Reductions and linear approximations

Luc Pellissier

Université Paris 13
—
Université Sorbonne Paris Cité

École doctorale Galilée

THÈSE
en vue d'obtenir le grade de
DOCTEUR DE L'UNIVERSITÉ SORBONNE PARIS CITÉ
Spécialité Informatique

Réductions & Approximations Linéaires

Présentée et soutenue publiquement par
Luc Pellissier
le 8 décembre 2017

devant les membres du jury:	M. Pierre-Louis Curien	
	M. Stefano Guerrini	*directeur*
	M. Damiano Mazza	*co-directeur*
	M. Guy McCusker	
	M. Luke Ong	
	M. Nicolas Tabareau	
	M$^{\text{me}}$ Christine Tasson	
et suite aux rapports de:	M. Guy McCusker	
	M. Thomas Streicher	

Remerciements

Mes premiers remerciements vont bien sûr à Damiano Mazza, qui m'a encadré avec une très grande souplesse et une énergie hors du commmun, et qui a toujours su m'accorder sa confiance. Si ces années se sont bien passées, c'est évidemment grâce à lui. Je remercie aussi évidemment Stefano Guerrini, qui a toujours été présent quand j'avais des résultats à présenter ou des problèmes à surmonter.

Je suis reconnaissant à Thomas Streicher et Guy McCusker d'avoir accepté de rapporter cette thèse, ce qu'ils ont fait avec précision et rapidité. Leurs nombreuses remarques sur le manuscrit ont contribué à l'améliorer. Je remercie aussi Pierre-Louis Curien, Luke Ong, Nicolas Tabareau et Christine Tasson d'avoir accepté d'être membres du jury de cette thèse, malgré un emploi du temps chargé. Leurs travaux sont une inspiration ; leur présence m'honore.

Cette thèse n'aurait sûrement pas eu lieu sans Paul-André Melliès, qui m'a aiguillé vers Damiano Mazza et incité à travailler sur des sujets me semblant hors d'atteinte. Les conversations avec lui sont toujours des moments de remise en question, de chamboulement.

J'ai eu l'occasion de collaborer sur des sujets repris dans ce manuscrit avec, en plus de Damiano Mazza, Giulio Guerrieri, Lorenzo Tortora de Falco et Pierre Vial ; et sur des directions peut-être moins fructueuses avec Pierre Boudes et Antoine Kaszczyc. Ils m'ont tous apporté quelque chose qui restera ; je m'excuse de tous mes manques à leur égard.

Cette thèse s'est nourrie de quelques années passées au Lipn, un laboratoire qui marche malgré tout. Merci à tout ceux qui en font un environnement vivant, au-dedans et au-dehors de l'axe Lo(gique). J'ai trouvé que c'était bien, la logique au milieu des moutons. C'était sa place. Les derniers mois, passés au Lip, m'ont fourni une autre expérience, elle aussi très enrichissante. Merci à toute l'équipe Plume. Plus généralement, je remercie aussi toute l'école française de logique, et particulièrement ses pionniers, d'avoir créé et entretenu un cadre de travail fécond.

Bien que le cadre matériel (Kemasang 2009) et personnel au-delà des collaborateurs au sens strict (Chauvac and Bès 2011) ait une grande importance, je ne vais pas pousser plus loin cet exercice. Je n'ai jamais été seul ; merci à toutes celles et ceux qui m'ont entouré.

Reductions & Linear approximations

Luc Pellissier

Contents

Introduction

Ce principe [doit] dominer la politique des nations aussi bien que celle des particuliers : *Quand l'effet produit n'est plus en rapport direct ni en proportion égale avec sa cause, la désorganisation commence.*

Balzac, *César Birotteau*

Reasoning about proofs and programs

What is a proof? It can be described as the process of making the necessary connection between premises and a conclusion explicit. And what is a program? In the same way, it can be defined as the process of making explicit a result, given parameters. The two notions, of proof and of program, are connected via the *Curry–Howard correspondence*, summed up in Figure 1 between proofs, programs, and natural operations in both contexts (Howard 1980).

	Logic	Programming Theory
Specification	Formula	Type
Static	Proof	Program
Dynamic	Cut-elimination	Computation

Figure 1: The Curry–Howard correspondence, in a nutshell

A *formula* is an assembly of symbols, satisfying certain rules of construction. A *proof* is a mathematical object, specified in a certain formal system, such as minimal implicative natural deduction **NJ** (see Figure 2), that embeds a formula that it proves. Proofs are then simplified by *cut-elimination*, a dynamic procedure that explicits the proof by deleting all the unnecessary generalizations.

A *type* is an assembly of symbols, satisfying certain rules of construction. A *program* is a mathematical object, specified in a certain formal system, such as simply-typed λ-calculus $\Lambda_{\rightarrow}$ (see Figure 3), that embeds a type which it is typed by. Programs are then simplified by *computation*, a dynamic procedure that explicits the result of the program by simplifying the computation steps.

In both cases, the dynamical aspect is of paramount importance.

Reasoning mathematically about a dynamical process is notoriously hard: while geometry and number theory were developed early in human history, a mathematical theory of gravitation had to wait for the late 17th century. The situation is even worse in situations when the dynamics is not governed by the physical time, but by an other, more subtle, temporality. Indeed, the recognition of the importance of the dynamical aspects of the proofs has been started by Gentzen (1936), centuries after the first investigation on the rules of reasoning by the ancient Greeks[1]. Great progresses have been made, in particular inspired by the Curry–Howard correspondence, in the past century. It is our thesis that the dynamics of linear approximations of proofs and programs give a unifying point of view on many developments around the formal study of proofs and programs.

Linearity

It can be argued that there is no concept of paramount importance for our ability to understand the world than *linearity*. Without touching the social aspects of it – Balzac, for instance, in 1837, asserts that society goes astray when capitalism abandons a proportionality between an effect and its cause – or the moral ones, of retribution or *Tun-Ergehen Zusammenhang*, modern science and modern thought have been built around the refutation of *causal over-determination*: there can be only one sufficient cause for each effect[2]. Linearity is but an extension of this principle: not only do effects have a unique sufficient cause, their intensity is dependent on the intensity of the cause, and in the simplest possible way.

[1] Although we tend to give to Aristotle the credit of inventing logic, he benefited a lot from the work – that he derided – of the Eleatic school, whose most well-known members are Parmenides and Zeno, who are said to be the inventors of the *reductio per absurdo* and of the principle of contradiction (Colli 1998).

[2] This may seem peculiar to the ear of a 21st century person, but has not always been the case. We can think of the Greek myths as examples of over-determination: Œdipus kills his father Laius because he refused to let him go first at a crossroad and because of a malediction on its family – both causes being independently sufficient.

$$\frac{}{\mathrm{A} \vdash \mathrm{A}}\,(\text{axiom}) \qquad \frac{\Gamma, \mathrm{A} \vdash \mathrm{B}}{\Gamma \vdash \mathrm{A} \to \mathrm{B}}\,(\to\text{-introduction})$$

$$\frac{\Gamma \vdash \mathrm{A} \to \mathrm{B} \qquad \Delta \vdash \mathrm{A}}{\Gamma ; \Delta \vdash \mathrm{B}}\,(\to\text{-elimination})$$

Figure 2: Logical rules of minimal intuitionnistic logic in natural deduction **NJ**

$$\frac{}{x : \mathrm{A} \vdash x : \mathrm{A}}\,(\text{variable}) \qquad \frac{\Gamma, x : \mathrm{A} \vdash t : \mathrm{B}}{\Gamma \vdash \lambda x.t : \mathrm{A} \to \mathrm{B}}\,(\text{abstraction})$$

$$\frac{\Gamma \vdash t : \mathrm{A} \to \mathrm{B} \qquad \Delta \vdash u : \mathrm{A}}{\Gamma ; \Delta \vdash tu : \mathrm{B}}\,(\text{application})$$

Figure 3: Simply-typed λ-calculus $\Lambda_{\to}$, without structural rules

The study of linearly dependent quantities, *linear algebra* has been an incredibly successful subfield of mathematics. Through its pervasiveness, the variety of approach it can harness and its numerous possibilities of generalizations, it provides an indispensable common ground, providing the (often unconscious) backbone to the rest of the mathematical activities.

In fact, for all the talking about new kinds of science, emerging spontaneously from huge quantities of data, be they understood through cellular automata or through deep neural networks, the greatest achievements of 20$^{\text{th}}$ century science are stubbornly linear. Let us cite only quantum mechanics, Grothendieck's theory of motives, the whole of particle physics or representation theory. Without making a dogma of linearity, it can be cautious to think that it might continue to be the case, and that linearity will still be a cornerstone of human thought in the foreseeable future.

Approximations

Despite the pleasing conceptual simplicity of linearity, most phenomena do not behave strictly so. Indeed, even objects as ubiquitous as polynomials do not define linear functions in the general case. Linear models that were instrumental in shaping a field can also be discarded: the hallmark of the scientific revolution, the gradual comprehension of the theory of universal gravitation, an eminently linear theory, has been replaced in the first half of last century by Einstein's relativity which is not.

