 1.8.1.

[61] Surprisingly one of the most tricky point was to prove that this is actually a category, that is, the composition of two stable functions is a stable function, assuming that the first one ranges in the unit ball of its codomain.

Plan for future research:

- This quite non-trivial result shows that our cone model is a conservative extension of probabilistic coherence spaces and opens the way to many applications, such as extending to stable functions the differential operations available in **Pcoh**, see Section 1.8.6, a reserch direction that I would like to investigate with Crubillé.

Given two cones P and Q, there is a natural definition of linear and continuous morphisms from P to Q, and of a cone $P \multimap Q$ whose elements are these morphisms. So it was natural to try and define a tensor product of cones making this category monoidal closed and turning cones into a model of intuitionistic Linear Logic. After several more concrete attempts, I managed to do so in a surprisingly simple way, using the very powerful Special Adjoint Functor Theorem and the (also surprising) observation that probabilistic coherence spaces are "dense" in cones, in a precise technical sense. These results have been presented at LICS in 2020 [Ehr20a].

Plan for future research:

- These results suggest that cones could be used for interpreting programming languages richer than PCF extended with continuous data-types. This still requires however to develop a general theory of integration for maps valued in cones (similar perhaps to Bochner integral in Banach spaces). Another, perhaps more tractable apporach, might be to see integration as a *structure* any of our cones should be equipped with (just as it is equipped with a notion of measurability), following ideas developped recently by Guillaume Geoffroy with whom I would be happy to develop this kind of ideas; he has just been hired as an MCF at IRIF.

1.8.6 Differentiation in probabilistic coherence spaces. One main feature of the probabilistic coherence space model is that the morphisms of its Kleisli category are analytic functions and therefore are extremely regular. For instance, as already explained in Section 1.8.1, in that category, a morphism $1 \to 1$ (remember that 1 is a constant of LL which corresponds to the type unit which has () as single value) is a function $f : [0,1] \to [0,1]$ such that $f(x) = \sum_{n=0}^{\infty} a_n x^n$ with $a_n \in \mathbb{R}_{\geq 0}$ for each n. These functions are in bijective correspondence with the families $(a_n)_{n\in\mathbb{N}}$ in $\mathbb{R}_{\geq 0}^{\mathbb{N}}$ such that $\sum_{n=0}^{\infty} a_n \leq 1$. Such a function f admits a derivative for each $x \in [0,1)$, the slope being possibly infinite at $x = 1$. A typical example is the function[62] $x \mapsto 1 - \sqrt{1-x}$ which is actually the interpretation of a program as explained in [Ehr19].

In that paper I show that each a_n has a natural operational interpretation: thinking of f as the interpretation of a program M of type $1 \to 1$, a_n is the probability that, fed with (), the program M will use its argument[63] n times[64]. Therefore $f'(1) = \sum_{n=0}^{\infty} n a_n$ is the (possibly infinite) expectation of the number of times M will use its argument () if we evaluate $(M)\,()$. It is actually more sensible to compute $f'(1)/f(1)$ which is this expectation conditioned by the termination of $(M)\,()$. By computing such derivatives formally on programs we can evaluate this expectation of termination, as illustrated on an example of [Ehr19], that I would like to generalize introducing a differential probabilistic functional programming language (current research).

The possibility of computing such differentials (or Jacobians), but now for a more general morphism $f : \mathsf{P}X \to [0,1]$ in the category $\mathbf{Pcoh}_!$ at any $x \in \mathsf{P}X$ such that $\|x\| < 1$, combined with the fact that f is upper bounded by 1 and has only non-negative coefficients (when considered as a power series), makes it easy to show that f is Lipschitz of ratio $1/(1-p)$ on the "ball" $\{x \in \mathsf{P}X \mid \|x\| \leq p\}$, for any $p \in [0,1)$. I show how this property can be used to upper-bound a p-tamed version of the observational distance by a distance naturally associated with the norm of the model. Let me explain this point shortly.