Nonetheless, many phenomena can be approximated by linear ones, which means that not only can a linear phenomenon be studied instead of a non-linear one, because it is easier to study; but that the error this shift of object of study causes can be apprehended, evaluated, bounded. So, for instance, under suitable hypotheses, a function f from the real numbers to the real numbers can be approximated, in the neighborhood of a point x, by a linear function $\mathrm{d}f_x$, its differential, and the difference between the two can be bounded by a known quantity.

Given the sheer variety and importance of non-linear phenomena, and the success and power of linear methods, it is no wonder that the task of approximating non-linear behaviors with linear ones has concentrated mathematicians' work for decades. Actually, mathematics has jokingly been described as the task of reducing every problem to linear algebra (it is at least the motto of some subfields of mathematics such as representation theory: study every group as a subgroup of the group of automorphisms over a vector space).

Types, denotational semantics and syntactical approximations

Approximations made different appearances in theoretical computer science and in proof theory. Let us concentrate on three such incarnations:

Types Types, and type derivations, can be seen as approximations of a program behavior: the fact, for instance, that a λ-term can be typed with a simple type $A \to B$ implies that the term interacts well when applied to any other term of type A. Depending on the type system, the type of a term witnesses different properties of the term: for instance, there are type systems ensuring different kinds of normalization or of other side effects. Every term having the same type can be expected to behave, according to certain approximation, in the same way.

A particular class of type systems, called *intersection types systems*, will be our main object of study. Their general philosophy is to allow for a sub-term to be typed in different ways in the same type derivation, so as to approximate different possible behaviors of the term.

Set-based semantics To understand better proofs and programs, a line of work traced back to Scott (1970) has been to interpret them in other well-known mathematical structures. So, for instance, the arithmetical expressions (built on constants and arithmetical operations $+, \times, \ldots$) can be interpreted as integers, and the rewriting rules of mathematical expression be proved sound by showing that an expression has the same interpretation that any other expression it rewrites to. The notion of approximation has been central since the first semantics of the λ-calculus, Scott domains (Scott 1970): the intuition behind Scott domains is that the only functions representable in the λ-calculus are continuous, in the sense that a program that terminates only reads a finite quantity of information of its input before returning its output. So, the output of a program on an infinite input is equal to the output of the same program on any sufficiently large (in the sense that it contains enough information) finite approximation of the input.

The idea of Scott domain is then to interpret types as partial orders (representing the information order) closed by certain operation and terms as functions that respect these operations. Other set-based semantics interpret also types as sets with structure, and in some cases, it is possible to view elements of these sets as approximations.

Syntactical approximations Syntactical approximations have become an object of study only later: indeed, for a good notion of syntactic approximations to be devised, there is a need for a good definition of the language of approximations.

The object of this thesis is to argue that syntactical approximations can be taken as the fundamental object, from which we can define and compute types and semantics.

Linear logic and Girard's approximation theorem

Linearity made its grand debut in logic and computer science with the work of Girard (1987),[3] where a decomposition of intuitionistic logic is introduced around the notion of linearity. In layman's terms, in linear logic, an hypothesis can only be used once during the course of reasoning: as such, a valid theorem in intuitionistic logic such as

$$A \Rightarrow A \wedge A,$$

(where $\Rightarrow$ is the intuitionistic implication and $\wedge$ the conjunction) is, brutally translated in linear logic as

$$A \multimap A \otimes A$$

(where $\multimap$ is the linear implication and $\otimes$ the multiplicative conjunction, that is, conjunction in the strongest possible sense) no longer provable for an arbitrary proposition A. This restriction is as drastic as it seems: proofs by induction are no longer possible (as the hypothesis of induction is used an arbitrary number of times), the conjunction of a formula and its (linear) negation does not imply everything…

Linear logic overcomes this restriction by not being linear at all: untamed reasoning, using all the power of infinity is available, but carefully circumscribed inside modalities $!$ and $?$. Nonetheless, linear logic, whose sequent calculus formulation **LL** is recalled in Figure 4, successfully separates linear and non-linear steps of reasoning. So, while the proposition $A \multimap A \otimes A$ is not in general provable, the proposition

$$!A \multimap A \otimes A$$

whose intuitive meaning is: "given an arbitrary number of the hypothesis A, A is proved twice" is provable, as well as

$$!A \multimap !A \otimes !A.$$

By delineating a strictly linear fragment of logic inside the larger logic, linear logic allows to define clearly approximations.

Indeed, ever since the seminal article of Girard, the rôle of approximations in linear logic has been recognized. The article actually ends with an approximation theorem, which shows a form of density of exponential-free linear logic in full linear logic, and thus justifies a crude form of approximations. If we define the *bounded exponential*

$$!_p A := \overbrace{(A \,\&\, 1) \otimes \cdots \otimes (A \,\&\, 1)}^{p \text{ times}},$$

and $?_p A := (!_p A^{\perp})^{\perp}$, which are exponential-free formulæ if A is, we have:

Theorem 1 (*Girard, 1987*)

> *Let A be a theorem of linear logic; with each occurrence of $!$ in A, assign a positive integer; then it is possible to assign positive integers to all occurrences of $?$ in such way that if B denotes the result of replacing each occurrence of $!$ (respectively $?$) by $!_n$ (respectively $?_n$) where n is the integer associated*

[3] For completeness' sake, ideas were already floating around in Lambek (1958) calculus and relevance logics.

$$\frac{\vdash \Gamma}{\vdash \sigma(\Gamma)}\text{(exchange, with }\sigma\text{ a permutation)}$$

(a) Structural rule

$$\frac{}{\vdash A, A^{\perp}}\text{(identity)} \qquad \frac{\vdash \Gamma, A \qquad \vdash \Delta, A^{\perp}}{\vdash \Gamma, \Delta}\text{(cut)}$$

(b) Identity rules

$$\frac{}{\vdash 1}\text{(one)} \qquad \frac{\vdash \Gamma}{\vdash \Gamma, \bot}\text{(bottom)} \qquad \frac{\vdash \Gamma, A \qquad \vdash \Delta, B}{\vdash \Gamma, \Delta, A \otimes B}\text{(tensor)} \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}\text{(par)}$$

(c) Multiplicative rules

$$\frac{}{\vdash \Gamma, \top}\text{(top)} \qquad \frac{\vdash \Gamma, A \qquad \vdash \Gamma, B}{\vdash \Gamma, A \& B}\text{(with)} \qquad \frac{\vdash \Gamma, A}{\vdash \Gamma, A \oplus B}\text{(plus}_1\text{)} \qquad \frac{\vdash \Gamma, B}{\vdash \Gamma, A \oplus B}\text{(plus}_2\text{)}$$

(d) Additive rules

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, ?A}\text{(dereliction)} \qquad \frac{\vdash \Gamma}{\vdash \Gamma, ?A}\text{(weakening)} \qquad \frac{\vdash \Gamma, ?A, ?A}{\vdash \Gamma, ?A}\text{(contraction)} \qquad \frac{\vdash ?\Gamma, A}{\vdash ?\Gamma, !A}\text{(promotion)}$$

(e) Exponential rules

Figure 4: Linear Logic sequent calculus **LL** (Girard 1987) presented with unilateral sequents

> *to it, then* B *is still a theorem of linear logic.*

For example, from the canonical proof of $!A \multimap (!A \otimes !A)$ (contraction, or duplication), we get proofs of

$$!_{p_1+p_2}A \multimap (!_{p_1}A \otimes !_{p_2}A)$$

for all $p_1, p_2 \in \mathbf{N}$, which means intuitively that, from $p_1 + p_2$ copies of A, one can get at least p_1 copies and separately, at least p_2 copies.

In many ways, the thesis that will follow is a dance around this approximation theorem, trying to collect and organize the many different ways of approximating proofs and programs.

Dynamical models

Let us turn back to the basic objects of study, that is, proofs and programs. As we saw, their dynamic properties are essential. One great inspiration is Girard's *sous-sols* (Girard 2006, Section 7.1). We try here to capture these intuitions, eyeing to a mathematical structure:

First level: truth, specification This level is populated by formulæ, that we will write with roman capital letters

$$A, B, C, \ldots$$

$$\begin{tikzcd}
 & \text{models} \arrow[dl, leftrightarrow] \arrow[dr, leftrightarrow] & \\
\text{intersection type systems} \arrow[rr, leftrightarrow] & & \text{linear approximations}
\end{tikzcd}$$

Figure 5: A well-known correspondence

or with star-shaped symbols

$$\bigstar, *, \dots$$

In general, we will not assume any structure on the formulæ. On certain cases, we will assume the existence of some constructors, for instance a binary operation $\odot$, such that for every formulæ A and B, there exists a formula

$$A \odot B.$$

In some cases, we will also ask certain equations to be verified of the formulæ. For instance, we could pose

$$A \odot B = B \odot A.$$

This level is sufficient to express the grammar of propositions, but is not able to express anything related to their validity.