In a probabilistic programming language, it is tempting to say that the distance between two terms (programs) M and M' which are closed of type σ is the sup of all $|q_C - q'_C|$ where, given C

[62] The coefficients of its Taylor expansion at 0 are actually all ≥ 0.

[63] The argument is a probabilistic program of type 1, it has only 2 possible behaviors: either produce () in a finite time, or diverge; in the model, it is interpreted by its probability to converge.

[64] In a Krivine-like stack machine.

closed of type $\sigma \Rightarrow 1$ (such a term is a *σ-context*), q_C is the probability that $(C)\,M$ converges and similarly for q'_C and M'. However this distance is well known to be way too discriminating. For instance (taking for σ the type of integers), for any $\varepsilon \in (0,1]$, it puts at distance 1 the program $\underline{0}$ (constant 0) and M_ε which has probability $1-\varepsilon$ to converge to $\underline{0}$ and ε to converge to $\underline{1}$. Nevertheless if we restrict the σ-contexts C used in the definition of the observational distance to be of shape $\lambda x^\sigma\,(D)\,(\mathsf{if}(\mathsf{coin}(p), x, \Omega^\sigma))$ where D is an arbitrary σ-context (that is: before feeding the context with the argument, we allow the argument to diverge with probability $1-p$), the obtained observational distance d_p satisfies $\mathsf{d}_p(M,M') \leq \frac{p}{1-p}\mathsf{d}([\![M]\!],[\![M']\!])$ where d is the distance of the model (and $\mathsf{d}([\![\underline{0}]\!],[\![M_\varepsilon]\!]) = 2\varepsilon$ so that $\mathsf{d}_p(\underline{0}, M_\varepsilon)$ goes to 0 when ε does as one would expect). I could prove this using the above mentioned Lipschitz property, see [Ehr19].

These ideas are at the origin of my current work on coherent differentiation 1.7.5.

1.8.7 Approximation from above in probabilistic coherence spaces. Denotational models such as **Pcoh** provide approximations from below for probabilities of convergence. Consider for instance a closed term M of type ι; we know that $[\![M]\!]_0$ is the probability that M reduces to the value $\underline{0}$. But this probability is obtained as a limit involving the computations of lubs of infinite monotone sequences of probabilities associated with the occurrence of fixpoints subterms (implementing general recursion, or while loops) in M. One can obtain lower approximations by replacing such fixpoint subterms with finite iterations starting from 0. One could obtain similarly approximations from above if we could iterate starting from a maximal element in the corresponding probabilistic coherence space.

The problem is that (just as ordinary coherence spaces) probabilistic coherence spaces in general do not have maximal elements! We can solve this problem by coming back to Berry's old trick of *bidomains*, equipping probabilistic coherence spaces with an "extensional" pre-order relation $\sqsubseteq$. This pre-order is actually defined only on a subset $\mathcal{E}$ of "extensional" elements. Such a pair $(\mathcal{E},\sqsubseteq)$ must be equal to its bidual for a natural notion of duality whose definition swaps in some sense the roles of $\mathcal{E}$ and $\sqsubseteq$. One obtains in that way a categorical model $\mathbf{Pcoh}^{\mathsf{e}}$ of LL of *extensional coherence spaces* whose main feature is that, in $X \multimap Y$, one has $s \in \mathcal{E}_{X\multimap Y}$ iff $\forall x,y \in \mathcal{E}_X\ x \sqsubseteq_X y \Rightarrow s\,x \sqsubseteq_Y s\,y$ and $s \sqsubseteq_{X\multimap Y} t$ iff $s,t \in \mathcal{E}_{X\multimap Y}$ and $\forall x \in \mathcal{E}_X\ s\,x \sqsubseteq_Y t\,x$. It is then possible to interpret ground types as objects of $\mathbf{Pcoh}^{\mathsf{e}}$ which have $\sqsubseteq$-maximal elements. For instance, one can interpret the type of integers with a an object whose underlying probabilistic coherence space is $\mathsf{N}\oplus 1$ (whose web is $\mathbb{N}\cup\{*\}$) and $\mathcal{E} = \mathsf{P}(\mathsf{N}\oplus 1)$; $*$ should be understood as a kind of error element. In this object one stipulates that $x \sqsubseteq y$ if $\sum_{n\in I}(x_n - y_n) \leq y_* - x_*$ where $I = \{n \in \mathbb{N} \mid x_n \geq y_n\}$. Of course $x \leq y \Rightarrow x \sqsubseteq y$ but the converse implication is false: we can have $x \sqsubseteq y$ whithout having $x_n \leq y_n$ for all $n \in \mathbb{N}$ (that is $x \leq y$), the only restriction is that $y_* - x_*$ must be greater than the sum of the "inversions" $x_n - y_n \geq 0$ for the n's such that it is not true that $x_n \leq y_n$. Then the special element $e_* \in \mathsf{P}(\mathbb{N}\oplus 1)$ such that $(e_*)_* = 1$ is $\sqsubseteq$-maximal. These ideas are presented in [Ehr20d] where it is shown how they can be used for addressing our initial upper-approximation problem: the main ingredients are the fact that types are interpreted as $\sqsubseteq$-monotone functions and the existence of $\sqsubseteq$-maximal elements in all types of PCF.