Second level: static compositionality In first approximation, an hypothetical argument proves a formula of the form:

$$\text{If some hypotheses } A_1, \dots, A_n \text{ are true, then B is true.}$$

In the same way, a program can be specified in the form:

$$\text{Given some inputs } A_1, \dots, A_n, \text{B is computed.}$$

In both cases, we see that the terms present a fundamental assymetry: they relate a list of objects on one side to a single object on the other side. As such, we will represent them as arrows

$$A_1, \dots, A_n \longrightarrow B$$

Though possible, we will not consider any particular structure on the arrows.

This level is a language proficient enough to express proofs and provability, but not their dynamics.

Third level: dynamism Reasoning proceeds by a succession of over-generalizations followed by specializations. These unnecessary steps can then be removed by following an explicitation procedure. Abstractly, it relates two terms of the same kind

$$\begin{tikzcd}
A_1, \dots, A_n \arrow[r, bend left, "t_1"] \arrow[r, bend right, "t_2"'] & B
\end{tikzcd}$$

in an oriented fashion:

$$\begin{tikzcd}
A_1, \dots, A_n \arrow[r, bend left, "t_1"{name=U}] \arrow[r, bend right, "t_2"'{name=D}] & B \arrow[Rightarrow, from=U, to=D]
\end{tikzcd}$$

meaning that the term t_1 reduces to the term t_2 of same type.

Calculi and languages

The calculi we will focus on are represented Figure 6. In particular, we will study linear approximations of:

- the simply-typed λ-calculus Λ, the prototypical programming language, whose only computation mechanism is unrestricted substitution of a variable by an argument;
- the simply-typed call-by-value λ calculus Λ_v, where the substitution can only happen when the argument is of a particular syntactic form, which is closer to the evaluation of implemented programming languages such as the `ML` family;
- the simply-typed $\lambda\mu$-calculus ΛM, which incorporates features allowing for having different outputs and switching between them, allowing for exception mechanisms (from a programming language viewpoint) and classical reasoning (from a logical viewpoint),

as well as the untyped versions of these calculi, respectively $\Lambda_\star$, $\Lambda_{v,\star}$, and $\Lambda M_\star$.

One of the ways in which we leverage linear logic is by translating these calculi we are interested in to a calculus based on linear logic, allowing to define approximations only once, in a calculus that already explicitely marks its non-linear features. We will introduce two calculi based on linear logic:

- the simply-typed intuitionnistic linear λ-calculus $\Lambda_!$, in which we translate the λ-calculus and the call-by-value λ-calculus through the so-called Girard translations;
- linear logic proof-nets $\mathsf{MELL}_?$, which is a calculus that allows to translate transparently classical features of the $\lambda\mu$-calculus, on top of $\Lambda_!$,

as well as their untyped variants $\Lambda_{!,\star}$ and $\mathsf{MELL}_{?,\star}$. $\mathsf{MELL}_{?,\star}$ actually has two types, one representing all the linear types, and one the cartesian ones.

Operads

Operads[4] offer a structure accommodating these three levels. More precisely, we will present most calculi as **Cat**-operads: structures whose elements can be exactly pictured as above: a **Cat**-operad is

[4] We use the term in a slightly non-standard way for *symmetric multicategory*.

characterized by its set of object (0-dimensional arrows), of multi-arrows (1-dimensional arrows, or arrows between objects) and 2-arrows (2-dimensional arrows, or arrows between arrows).

Classical systems, such as $\lambda\mu$-calculus or MELL proof-nets do not fit in this framework: an essential feature of such systems is that they either have multiple conclusions at the same time (in the case of MELL proof-nets) or can switch their main conclusion at will (in the case of $\lambda\mu$-calculus. This motivates the move from **Cat**-operads to more general structures, such as cyclic **Cat**-operads (where every input of a multi-arrow can be exchanged with the output) or even semi-cyclic **Cat**-operads (where some of the inputs are marked and can be exchanged with the output).

Just as the calculi fit in the operadic framework, their models do too. We believe that, in the same way the systematic study of linear approximations offered a new clarity to the study of models of linear logic and the λ-calculus (for instance, differential linear logic clarified the study of the relational model), linear approximations are the first step to study and understand more general operadic models, where the interpretation of a term is not an invariant of the reduction but varies.

Just as our calculi are represented by **Cat**-operads, translations between them are represented as morphisms of **Cat**-operads. More precisely, such a morphism $f : \mathscr{D} \to \mathscr{C}$ is a modular, semantic-preserving translation of calculi:

- a type A in $\mathscr{D}$ is encoded by $f(\mathrm{A})$ in $\mathscr{C}$;
- a term $x_1 : \mathrm{C}_1, \dots, x_n : \mathrm{C}_n \vdash t : \mathrm{A}$ of $\mathscr{D}$ is encoded by $x_1 : f(\mathrm{C}_1), \dots, x_n : f(\mathrm{C}_n) \vdash f(t) : f(\mathrm{A})$;
- the translation is sound with respect to substitution;
- the translation preserves the reductions.

Syntactic approximations

The starting point of our syntactic approximations is the desire to give a precise meaning, at the level of terms, of the equation, inspired by Girard's approximation theorem:

$$!\mathrm{A} = \lim_{p\to\infty} \overbrace{(\mathrm{A} \mathbin{\&} 1) \otimes \cdots \otimes (\mathrm{A} \mathbin{\&} 1)}^{p \text{ times}}.$$

This prompts us to introduce a family of connectives, one for each arity, representing this n-fold tensor product. So, for intuitionistic linear logic, we consider a calculus containing a sequence constructor $\langle\cdot\rangle$, and for classical linear logic, we introduce a ! cell for every arity. We will do the following discussion in the intuitionistic case, as it is easier to write.

A term !M (meaning M at will) will thus be approximated by terms of the form:

$$\langle t_1, \dots, t_n \rangle$$

where all the $t_1, \dots, t_n$ approximate M, recursively. Dually, an abstraction has to cope with many different inputs and needs *tracks*: indices indicating which approximation goes to which instance of the variable.

Although the constructor itself is non-commutative, and is not endowed with any operation giving it a structure more complicated than that of a list, some flexibility can be recovered: indeed, we can embed some structural rules, making the list constructor able to look like other structures such as sets or multisets. By choosing or not the different structural rules, we define different flavors of syntactic approximation. Following linear logic, we will always consider exchange, but not weakening and contraction. This yields four calculi:

linear in the linear polyadic setting, no structural rule is allowed. A linear polyadic term can be seen as a rigid representation of a resource term (Boudol 1993; Tsukada, Asada, and Ong 2017). Nonetheless, if an approximation $\langle t_1, \dots, t_n \rangle$ approximates !M, any permutation of it approximates it too.

affine in the affine polyadic setting, only weakening is allowed, which means that approximations can be discarded at will. Of the calculi we will consider, it is the closest one to Girard's approximation theorem, and as we will see, it is of crucial importance.

relevant in the relevant polyadic setting, we only allow contraction, which means that a given approximation may be used multiple times. This has not been studied much in the past, and we only include it for completeness' sake.

cartesian in the cartesian polyadic setting, we allow every kind of structural rule, which means that an approximations may be used any number of times, including zero. This is the situation of the classical theory of intersection types.

So, although starting from a linear perspective, polyadic approximations are general enough to encompass situations completely non-linear.

The language of approximation is a language in its own right: constructions defined routinely on calculi can be carried on it too. In particular, approximations can be typed in the most simple way. So, to a term t in the full language can be associated typing derivations $\Gamma \vdash \delta : \mathrm{A}$, where Γ is a list of type, A a type, and δ an approximation of t.

For each of these flavors, it is possible to define an approximation functor $\Lambda_{!,\star} \to \mathfrak{Dist}$ (where $\mathfrak{Dist}$ is a well-chosen cyclic **Cat**-operad) that associates (in the simplest case)

- to the linear type in $\Lambda_{!,\star}$ the set of normal types, and to the cartesian type the set of polyadic types (built from $\langle\cdot\rangle$);
- to a multi-arrow the family of set, parametrized by the types, of simply-typed approximations of the multi-arrow: for a list Γ of types and a type A, the image of the multi-arrow f is the set of type derivations of $\Gamma \vdash \delta : \mathrm{A}$ such that δ approximates t.
- to a reduction the relation of the type derivations that are rewritten one into the other by the reduction.

And the same thing can be done for $\mathsf{MELL}_{?,\star}$. These functors are depicted by the blue arrow in Figure 6.

So, the approximations are also captured by certain morphisms of operad-like structures.

Type systems

Type systems may also be encoded in this framework: (Melliès and Zeilberger 2015)'s idea is that a morphism $f : \mathscr{D} \to \mathscr{C}$ may actually be seen as a type (refinement) system for the programming language presented by $\mathscr{C}$:

- given a type c of $\mathscr{C}$, $f^{-1}(c)$ is seen as the set of types refining c; in case $\mathscr{C}$ is monochromatic/untyped, $f^{-1}(*)$ is just the set of types *tout court*;
- a multimorphism $\delta \in \mathscr{D}(\Gamma; \mathrm{A})$ such that $f(\delta) = t \in \mathscr{C}(\Xi; c)$ is seen as a type derivation that the typing $\Xi \vdash t : c$ in $\mathscr{C}$ may be refined to $\Gamma \vdash t : \mathrm{A}$ (if $\mathscr{C}$ is monochromatic, δ is just a type derivation for the untyped term t);
- a 2-arrow $\delta \Rightarrow \delta'$ is seen as a typing of its image, which is a reduction $f(\pi) \Rightarrow f(\pi')$.

In order for this morphism to be a meaningful type system, we need to require moreover that the reductions in the typed language have the same structure as the reductions in the untyped language. More specifically,

- the identity reduction is typed only by the identity;
- reductions are typed modularly: if a sequence of reductions is typable, each reduction of the sequence is typable.

Interestingly, we do not require any conditions of this kind for terms, only for reductions. We will call such morphisms of **Cat**-operads *Niefield fibrations*.

We will consider different type systems, that are depicted as red arrows in Figure 6.

From approximations to types

Approximations can be used to compute type systems, thanks to an operadic variant of the *Grothendieck construction*. The Grothendieck construction, especially the generalizations described by Bénabou to distributors, is a far-reaching generalization of the equivalence between a function from an arbitrary set to a set B and a set-valued function defined on B: indeed, to a function $f : A \to B$, we can associate a function ∂f defined on $b \in B$ as the set of pre-images of B, and to a set-valued function on $g : B \to \mathbf{Set}$, we can associate the function $\int f$, defined on the disjoint union $\sqcup_{b \in B} g(b)$, that, to an element $a \in g(b)$ associates b.

In the case we are interested in, this generalizes to the following equivalence of categories: the category of Niefield fibrations of cyclic bioperads[5] with image $\mathscr{B}$ is equivalent to the category of morphisms of cyclic bioperads from $\mathscr{B}$ to $\mathfrak{Dist}_s$.

$$\begin{tikzcd}
\mathscr{E}(F) \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & \mathfrak{Dist}_{s,*} \arrow[d] \\
\mathscr{B} \arrow[r, "F"] & \mathfrak{Dist}_s
\end{tikzcd}$$

So, every approximation morphism give rise to a type system. And indeed, the linear, affine, relevant, and cartesian approximations give rise to the type systems **LinPoly**, **AffPoly**, **RelPoly** and **CartPoly** for $\Lambda_{!,\bigstar}$ and **LinPolyLL**, **AffPolyLL**, **RelPolyLL** and **CartPolyLL** for $\mathsf{MELL}_\bigstar$.

The Grothendieck construction, by allowing to represent type systems alternatively as morphisms of **Cat**-operads to and from the calculus we are interested in, allows to transport type systems along translations of languages. Suppose we have a type system

$$\mathscr{D} \to \mathscr{B}$$

for $\mathscr{B}$. By the construction above, we can represent it faithfully as a morphism of **Cat**-operads with domain $\mathscr{B}$ and co-domain $\mathfrak{Dist}_s$, which can be pre-composed with any morphism of **Cat**-operads

$$\mathscr{C} \to \mathscr{B}$$

with codomain $\mathscr{B}$, yielding, by applying again the Grothendieck construction, a type system for $\mathscr{C}$. This is indeed how we will build intersection type systems for the variants of the λ-calculus,

[5] A slight generalization of cyclic **Cat**-operads

and more generally, for any language that can be embedded into linear logic. This has allowed us to recover known systems, and also to compute previously unknown ones, such as for $\lambda\mu$-calculus. This justifies a slogan:

Intersection type systems are simply-typed polyadic approximations

Characterizing normalisation

Computing intersection type systems, and showing in the process that they are shadows of the same polyadic approximation systems is already an interesting result, but would be disappointing if it did not allow to prove properties of these systems. The first step is to remark that subject reduction and subject expansion translate in the language we considered. Subject reduction is the property that typing is stable by reduction. Rephrased in the operadic language, it means that, if a term t is in the image of a type system functor, all its reducts are too. We can strengthen the property in a way that stays close to the actual practice by asking that any reduction from t lifts to type derivations. So, subject reduction and expansion are (op)lifting properties, and the fact that a type system enjoys these properties is closely related to weak (op)fibrations.

These fibration-like properties translate also, through the Grothendieck construction, in the language of morphisms into $\mathfrak{Dist}$. In particular, it means that we can characterize that these properties can be transferred through well-behaved translations and that the question of their validity is meaningful for any set of reductions in the calculus, allowing us to show separately that some reductions enjoy subject reduction/expansion for a certain type system, thus paving the way to prove that a certain intersection type system characterizes normalization for this type of reduction.

From approximations to semantics

The third leg of the correspondence of Figure 5 is about semantics: invariant of the reductions in a well-structured mathematical universe. Indeed, types of a type system that enjoys both subject expansion and reduction are invariant along the reduction. The equation summarizing Girard's approximation theorem suggests to define semantics of the exponentials in the following way:

- define semantics for approximations;
- compute the supremum of the semantics of the approximations of a term.

This approach fails in most cases as this supremum may not exist. Two workarounds have been proposed.

The first is due to Melliès, Tabareau, and Tasson (2009), who rephrased the question in categorical terms, where the question of computing the semantics of the exponential translates in the following way. Given a way of interpreting the linear part of linear logic, the exponential of every formula A can be interpreted as the *free commutative comonoid* over the interpretation of A. Using previous work by the first two authors (Melliès and Tabareau 2008), showed that one may proceed as follows, to compute this free commutative comonoid:

- consider the affine version of A, that is, compute the free co-pointed object $\mathrm{A}^\bullet$ on A (which is A & 1 if the category has binary products);
- compute the symmetric versions of the tensorial powers of $\mathrm{A}^\bullet$, the following equalizers, where $\mathfrak{S}_n$ is the set of canonical symmetries of $(\mathrm{A}^\bullet)^{\otimes n}$:

$$\begin{tikzcd} \mathrm{A}^{\leqslant n} \arrow[r] & (\mathrm{A}^\bullet)^{\otimes n} \arrow[loop right, "\mathfrak{S}_n"] \end{tikzcd}$$

$$\begin{tikzcd}
 & \mathbf{LinPoly} \arrow[r, hook] \arrow[d, hook] & \mathbf{AffPoly} \arrow[d, hook] & & \mathbf{AffPolyLL} \arrow[d, hook] & \mathbf{LinPolyLL} \arrow[l, hook'] \arrow[d, hook] \arrow[dll, bend right] \\
 & \mathbf{RelPoly} \arrow[r, hook] & \mathbf{CartPoly} \arrow[d, red] & \mathbf{LinPolyLL}_{?,!} \arrow[d, red] & \mathbf{CartPolyLL} \arrow[d, red] & \mathbf{RelPolyLL} \arrow[l, hook'] \\
\Lambda_{v,\star} \arrow[r, bend right, "\mathrm{CPS}"] \arrow[rr, bend left, "\mathbf{G}_v"] & \Lambda_{\star} \arrow[l, bend right] \arrow[r, "\mathbf{G}"] & \Lambda_{!,\star} \arrow[r] & \mathsf{MELL}_{?,\star} \arrow[rdd, blue] & \mathsf{MELL}_{\star} \arrow[l] & \\
\Lambda_{v} \arrow[u, red] \arrow[r, bend right, "\mathrm{CPS}"] \arrow[rr, bend right, "\mathbf{G}_v"'] & \Lambda \arrow[u, red] \arrow[l, bend right] \arrow[r, "\mathbf{G}"] & \Lambda_{!} \arrow[u, red] \arrow[r] & \mathsf{MELL}_{?} \arrow[u, red] & \mathsf{MELL} \arrow[u, red] \arrow[l] & \\
 & & \Lambda\mathrm{M}_{\star} \arrow[ruu] & & \mathfrak{Dist} \arrow[d, blue] & \\
 & & \Lambda\mathrm{M} \arrow[u, red] \arrow[ruu, "\mathbf{L}"] & & \mathscr{Rel} &
\end{tikzcd}$$

Figure 6: The different systems studied in the thesis

- compute the following projective limit, where $A^{\leqslant n} \longleftarrow A^{\leqslant n+1}$ is the canonical arrow "throwing away" one component:

$$\begin{tikzcd}
1 & A^{\leqslant 1} \arrow[l] & A^{\leqslant 2} \arrow[l] & \cdots \arrow[l] & A^{\leqslant n} \arrow[l] & \cdots \arrow[l] \\
 & & & A^{\infty} \arrow[lllu] \arrow[llu] \arrow[lu] \arrow[ru] \arrow[rru] & &
\end{tikzcd}$$

At this point, for A^{∞} to be the commutative comonoid on A it is enough that all relevant limits (the equalizers and the projective limit) commute with the tensor. Although not valid in general, this condition holds in several different models of the linear part of linear logic.