Plan for future research:

- The construction above should clearly be presented as a kind of extension of **Pcoh** with an error monad (in the sense of Haskell) but it is not completely clear how to formalize this in a way which takes properly into account the additional "extensional" structure of the objects of $\mathbf{Pcoh}^{\mathsf{e}}$, this is one of the next steps to be taken in this research direction.

- It is well-known that approximation of fixpoints by iteration is usually slow and inefficient and that Newton method provides often much better results. I plan to adapt Newton method to probabilistic coherence spaces — using the regularity of the morphisms of $\mathbf{Pcoh}_!$ (analytic functions) — for accelerating convergence to fixpoint approximations in the framework of 1.8.7. *A priori* this should work only for lower approximations (which use the algebraic order relation $\leq$), extending it to $\sqsubseteq$-approximations from above will certainly be more challenging. In any case this project is a clear incentive for extending LL with differential constructs compatible with **Pcoh** since this Newton algorithm will require the use of formal differentials of programs: the new setting of coherent differentiation 1.7.5 might be helpful with this respect.

1.9 Linear Logic with fixpoints and its semantics

From March to July 2018, I supervised the M1 MPRI internship of Farzad Jafarrahmani and he has started a PhD under my and Alexis Saurin's supervision in september 2019. We focused our attention on the denotational semantics of linear logic extended with least and greatest fixpoints. In the current litterature this extension $\mu\mathsf{LL}$ of LL is mostly considered from the viewpoint of system specification/verification.

Our starting point was the observation that Girard's coherence spaces provide a natural model of this system, but that this semantics does not make any distinction between least and largest fixpoints (allowing also to interpret contravariant recursive types which are well known to be incompatible with strong normalization, for instance they can be used to define models of the pure lambda-calculus). Our key observation was that Girard's coherence spaces with totality allow to recover the distinction between least and greatest fixpoints as well as strong normalization: totality is a denotational account of strong normalization.

A notion of totality on a coherence space E is a collection of cliques $\mathcal{T}$ such that $\mathcal{T} = \mathcal{T}^{\perp\perp}$ where $\mathcal{T}^{\perp} = \{x' \in \mathsf{Cl}(E^{\perp}) \mid x \cap x' \neq \emptyset\}$ and the set of all notions of totality of a coherence space form a complete lattice (for $\subseteq$). Then we showed how to use the Knaster Tarski theorem to interpret least and greatest type fixpoints non trivially in the coherence space with totality model of LL.