The second approach, due to Mazza (2012), is topological, and is based directly on the syntax. One considers the system **AffPoly** aforementioned. It strongly normalizes even in absence of types (the size of terms strictly decreases with reduction). The set of terms is equipped with the structure of uniform space[6], the Cauchy-completion of which, denoted by $\Lambda_{\infty}^{\mathrm{aff}}$, contains infinitary terms (allowing infinite sequences $\langle u_1, u_2, u_3, \ldots\rangle$). The original calculus embeds (and is dense) in $\Lambda_{\infty}^{\mathrm{aff}}$ by considering a finite sequence as an almost-everywhere $\bot$ sequence. Reduction is continuous and allows infinitely many substitutions to occur. This yields non-termination, in spite of the calculus

[6] The generalization of a metric space, still allowing one to speak of Cauchy sequences.

still being affine. Terms in this language are too many, and are then quotiented in order to get back the usual λ-calculus.

So, one approach, written in a categorical language, can be seen as quotienting the semantics and then completing the result. The other, in a language of approximations, consists of completing the approximants and then quotienting the result. By rephrasing the second approach in a categorical language, we draw a bridge between them, showing that both these approach consist in computing semantics from affine approximations.

From approximations back to a term

We have shown that many constructions of interest can be tackled at the level of approximations. One important question then is to understand in which case the approximations are not enough. A good way to answer this question is to understand in which cases approximations can be used to compute back the term they approximate.

In the case of the λ-calculus, the question is settled: an approximation allows to compute, in linear time, back a term if no sub-term is approximated by the empty approximant. In the more general case of linear logic proof-nets, the situation gets much more complicated. Indeed, even without considering the problem of empty approximants, some polyadic proof-nets approximates many different proof-nets. Furthermore, there is no good way to know if a set of polyadic proof-nets approximate the same proof-net.

We define and develop a geometric restriction on proof-nets, *box-connexity*, which is large enough to contain the translation of the λ-calculus, for which polyadic approximations behave well. More precisely, we show that:

- a box-connected proof-net is completely characterized by a rich enough approximant, which can moreover be chosen uniformly for all box-connected proof-nets;
- it is possible to characterize polyadic proof-nets that approximate the same box-connected proof-net.

Polyadic approximants may be considered too strong a notion of approximants. These results transfer to the more standard resource proof-nets, which is system $\mathbf{LinPolyLL}_{?,!}$ of Figure 6.

Outline

In the Chapters 1 and 2, we define the usual calculi: λ-calculus, call-by-value λ-calculus, λμ-calculus, MELL proof-nets in the not-so-standard framework of **Cat**-operads. Actually, the classical calculi do not fit in this framework, and need generalizations of it, which are defined in Chapter 2.

Having defined the calculi allow us to view translations between calculi, as well as semantics and type systems, as morphisms of operads, in Chapter 3.

Chapter 4 defines polyadic approximations for both the λ-calculus of intuitionistic linear logic and MELL proof-nets, in different approximation flavors.

Chapter 5 states and prove fundamental theorem on intersection types: they can be defined and computed from polyadic approximations, and their main properties derive from the properties of polyadic approximants.

Chapters 6 and 7 study more precisely the relation between a proof-net and an approximant and defines the notion of box-connexity. Some applications to the study of the relational model are then given.

Finally, Chapter 8 show how approximations can be used to define and understand semantics.

Contributions

The work has been done with co-authors, and some parts may be found in already published form. Chapter 8 is available as

- "A Functorial Bridge between the Infinitary Affine Lambda-Calculus and Linear Logic"
  Damiano Mazza & Luc Pellissier
  In : *12th International Colloquium on Theoretical Aspects of Computing*, Lecture Notes in Computer Science 9399, pages 144–161. Under the direction of Martin Leucker, Camilo Rueda and Frank Valencia, Springer, 2015;

the first part of Chapter 6 as

- « Computing (connected) proof-structures from their Taylor expansion »
  Giulio Guerrieri, Luc Pellissier & Lorenzo Tortora de Falco
  In : *1st International Confererence on Formal Structures for Computation and Deduction*, Leibniz International Proceedings in Informatics 52, pages 20:1–20:18. Under the direction of Delia Kesner and Brigitte Pientka, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, Dagstuhl, 2016;

and the essence of Chapter 5 is available in.

- « Approximations, Fibrations and Intersection Type Systems »
  Damiano Mazza, Luc Pellissier & Pierre Vial
  In : *Proceedings of the ACM on Programming Languages* (PACM PL), **2** (Principles of Programming Languages), pages 6:1–6:28. Under the direction of Philip Wadler and Andrew Myers, Association for Computing Machinery, New York, 2018.

Although the notions of multicategories/operads and cyclic operads are well-known, the extension of cyclic operads to the colored case, and more interestingly, to the case where some colors can only be inputs or outputs (the *semi-colored operads*) is original, as is the systematic presentation of well-known languages and their translations in that framework.

The definition of type systems in this framework is directly adapted from (Melliès and Zeilberger 2015). The application of the Grothendieck construction, allowing to equate intersection type derivations and simply-typed linear approximations is new.

Linear approximations have been studied extensively since Girard's aforementioned approximation theorem. Indeed, Kfoury (2000) has defined a linearization of the λ-calculus and studied the relationship between it and the usual calculus. In the context of game semantics, so does Melliès (2004). Finally, the resource calculus of Boudol (1993), which can be used as a target language of Ehrhard and Regnier (2008)'s Taylor expansion will be studied more directly. An independent work, that uses a closely related language is Tsukada, Asada, and Ong (2017)'s. Up to our knowledge, our definition of the Taylor expansion of MELL proof-nets as a pullback is original.

The definition and developments around box-connexity both for polyadic proof-nets and for the relational model, have been introduced in this work.

Finally, the adaptation of Mazza (2012) to a categorical language and the ensuing bridge is also novel work.

1

Calculi as operads

1.1 Structures for intuitionnistic reasonning

The Lambek tradition

1.1.1 Following a tradition starting back with Lambek (1969), it is customary to interpret types as objects in a category and terms (in normal form) as arrows between them. Recall, first, the definition of a category:

Definition 1 (*category*)

A **category** $\mathscr{C}$ is the data of:

- a class C of *objects*;
- for all couples (A, B) of objects, a class

$$\mathscr{C}(A; B)$$

of morphisms from A to B. A morphism $f \in \mathscr{C}(A; B)$ may be represented as an arrow $f : A \to B$ or an arrow

$$\begin{tikzcd} A \arrow[r, "f"] & B \end{tikzcd}$$

- for every object A, a distinguished morphism

$$\mathrm{id}_A \in \mathscr{C}(A; A)$$

- a family of operations

$$\begin{aligned} \circ_{A,B,C} : \mathscr{C}(B; C) \times \mathscr{C}(A; B) &\to \mathscr{C}(A; C) \\ (f, g) &\mapsto f \circ g \end{aligned}$$

  indexed on triples of objects of $\mathscr{C}$ (we will always omit the subscripts) verifying

$$\forall (A, B, C, D) \in C^4, \forall f : A \to B, \forall g : B \to C, \forall h : C \to D, (h \circ g) \circ f = h \circ (g \circ f)$$

  and

$$\forall (A, B) \in C, \forall f : A \to B, \mathrm{id}_B \circ f = f = f \circ \mathrm{id}_A$$

If $\mathscr{C}$ is a category, we will write

$$A : \mathscr{C}$$

to mean that A is an object in $\mathscr{C}$ and

$$f : A \to B :: \mathscr{C}$$

to mean that $f \in \mathscr{C}(A; B)$. We will often consider that this notation defines A, B and f in the same move, if A and B are not already defined.

Remark 1

We will not be really interested in size issues. A distinction can be made between "small" categories, whose underlying class of object and all the classes of morphisms are sets, and "big" categories, that feature at least a proper class.

This distinction can be achieved by either working in an extension of Zermelo–Fraenkel set theory rich enough to accomodate proper classes, such as Von Neumann–Bernays–Gödel's, or through Grothendieck unverses. The reader can choose between the two according to their taste.

We will often omit to mention whether categories (or other structures defined later) are small or big.

Definition 2 (*functor*)

Let $\mathscr{C}$ and $\mathscr{D}$ be two categories. A functor

$$F : \mathscr{C} \to \mathscr{D}$$

is a function (also written F) from objects of $\mathscr{C}$ to objects of $\mathscr{D}$ and, for every objects $A : \mathscr{C}$ and $B : \mathscr{C}$, a function (also noted F)

$$F : \mathscr{C}(A; B) \to \mathscr{D}(FA; FB)$$

such that,

$$\forall f : \mathrm{A} \to \mathrm{B} :: \mathscr{C}, \forall g : \mathrm{B} \to \mathrm{C} :: \mathscr{C}, \mathrm{F}(g \circ f) = \mathrm{F}g \circ \mathrm{F}f$$

and

$$\forall \mathrm{A} : \mathscr{C}, \mathrm{F}(\mathrm{id}_\mathrm{A}) = \mathrm{id}_{\mathrm{FA}}$$

1.1.2 The structure of category is extremely versatile. The philosophy underlying its interest is that all mathematical structures ought to be understood, not *per se*, but by the way they are preserved or twisted.

Example : These examples will be used throughout the text:

sets and functions Let **Set** be the (big) category whose objects are sets, morphisms are functions, and composition the usual composition of functions.

categories and functors Let **Cat** be the (big) category whose objects are (small) categories, morphisms are functors, and composition the composition of functors.

categories and functor, huge edition Let **CAT** be the (huge) category whose objects are (big and small) categories and morphisms are functors. This construction can be iterated as long as our ambient theory is reach enough to handle bigger and bigger classes.

sets and relations Let **Rel** be the (big) category whose objects are sets, morphisms are relations, and composition the usual composition of relations: for $f : \mathrm{A} \to \mathrm{B} :: \mathbf{Rel}$ and $g : \mathrm{B} \to \mathrm{C} :: \mathbf{Rel}$,

$$g \circ f = \left\{(a,c) \in \mathrm{A} \times \mathrm{C} \mid \exists b \in \mathrm{B}, (a,b) \in f, (b,c) \in g\right\}.$$

discrete categories Every class S defines a category $\mathscr{S}$ in the following fashion: the underlying class of objects of $\mathscr{S}$ is S and the only morphisms in $\mathscr{S}$ are the identities. Such a category is called *discrete*. We will often view sets as discrete categories, when needed.

Remark 2

Although conventional, the names of categories we just presented is not completely coherent. The categories **Set** and **Cat** are named after their objects, while **Rel** is named after its morphisms.

The general philosophy of category theory is that all the relevant structural information is captured by the morphisms. It would thus be brave and coherent to rename every category after its morphisms.

1.1.3 A generalization of categories will come out handy later: categories whose objects do not necessarily have identities.

Definition 3 (*semi-category*)

A **semi-category** $\mathscr{C}$ is the data of:

- a class C of *objects*;
- for all couples (A, B) of objects, a class

$$\mathscr{C}(\mathrm{A};\mathrm{B})$$

  of morphisms from A to B. A morphism $f \in \mathscr{C}(\mathrm{A};\mathrm{B})$ may be represented as an arrow $f : \mathrm{A} \to \mathrm{B}$ or an arrow

$$\mathrm{A} \xrightarrow{f} \mathrm{B}$$

- a family of operations

$$\begin{aligned}\circ_{\mathrm{A,B,C}} : \mathscr{C}(\mathrm{B};\mathrm{C}) \times \mathscr{C}(\mathrm{A};\mathrm{B}) &\to \mathscr{C}(\mathrm{A};\mathrm{C}) \\ (f,g) &\mapsto f \circ g\end{aligned}$$

indexed on triples of objects of $\mathscr{C}$ (we will always omit the subscripts) verifying

$$\forall(\mathrm{A},\mathrm{B},\mathrm{C},\mathrm{D}) \in \mathrm{C}^4, \forall f : \mathrm{A} \to \mathrm{B}, \forall g : \mathrm{B} \to \mathrm{C}, \forall h : \mathrm{C} \to \mathrm{D}, (h \circ g) \circ f = h \circ (g \circ f)$$

A morphism $h \in \mathscr{C}(\mathrm{A};\mathrm{A})$ such that

$$\forall(\mathrm{A},\mathrm{B}) \in \mathrm{C}, \forall f : \mathrm{A} \to \mathrm{B}, f = f \circ \mathrm{id}_{\mathrm{A}}$$

and

$$\forall(\mathrm{A},\mathrm{B}) \in \mathrm{C}, \forall f : \mathrm{B} \to \mathrm{A}, \mathrm{id}_{\mathrm{B}} \circ f = f$$

is called the *identity* of A.

The definition of *functor of semi-categories* is the same as functor of categories.

Our basic object: **Cat**-*operads*

1.1.4 We depart from this traditional presentation of calculi as categories for the following reason:

- we find more natural to consider programs with possibly more than one free variable. Although the many-variable case can be mimicked by one free variable using a technical trick called "currification", and that all the definitions are leaner in this case, we will soon need the ability to treat each variable according to different modalities. As such, we will move from categories to *multicategories*. This is directly inspired from (Hyland 2017);
- we are interested not only in the static behaviour of programs but also in the reduction between them. As such, we need to add a second dimension, arrows between arrows, as presented in (Seely 1987).

Definition 4 (Cat-*operad*)

A **Cat**-*multicategory* $\mathscr{C}$ is the data of:

- a class C of *objects*;
- for each integer $n \in \mathbf{N}$ and all tuples $(c_1, \dots, c_n, c) \in \mathrm{C}^{n+1}$, a category

$$\mathscr{C}(c_1, \dots, c_n; c)$$

whose objects are *multi-morphisms* (or *multi-arrows*) from $(c_1, \dots, c_n)$ to c. Such a multi-arrow is written $f : c_1, \dots, c_n \to c$. Morphisms of $\mathscr{C}(c_1, \dots, c_n; c)$ are called 2-arrows;
- for each object $c \in \mathrm{C}$, a distinguished multimorphism

$$\mathrm{id}_c : c \to c,$$

the *identity* of c;

- for every tuple

$$(c_1, \dots, c_n, c),$$

together with n other tuples

$$(d_1^1, \dots, d_1^{k_1}, c_1), \dots, (d_n^1, \dots, d_n^{k_n}, c_n)$$

a morphism

$$\circ : \mathscr{C}(c_1, \dots, c_n; c) \times \mathscr{C}(d_1^1, \dots, d_1^{k_1}; c_1) \times \cdots \times \mathscr{C}(d_n^1, \dots, d_n^{k_n}; c_n) \to \mathscr{C}(d_1^1, \dots, d_n^{k_n}; c)$$

such that composition is associative:

$$\begin{aligned} &\theta \circ \left(\theta_1 \circ (\theta_1^1, \dots, \theta_1^{n_1}), \dots, \theta_k \circ (\theta_k^1, \dots, \theta_k^{n_k}\right) \\ =&\theta \circ (\theta_1, \dots, \theta_k) \circ (\theta_1^1, \dots, \theta_k^{n_k}) \end{aligned}$$

(whenever the mulimorphisms are composable) and unital with respect to the identities:

$$\forall \theta : c_1, \dots, c_n \to c, \qquad \theta \circ (\mathrm{id}_{c_1}, \dots, \mathrm{id}_{c_n}) = \theta = \mathrm{id}_c \circ \theta.$$

A **Cat-operad** $\mathscr{O}$ is a **Cat**-multicategory endowed with, for all object A, integer n, permutation $\sigma \in \mathfrak{S}_n$, and objects $\Gamma = A_1 \dots A_n$, a functor

$$\mathsf{exch}_\sigma^{\Gamma;A} : \mathscr{O}(A_1, \dots, A_n; A) \to \mathscr{O}(A_{\sigma^{-1}(1)}, \dots, A_{\sigma^{-1}(n)}; A)$$

that satisfy compatibility laws: the action is compatible with the composition in $\mathfrak{S}_n$:

$$\begin{gathered} \forall \sigma, \sigma' \in \mathfrak{S}_n, \mathsf{exch}_{\sigma'}^{\sigma^{-1}\Gamma;A} \circ \mathsf{exch}_\sigma^{\Gamma;A} = \mathsf{exch}_{\sigma' \circ \sigma}^{\Gamma;A} \\ \mathsf{exch}_{\mathrm{id}}^{\Gamma;A} = \mathrm{id} \end{gathered}$$

and with the composition in $\mathscr{O}$.

Remark 3

We have to impose the reader a short terminological note, as the name operad is used here in a slightly non-standard way.

Multicategories have been introduced by Lambek (1969), in the context of logics and linguistics. Multicategories are in general not supposed symmetrical, and are thought of as generalization of categories, accounting for multilinear maps.

Operads have been defined in the field of homotopy theory, by May (1972). They are thought of as abstract operations, are often symmetric, and have only one object.

The case of an operad with multiple objects is studied under the name of *colored operads* and the objects are dubbed *colors*.

So *symmetric multicategories* and *coloured operads* are the same objects. We have decided here to simplify the terminology, by calling them simply operads.

Example : We first give ways to see categories as operads.

- The first example is the most pedestrian: every category $\mathscr{C}$ defines an operad $\mathscr{O}$ with the same set of objects, morphisms of $\mathscr{C}$ as unary multimorphisms of $\mathscr{O}$ (and $\mathscr{O}$ has no other multimorphisms), and the categories of 2-arrows as trivial;
- Every strict monoidal category defines an operad with the same objects, and arrows $c_1 \otimes \cdots \otimes c_n \to c$ as multiarrows $c_1, \ldots, c_n \to c$.

Example: Not all multicategories arise from monoidal categories. For instance, let V be any symetric monoidal category and let Ab(V) be the category of abelian groups in V. It is a multicategory, but not a monoidal category in general.

It has been remarked from the beginning of category theory that the objects play a very dim role, and can actually be forgotten and represented by a special class of arrows: the identities. Much the same can be done in higher-dimensional structures. In our case, it means that the operads we will define in the following are entirely characterized by their 2-arrows.

In other words, we will give a presentation of the λ-calculus and the linear logic proof-nets by the reductions, and not by the terms/nets themselves.