The same approach allows to interpret this logical system in Köthe spaces, finiteness spaces and probabilistic coherence spaces with totality. These results are reported in [EJ19]. We submitted a considerably improved version of this work to LICS'21 where it has been accepted [EJ21]. In that paper we give in particular the first general categorical definition of a "Seely model" of $\mu\mathsf{LL}$ based on the concepts of strong functors and of final coalgebras of functors. One major benefit of considering $\mu\mathsf{LL}$ as we do rather than $\mu\mathsf{MALL}$ (as in earlier work in particular by Baelde, Doumane and Saurin) is that the presence of exponentials[65] makes the presentation of the introduction rule for greatest fixpoints much more natural, by allowing the presence of a context (which must be of shape $?A_1, \ldots, ?A_n$ for deep reasons related to cut-elimination, this is why they are absent from $\mu\mathsf{MALL}$).

Then, together with Alexis Saurin, we have developed a single-sided polarized version of $\mu\mathsf{LL}$ with a term calculus based on ideas borrowed to the work of Curien, Herbelin and Munch-Maccagnoni. We developped a new proof technique for establishing normalization for this kind of calculus in Spring 2021, we are fairly confident that this method (based on reducibility) will smoothly incorporate least and greatest fixpoint constructs by a standard application of the Knaster-Tarski theorem. We also developped in parallel the corresponding categorical notion of *positive strong functor* and the general interpretation of least and greatest fixpoints (as initial algebras and final coalgebras of functors) in this setting. We plan to submit this work to a conference in Fall 2021: the outcome of this work will be a simple normalizing "programming language" extending Gödel's System T, fully compatible with classical logic and offering the possibility of defining a huge class of inductive and coinductive types.

This first semantic work is based on a *finitary* presentation of $\mu\mathsf{LL}$, where the greatest fixpoint rule is a syntactic account of the fact that greatest fixpoints are final object in a category of coalgebras. In this system, proofs are finite trees, but since coinductively defined objects are naturally thought of as infinite structures, it is quite natural to develop logical systems where proofs are infinite trees as in [BDS16]. Of course not all infinte proof trees are correct, only those whose all infinite branches satisfy a criterion dealing with the occurrences of fixpoint formulas. It is not difficult to define the relational (or coherent) semantics of any proof tree, even of incorrect ones; proving that the correct proof trees are interpreted as total sets was one of the goals of the PhD which has been achieved this year and presented by Farzad Jafarrahmani at the TLLA 2021 workshop.

Plan for future research:

- As noticed by Baelde [Bae12], it is possible in $\mu\mathsf{LL}$ to define an exponential as a coinductive formula and this formula has an interesting denotational semantics based on binary trees instead of finite multisets as for the usual exponentials of LL: it is a free pointed (co)magma instead of

[65] In the proof-search perspective of Baelde, there were excellent reasons for rejecting the exponentials: LL provability is a well-known undecidable problem. This is not an issue if we have a Curry-Howard kind of perspective on $\mu\mathsf{LL}$.

being a free commutative (co)monoid. Categorically, it turns out to be a comonad as expected, but does not satisfy the Seely isomorphisms required for giving rise to a cartesian closed Kleisli category. It provides a denotational interpretation of LL which is invariant under cut-elimination but not under the Rétoré conversion rules on proof-nets. I expect that study of the connection between this exponential and the usual one in models such as coherence spaces will shed a new light on the notion of *uniformity* (an interpretation is uniform if it assumes some determinism from the arguments of programs). Also this comagmatic exponential provides a semantics of programs which is "non commutative" in the sense that, just as in game models, the order of elementary operations is faithfully reflected by the interpretation. This analogy with game semantics deserves further inquieries that I would like to undertake in a cooperation with Paul-André Melliès.

- We would like to use our new term calculus based on polarized μLL to better understand the formalization of least and greatest fixpoints in systems such as the Calculus of Inductive Constructions where termination is obtained by means of guardedness conditions.

References

[AJM00] Samson Abramsky, Radha Jagadeesan, and Pasquale Malacaria. Full abstraction for PCF. *Information and Computation*, 163(2):409–470, 2000.

[Bae12] David Baelde. Least and Greatest Fixed Points in Linear Logic. *ACM Trans. Comput. Log.*, 13(1):2:1–2:44, 2012.