1.1.5 This language allows to express properties of the reduction, such as normalization:

Definition 5 (*normal, weakly and strongly normalizable multimorphism*)

Let $\mathscr{C}$ be a **Cat**-operad and let M be a multimorphism of $\mathscr{C}$. We say that M is:

- *normal* if there is no non-identity 2-arrow $\mathrm{M} \Rightarrow \mathrm{M}'$ in $\mathscr{C}$;
- *weakly normalizable* if there is a 2-arrow $\mathrm{M} \Rightarrow \mathrm{N}$ in $\mathscr{C}$ with N normal;
- *strongly normalizable* if there is no sequence $(\psi_i)_{i \in \mathbf{N}}$ of non-identity 2-arrows $\psi_i : \mathrm{M}_i \Rightarrow \mathrm{M}_{i+1}$ in $\mathscr{C}$ with $\mathrm{M}_0 = \mathrm{M}$.

Note that, when $\mathscr{C}$ is a **Cat**-operad presenting a term calculus (multimorphisms are terms, 2-arrows are reduction sequences), then the above definition matches the standard terminology.

1.2 The λ-calculus as an operad

Typed, untyped and unityped

1.2.1 The λ-calculus is often defined in two fashions: either typed or untyped. Before discussing this distinction, we will first clarify what we mean by a type. It will then become clear that we ought to treat untyped λ-calculus as a special case of simply-typed λ-calculus.

One of themselves, even a prophet of their own, said, the Cretians are alway liars, evil beasts, slow bellies. Tit, **1**, 12

1.2.2 Types are usually understood in one of the two following fashion: in the intrinsic view of typing, also known as types *à la Church*, every well-formed term has a uniquely associated type: indeed, the basic grammar of expressions uses the types to guarantee the well-formation, and thus making certain expressions, such as Epimenides' paradox, unwritable. Untyped terms have no meaning with these spectacles. On the other side, the extrinsic view of typing, of so-called types *à la Curry*, sees types as expressions of properties of terms, which have a meaning independently of these types. A term can have many different types, or no type at all.

Pragmatically, while a Church type is embedded in a term, determining the Curry type of a term can be computationally expensive, or even undecidable.

1.2.3 As our main viewpoint on the languages is categorical, that is, through the pragmatics of composition and chaining, we naturally favor Church typing, and interpret terms as arrows in an operad (and as such, have sources and a target, defined at the same time that the term). We will then define first simply-typed λ-calculus, parametrized by a set of types, and consider the untyped λ-calculus as the special case of the *unityped* λ-calculus, that is, simply typed λ-calculus, typed with a single type.

The Λ *operad*

1.2.4 We let

$$\mathbb{V} = \{x_1, x_2, \dots, x_n, \dots\}$$

be an infinite countable set of *variables*. We suppose given a set $\mathbb{T}$ closed by a binary operation → and disjoint from $\mathbb{V}$, that will be a parameter of all our definitions. We will soon see that this parameter is functorial.

1.2.5 Following the philosophy we highlighted, we will directly define reductions, inductively. As such, we will define sequents of the form

$$\Gamma \vdash \theta : t \Rightarrow t' : \mathrm{B}$$

where Γ is a list of the form $x_1 : \mathrm{A}_1, \dots, x_n : \mathrm{A}_n$, the *context*, θ is the reduction, t and t' are two *terms* and B is a type. The rules of reduction formation in Figure 1.1, where $\cdot\{\cdot \leftarrow \cdot\}$ is the substitution operation, that we define now. We first define *free* and *bounded* variables of a term by:

Definition 6 (*free variables, bounded variables*)

The sets of *free variables* $\mathrm{fv}(t)$ and of *bounded variables* $\mathrm{bv}(t)$ of a term t are inductively defined by:

$$\begin{aligned} \mathrm{fv}(x) &= \{x\} & \mathrm{bv}(x) &= \varnothing \\ \mathrm{fv}(uv) &= \mathrm{fv}(u) \cup \mathrm{fv}(v) & \mathrm{bv}(uv) &= \mathrm{bv}(u) \cup \mathrm{bv}(v) \\ \mathrm{fv}(\lambda x.u) &= \mathrm{fv}(u) \setminus \{x\} & \mathrm{bv}(\lambda x.u) &= \mathrm{bv}(u) \cup \{x\} \\ \mathrm{fv}(\beta x.u)v &= \mathrm{fv}(u) \setminus \{x\} \cup \mathrm{fv}(v) & \mathrm{bv}(\beta x.u) &= \mathrm{bv}(u) \cup \{x\} \cup \mathrm{bv}(v) \end{aligned}$$

We are ready to define the substitution operation $\cdot\{\cdot \leftarrow \cdot\}$:

Definition 7 (*capture-free substitution*)

Let u and v be two terms, such that $\mathrm{fv}(v) \cap \mathrm{bv}(u) = \varnothing$. Let x be a variable not bounded in u. The substitution of x by v in u is defined, by induction on the structure of u, by:

$$\begin{aligned} x\{x \leftarrow v\} &= v \\ y\{x \leftarrow v\} &= y \\ u_1u_2\{x \leftarrow v\} &= u_1\{x \leftarrow v\}\, u_2\{x \leftarrow v\} \\ (\lambda y.u)\{x \leftarrow v\} &= \big(\lambda y.u\{x \leftarrow v\}\big) \\ (\beta y.u)\{x \leftarrow v\} &= \big(\beta y.u\{x \leftarrow v\}\big) \end{aligned}$$

The free variables of $u\{x \leftarrow v\}$ are

$$\mathrm{fv}(u\{x \leftarrow v\}) = \mathrm{fv}(u) \setminus \{x\} \cup \mathrm{fv}(v)$$

$$\frac{}{\Gamma, x : \mathrm{A} \vdash x : x \Rightarrow x : \mathrm{A}}\,(\text{variable})$$

$$\frac{\Gamma \vdash \theta : t \Rightarrow t' : \mathrm{A} \qquad \Gamma \vdash \theta' : t' \Rightarrow t'' : \mathrm{A}}{\Gamma \vdash \theta ; \theta' : t \Rightarrow t'' : \mathrm{A}}\,(\text{composition})$$

$$\frac{\Gamma \vdash \theta : t \Rightarrow t' : \mathrm{A} \to \mathrm{B} \qquad \Gamma \vdash \kappa : u \Rightarrow u' : \mathrm{A}}{\Gamma \vdash \theta\kappa : tu \Rightarrow t'u' : \mathrm{B}}\,(\text{application})$$

$$\frac{\Gamma, x : \mathrm{A} \vdash \theta : t \Rightarrow t' : \mathrm{B}}{\Gamma \vdash \lambda x.\theta : \lambda x.t \Rightarrow \lambda x.t' : \mathrm{A} \to \mathrm{B}}\,(\text{abstraction})$$

$$\frac{\Gamma, x : \mathrm{A} \vdash \theta : t \Rightarrow t' : \mathrm{B} \qquad \Gamma \vdash \kappa : u \Rightarrow u' : \mathrm{A}}{\Gamma \vdash (\beta x.\theta)\kappa : (\lambda x.t)u \Rightarrow t'\,\{x \leftarrow u'\} : \mathrm{B}}\,(\beta\text{-reduction})$$

Figure 1.1: The λ-calculus reductions

In general, we will not redefine capture-free substitution for each language we will define in the remainder of this thesis, but only specify which are the binders.

1.2.6 The reduction terms must consider reduction sequences modulo permutation equivalence. Consider the case of a term $\mathrm{M} : \star \to \star$ with one free variable, the substitution of which by a term $\mathrm{N} : \star^n \to \star$ we denote by $\mathrm{M}\{\mathrm{N}\}$. Let $\mathrm{M} \xrightarrow{\rho} \mathrm{M}' \xrightarrow{\rho'} \mathrm{M}''$ and $\mathrm{N} \xrightarrow{\tau} \mathrm{N}' \xrightarrow{\tau'} \mathrm{N}''$ be reduction sequences.

Consider the two reductions $(\rho' ; \rho) \circ (\tau' ; \tau)$ and $(\rho' \circ \tau') ; (\rho \circ \tau)$, which are both from $\mathrm{M}\{\mathrm{N}\}$ to $\mathrm{M}''\{\mathrm{N}''\}$ and which, by functoriality, must be equal (in order for the structure to be a **Cat**-operad). These will form the top and bottom paths in the diagram

$$\begin{tikzcd}
\mathrm{M}\{\mathrm{N}\} \arrow[r, "\tau"] & \mathrm{M}\{\mathrm{N}'\} \arrow[r, "\tau'"] \arrow[d, "\rho"] & \mathrm{M}\{\mathrm{N}''\} \arrow[d, "\rho"] & \\
 & \mathrm{M}'\{\mathrm{N}'\} \arrow[r, "\tau'"] & \mathrm{M}'\{\mathrm{N}''\} \arrow[r, "\rho'"] & \mathrm{M}''\{\mathrm{N}''\}
\end{tikzcd}$$

Hence, we need to ask that the square in the middle commutes, which is ensured by permutation equivalence, the equations of Figure 1.2, where $\theta_{t \Rightarrow t'}$ is an abbreviation for

$$\Gamma \vdash \theta : t \Rightarrow t' : \mathrm{A}$$

for some context Γ and type A.