[BDS16] David Baelde, Amina Doumane, and Alexis Saurin. Infinitary Proof Theory: the Multiplicative Additive Case. In Jean-Marc Talbot and Laurent Regnier, editors, *25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 - September 1, 2016, Marseille, France*, volume 62 of *LIPIcs*, pages 42:1–42:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.

[BE91a] Antonio Bucciarelli and Thomas Ehrhard. Extensional embedding of a strongly stable model of PCF. In *Proceedings of the 18th International Colloquium on Automata, Languages and Programming*, number 510 in Lecture Notes in Computer Science, pages 35–47. Springer-Verlag, 1991.

[BE91b] Antonio Bucciarelli and Thomas Ehrhard. Sequentiality and strong stability. In *Proceedings of the sixth Symposium on Logic in Computer Science*. IEEE Computer Society Press, 1991.

[BE93] Antonio Bucciarelli and Thomas Ehrhard. A theory of sequentiality. *Theoretical Computer Science*, 113:273–291, 1993.

[BE94] Antonio Bucciarelli and Thomas Ehrhard. Sequentiality in an extensional framework. *Information and Computation*, 110(2):265–296, 1994.

[BE00] Antonio Bucciarelli and Thomas Ehrhard. On phase semantics and denotational semantics in multiplicative-additive linear logic. *Annals of Pure and Applied Logic*, 102(3):247–282, 2000.

[BE01] Antonio Bucciarelli and Thomas Ehrhard. On phase semantics and denotational semantics: the exponentials. *Annals of Pure and Applied Logic*, 109(3):205–241, 2001.

[BEM07] Antonio Bucciarelli, Thomas Ehrhard, and Giulio Manzonetto. Not enough points is enough. In *Proceedings of the 21st Annual Conference of the European Association for Computer Science Logic (CSL'07)*, Lecture Notes in Computer Science. Springer-Verlag, September 2007.

[BEM09] Antonio Bucciarelli, Thomas Ehrhard, and Giulio Manzonetto. A Relational Model of a Parallel and Non-Deterministic lambda-calculus. In Sergei N. Artëmov and Anil Nerode, editors, *LFCS*, volume 5407 of *Lecture Notes in Computer Science*, pages 107–121. Springer, 2009.

[BEM10] Antonio Bucciarelli, Thomas Ehrhard, and Giulio Manzonetto. Categorical Models for Simply Typed Resource Calculi. In *Proceedings of the Twenty-Sixth Conference on the Mathematical Foundations of Programming Semantics - MFPS XXVI*, volume 265 of *Electronic Notes in Theoretical Computer Science*, pages 213–230. Elsevier, 2010.

[BEM12] Antonio Bucciarelli, Thomas Ehrhard, and Giulio Manzonetto. A relational semantics of parallelism and non-determinism in a functional setting. *Annals of Pure and Applied Logic*, 163(7):918–934, 2012.

[Ber78] Gérard Berry. Stable models of typed lambda-calculi. In *Proceedings of the 5th International Colloquium on Automata, Languages and Programming*, number 62 in Lecture Notes in Computer Science. Springer-Verlag, 1978.

[BET12] Richard Blute, Thomas Ehrhard, and Christine Tasson. A convenient differential category. *Cahiers de Topologie et Géométrie Différentielle Catégoriques*, 53, 2012.

[BM20] Davide Barbarossa and Giulio Manzonetto. Taylor subsumes Scott, Berry, Kahn and Plotkin. *Proc. ACM Program. Lang.*, 4(POPL):1:1–1:23, 2020.

[CE87] Thierry Coquand and Thomas Ehrhard. An equational presentation of higher order logic. In *Proceedings of Category Theory in Computer Science 1987*, number 283 in Lecture Notes in Computer Science. Springer-Verlag, 1987.

[CE94] Loïc Colson and Thomas Ehrhard. On strong stability and higher-order sequentiality. In *Proceedings of the 9th Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society, 1994.

[CEPT17] Raphaëlle Crubillé, Thomas Ehrhard, Michele Pagani, and Christine Tasson. The free exponential modality of probabilistic coherence spaces. In Javier Esparza and Andrzej S. Murawski, editors, *Foundations of Software Science and Computation Structures - 20th International Conference, FOSSACS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings*, volume 10203 of *Lecture Notes in Computer Science*, pages 20–35, 2017.

[CES10] Alberto Carraro, Thomas Ehrhard, and Antonino Salibra. Resource Combinatory Algebras. In *Mathematical Foundations of Computer Science 2010, 35th International Symposium, MFCS 2010, Brno, Czech Republic, August 23-27, 2010. Proceedings*, volume 6281 of *Lecture Notes in Computer Science*, pages 233–245. Springer-Verlag, 2010.

[DE11] Vincent Danos and Thomas Ehrhard. Probabilistic coherence spaces as a model of higher-order probabilistic computation. *Information and Computation*, 152(1):111–137, 2011.

[ECS10] Thomas Ehrhard, Alberto Carraro, and Antonino Salibra. Exponentials with Infinite Multiplicities. In *Computer Science Logic, CSL 2010, 19th Annual Conference of the EACSL, Brno, Czech Republic, August 23-27, 2010. Proceedings*, volume 6247 of *Lecture Notes in Computer Science*, pages 170–184. Springer-Verlag, 2010.

[EG16] Thomas Ehrhard and Giulio Guerrieri. The bang calculus: an untyped lambda-calculus generalizing call-by-name and call-by-value. In James Cheney and Germán Vidal, editors, *Proceedings of the 18th International Symposium on Principles and Practice of Declarative Programming, Edinburgh, United Kingdom, September 5-7, 2016*, pages 174–187. ACM, 2016.

[Ehr89a] Thomas Ehrhard. A categorical semantics of constructions. In *Proceedings of the 4th Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society, 1989.

[Ehr89b] Thomas Ehrhard. Dictoses. In *Proceedings of Category Theory in Computer Science 1989*, number 389 in Lecture Notes in Computer Science. Springer-Verlag, 1989.

[Ehr93] Thomas Ehrhard. Hypercoherences: a strongly stable model of linear logic. *Mathematical Structures in Computer Science*, 3:365–385, 1993.

[Ehr96] Thomas Ehrhard. Projecting sequential algorithms on strongly stable functions. *Annals of Pure and Applied Logic*, 77:201–244, 1996.

[Ehr99] Thomas Ehrhard. A relative definability result for strongly stable functions and some corollaries. *Information and Computation*, 152:111–137, 1999.

[Ehr00] Thomas Ehrhard. Parallel and serial hypercoherences. *Theoretical Computer Science*, 247:39–81, 2000.

[Ehr02] Thomas Ehrhard. On Köthe sequence spaces and linear logic. *Mathematical Structures in Computer Science*, 12:579–623, 2002.

[Ehr04] Thomas Ehrhard. A completeness theorem for symmetric product phase spaces. *Journal of Symbolic Logic*, 69(2):340–370, 2004.

[Ehr05] Thomas Ehrhard. Finiteness spaces. *Mathematical Structures in Computer Science*, 15(4):615–646, 2005.

[Ehr09] Thomas Ehrhard. The Scott model of Linear Logic is the extensional collapse of its relational model. CCSD-HAL hal-00369831, Preuves, Programmes et Systèmes, 2009. Submitted for publication.

[Ehr10] Thomas Ehrhard. A finiteness structure on resource terms. In *LICS*, pages 402–410. IEEE Computer Society, 2010.

[Ehr12a] Thomas Ehrhard. Collapsing non-idempotent intersection types. In *Computer Science Logic, 21nd Annual Conference of the EACSL, CSL 2012, September, 2012, Fontaineblau, France, Proceedings*, volume 16 of *LIPIcs*, pages 259–273. Schloss Dagstuhl - Leibniz Zentrum fuer Informatik, 2012.

[Ehr12b] Thomas Ehrhard. The Scott model of Linear Logic is the extensional collapse of its relational model. *Theoretical Computer Science*, 424:20–45, 2012.

[Ehr16] Thomas Ehrhard. Call-By-Push-Value from a Linear Logic Point of View. In Peter Thiemann, editor, *Programming Languages and Systems - 25th European Symposium on Programming, ESOP 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings*, volume 9632 of *Lecture Notes in Computer Science*, pages 202–228. Springer-Verlag, 2016.

[Ehr18] Thomas Ehrhard. An introduction to differential linear logic: proof-nets, models and antiderivatives. *Mathematical Structures in Computer Science*, 28(7):995–1060, 2018.

[Ehr19] Thomas Ehrhard. Differentials and distances in probabilistic coherence spaces. In Herman Geuvers, editor, *4th International Conference on Formal Structures for Computation and Deduction, FSCD 2019, June 24-30, 2019, Dortmund, Germany.*, volume 131 of *LIPIcs*, pages 17:1–17:17. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019.

[Ehr20a] Thomas Ehrhard. Cones as a Model of Intuitionistic Linear Logic. In Holger Hermanns, Lijun Zhang, Naoki Kobayashi, and Dale Miller, editors, *LICS '20: 35th Annual ACM/IEEE Symposium on Logic in Computer Science, Saarbrücken, Germany, July 8-11, 2020*, pages 370–383. ACM, 2020.

[Ehr20b] Thomas Ehrhard. Differentials and Distances in Probabilistic Coherence Spaces (extended version). *CoRR*, abs/2005.12582, 2020.

[Ehr20c] Thomas Ehrhard. Non-idempotent Intersection Types in Logical Form. In Jean Goubault-Larrecq and Barbara König, editors, *Foundations of Software Science and Computation Structures - 23rd International Conference, FOSSACS 2020, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2020, Dublin, Ireland, April 25-30, 2020, Proceedings*, volume 12077 of *Lecture Notes in Computer Science*, pages 198–216. Springer, 2020.

[Ehr20d] Thomas Ehrhard. Upper approximating probabilities of convergence in probabilistic coherence spaces. *CoRR*, abs/2008.04534, 2020.

[Ehr21a] Thomas Ehrhard. Coherent differentiation. *CoRR*, abs/2107.05261, 2021.

[Ehr21b] Thomas Ehrhard. Differentials and distances in probabilistic coherence spaces. *CoRR*, abs/1902.04836, 2021. Long version of an FSCD'19 article with the same title.

[EJ15] Thomas Ehrhard and Ying Jiang. A dendroidal process calculus. Technical report, 2015.

[EJ19] Thomas Ehrhard and Farzad Jafar-Rahmani. On the denotational semantics of linear logic with least and greatest fixed points of formulas. *CoRR*, abs/1906.05593, 2019.

[EJ21] Thomas Ehrhard and Farzad Jafar-Rahmani. Categorical models of linear logic with fixed points of formulas. In *36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2021, Rome, Italy, June 29 - July 2, 2021*, pages 1–13. IEEE, 2021.

[EJK19] Thomas Ehrhard, Ying Jiang, and Jean Krivine. A Calculus of Branching Processes. *Theoretical Computer Science*, 2019. To appear.

[EL06] Thomas Ehrhard and Olivier Laurent. Embedding the finitary pi-calculus in differential interaction nets. In *Proceedings of the Higher Order Rewriting workshop (HOR 2006)*, 2006. Electronic publication.

[EL07] Thomas Ehrhard and Olivier Laurent. Interpreting a Finitary Pi-calculus in Differential Interaction Nets. In Luís Caires and Vasco Thudichum Vasconcelos, editors, *CONCUR*, volume 4703 of *Lecture Notes in Computer Science*, pages 333–348. Springer, 2007.

[EL10a] Thomas Ehrhard and Olivier Laurent. Acyclic Solos and Differential Interaction Nets. *Logical Methods in Computer Science*, 6(3), 2010.

[EL10b] Thomas Ehrhard and Olivier Laurent. Interpreting a finitary pi-calculus in differential interaction nets. *Information and Computation*, 208(6):606–633, 2010.

[EPT11] Thomas Ehrhard, Michele Pagani, and Christine Tasson. The computational meaning of probabilistic coherent spaces. In *Proceedings of the 26th Annual IEEE Symposium on Logic in Computer Science, LICS 2011, June 21-24, 2011, Toronto, Ontario, Canada*, pages 87–96. IEEE Computer Society, 2011.

[EPT14] Thomas Ehrhard, Michele Pagani, and Christine Tasson. Probabilistic coherence spaces are fully abstract for probabilistic PCF. In Suresh Jagannathan and Peter Sewell, editors, *POPL*, pages 309–320. ACM, 2014.

[EPT18a] Thomas Ehrhard, Michele Pagani, and Christine Tasson. Full Abstraction for Probabilistic PCF. *Journal of the ACM*, 65(4):23:1–23:44, 2018.

[EPT18b] Thomas Ehrhard, Michele Pagani, and Christine Tasson. Measurable cones and stable, measurable functions: a model for probabilistic higher-order programming. *Proc. ACM Program. Lang.*, 2(POPL):59:1–59:28, 2018.

[ER03] Thomas Ehrhard and Laurent Regnier. The differential lambda-calculus. *Theoretical Computer Science*, 309(1-3):1–41, 2003.

[ER06a] Thomas Ehrhard and Laurent Regnier. Böhm trees, Krivine machine and the Taylor expansion of ordinary lambda-terms. In Arnold Beckmann, Ulrich Berger, Benedikt Löwe, and John V. Tucker, editors, *Logical Approaches to Computational Barriers*, volume 3988 of *Lecture Notes in Computer Science*, pages 186–197. Springer-Verlag, 2006. Long version available on http://www.irif.fr/~ehrhard/.

[ER06b] Thomas Ehrhard and Laurent Regnier. Differential interaction nets. *Theoretical Computer Science*, 364(2):166–195, 2006.

[ER08] Thomas Ehrhard and Laurent Regnier. Uniformity and the Taylor expansion of ordinary lambda-terms. *Theoretical Computer Science*, 403(2-3):347–372, 2008.

[ET16] Thomas Ehrhard and Christine Tasson. Probabilistic call by push value. Technical report, 2016.

[ET19] Thomas Ehrhard and Christine Tasson. Probabilistic call by push value. *Logical Methods in Computer Science*, Volume 15, Issue 1, January 2019.

[Gir86] Jean-Yves Girard. The system F of variable types, fifteen years later. *Theoretical Computer Science*, 45:159–192, 1986.

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[Gir88] Jean-Yves Girard. Normal functors, power series and the λ-calculus. *Annals of Pure and Applied Logic*, 37:129–177, 1988.

[Gir99] Jean-Yves Girard. On denotational completeness. *Theoretical Computer Science*, 227:249–273, 1999.

[Gri90] Timothy G. Griffin. The formulae-as-types notion of control. In *Proceedings of the 17h ACM Symposium on Principles of Programming Languages (POPL)*, pages 47–57. Association for Computing Machinery, January 1990.

[HO00] Martin Hyland and Luke Ong. On full abstraction for PCF: I, II, and III. *Information and Computation*, 163(2):285–408, 2000.

[JLE19] Ying Jiang, Shichao Liu, and Thomas Ehrhard. A fully abstract semantics for value-passing CCS for trees. *Frontiers Comput. Sci.*, 13(4):828–849, 2019.

[Lef42] Solomon Lefschetz. *Algebraic topology*. Number 27 in American mathematical society colloquium publications. American Mathematical Society, 1942.

[LR03] Olivier Laurent and Laurent Regnier. About Translations of Classical Logic into Polarized Linear Logic. In *18th IEEE Symposium on Logic in Computer Science (LICS 2003), 22-25 June 2003, Ottawa, Canada, Proceedings*, pages 11–20. IEEE Computer Society, 2003.

[Nic94] Hanno Nickau. Hereditarily sequential functionals. In Anil Nerode and Yu. V. Matiyasevich, editors, *Proc. Symp. Logical Foundations of Computer Science: Logic at St. Petersburg*, volume 813 of *Lecture Notes in Computer Science*, pages 253–264. Springer-Verlag, 1994.