Remark 4

> The equations could be oriented, from left to right, to define a standardisation rewriting system. It would amount to add a dimension to our operad structure, making it the following ladder:

1. types,
2. terms,
3. reductions,
4. standardizing rewritings.

We leave the study of this structure for an other time.

$$\begin{aligned}(\theta_{t\Rightarrow t'}\kappa_{u\Rightarrow u'})_{tu\Rightarrow t'u'};(\theta'_{t'\Rightarrow t''}\kappa'_{u'\Rightarrow u''})_{t'u'\Rightarrow t''u''} &= (\theta_{t\Rightarrow t'};\theta'_{t'\Rightarrow t''})_{t\Rightarrow t''}(\kappa_{u\Rightarrow u'};\kappa'_{u'\Rightarrow u''})_{u\Rightarrow u''}\\ (\lambda x.\theta_{t\Rightarrow t'});(\lambda x.\theta'_{t'\Rightarrow t''}) &= \lambda x.(\theta;\theta')_{t\Rightarrow t''}\\ (\lambda x.\theta_{t\Rightarrow t'})\kappa_{u\Rightarrow u'};(\beta x.\theta'_{t\Rightarrow t'})\kappa'_{u'\Rightarrow u''} &= (\beta x.(\theta;\theta')_{t\Rightarrow t''})(\kappa;\kappa')_{u\Rightarrow u''}\end{aligned}$$

Figure 1.2: The λ-calculus reduction equations.

Lemma 1

Let $\Gamma \vdash t : t \Rightarrow t : \mathrm{B}$ *be a* β*-less term. For all* $\Gamma \vdash \theta_1 : t' \Rightarrow t : \mathrm{B}$ *and* $\Gamma \vdash \theta_2 : t \Rightarrow t' : \mathrm{B}$, $t;\theta_1 = \theta_1$ *and* $\theta_2;t = \theta_2$.

1.2.7 We can then define λ-*terms* as terms without β of this calculus, and remark that they have the same source and target, which has the same syntax of the terms themselves.

Example : Anticipating §1.2.10, in the unityped setting (that is, with only one type ★ satisfying ★ → ★ = ★)

$$\begin{aligned}&c:\star,d:\star \vdash (\beta b.\lambda x.\lambda y.((\beta z.bz)x)y)((\beta a.acd)(\lambda e.\lambda f.e))\\ &\qquad : (\lambda b.\lambda x.\lambda y.((\beta z.bz)x)y)((\lambda a.acd)(\lambda e.\lambda f.e)) \Rightarrow \lambda x.\lambda y.(\lambda e.\lambda f.e)cdxy : \star,\star \to \star\end{aligned}$$

is a reduction of the calculus. It can be composed with

$$\begin{aligned}&c:\star,d:\star \vdash \lambda x.\lambda y.(\beta e.\lambda f.e)cdxy\\ &\qquad : \lambda x.\lambda y.(\lambda e.\lambda f.e)cdxy \Rightarrow \lambda x.\lambda y.(\lambda f.c)dxy : \star,\star \to \star\end{aligned}$$

and

$$\begin{aligned}&c:\star,d:\star \vdash \lambda x.\lambda y.(\beta f.c)dxy\\ &\qquad : \lambda x.\lambda y.(\lambda f.c)dxy \Rightarrow \lambda x.\lambda y.cxy : \star,\star \to \star\end{aligned}$$

making a reduction from $(\beta b.\lambda x.\lambda y.((\beta z.bz)x)y)((\beta a.acd)(\lambda e.\lambda f.e))$ to $\lambda x.\lambda y.cxy$ in the context c,d.

Definition 8

The **Cat**-operad $\Lambda_{\mathbb{T}}$ is defined as the operad with:

- as objects the elements of $\mathbb{T}$;
- as arrows

$$\mathrm{A}_1, \dots, \mathrm{A}_n \longrightarrow \mathrm{B}$$

β-less terms of Figure 1.1, quotiented by the equations of Figure 1.2

$$x_1 : \mathrm{A}_1, \dots, x_n : \mathrm{A}_n \vdash t : t \Rightarrow t : \mathrm{B}$$

that is, λ-terms t with free variables included in $\{x_1, \dots, x_n\}$.

- as compositions the substitution and renaming of variables. That is, let

$$x_1 : \mathrm{A}_1, \dots, x_i : \mathrm{A}_i, \dots x_n : \mathrm{A}_n \vdash t : t \Rightarrow t : \mathrm{B}$$

and, for all $1 \leqslant i \leqslant n$,

$$x^i_1 : \mathrm{A}^i{}_1, \dots, x^i_{n_i} : \mathrm{A}^i{}_{n_i} \vdash u_i : u_i \Rightarrow u_i : \mathrm{A}_i$$

be λ-terms. Their composite $t \circ (u_1, \dots, u_n)$ is the substitution of the λ-term

$$x^1_1 : \mathrm{A}^1{}_1, \dots, x^1_{n_1} : \mathrm{A}^1{}_{n_1}, x^2_1 : \mathrm{A}^2{}_1, \dots, x^n_{n_n} : \mathrm{A}^n{}_{n_n} \vdash t\{x_i \leftarrow u_i\} : t\{x_i \leftarrow u_i\} \Rightarrow t\{x_i \leftarrow u_i\} : \mathrm{B}$$

with the free variables renamed from x_1 to $x_{\sum_{i=1}^n n_i}$.

- as 2-arrows between two terms

$$x_1 : \mathrm{A}_1, \dots, x_i : \mathrm{A}_i, \dots x_n : \mathrm{A}_n \vdash t : t \Rightarrow t : \mathrm{B}$$

and

$$x_1 : \mathrm{A}_1, \dots, x_i : \mathrm{A}_i, \dots x_n : \mathrm{A}_n \vdash t' : t' \Rightarrow t' : \mathrm{B}$$

all the reductions

$$x_1 : \mathrm{A}_1, \dots, x_i : \mathrm{A}_i, \dots x_n : \mathrm{A}_n \vdash \theta : t \Rightarrow t' : \mathrm{B}$$

- composition ; as composition of 2-arrows.

So, the operad $\Lambda_{\mathbb{T}}$ is the operad of all λ-reductions, structured in the three layers presented earlier.

1.2.8 Let us consider the category **BinOp** whose objects are couples $(\mathrm{X}, \odot)$ of sets endowed with a binary operation

$$\odot : \mathrm{X} \times \mathrm{X} \to \mathrm{X}$$

and morphisms $f : (\mathrm{X}, \odot) \to (\mathrm{Y}, \odot)$ functions $f : \mathrm{X} \to \mathrm{Y}$ such that,

$$\forall x, x' \in \mathrm{X}, f(x \odot x') = f(x) \odot f(x').$$

Theorem 2

The function

$$\begin{aligned} \mathbf{BinOp} &\to \mathbf{Op} \\ \mathbb{T} &\mapsto \Lambda_{\mathbb{T}} \end{aligned}$$

extends to a functor.

This functor maps a set $(\mathrm{X}, \odot)$ to an operad Λ_{X}: it views the set X as the set of types and the binary operation $\odot$ as the type constructor $\to$.

Extensionality

1.2.9 The $\Lambda_{\mathbb{T}}$ operad we just defined lacks extensionality properties. We may enrich our calculus with a new binder η and the new formation rule of Figure 1.3. This defines an operad $\Lambda^{\eta}_{\mathbb{T}}$.

$$\frac{\Gamma \vdash \theta : t \Rightarrow t' : \mathrm{A} \to \mathrm{B}}{\Gamma \vdash (\eta x.\theta) : t \Rightarrow \lambda x.(t'x) : \mathrm{A} \to \mathrm{B}}(\eta\text{-expansion})$$

Figure 1.3: The λ-calculus η-expansion

We will not be very interested in extensionality, apart as a sanity check: we ought to use only operations that do not break it. For instance, in Chapter 3 we will consider morphisms of **Cat**-operads, understood as translations of one language into antoher. We will only consider morphisms that can be pass the extensionality test: if $f : \Lambda \to \mathscr{C}$ is a morphism of **Cat**-operads, and f does not lift to a morphism of **Cat**-operads $\Lambda^{\eta} \to \mathscr{C}$ (that is, there exists a 2-arrow $t_1 \Rightarrow t_2$ in Λ^{η} such that there are no 2-arrows $f(t_1) \Rightarrow f(t_2)$ in $\mathscr{C}$) we will not consider this morphism to be a suitable translation.

The unityped and simply-typed case

$$\frac{}{\Gamma, x \vdash \mathrm{id} : x \Rightarrow x}(\text{variable})$$

$$\frac{\Gamma \vdash \theta : t \Rightarrow t' \qquad \Gamma \vdash \theta' : t' \Rightarrow t''}{\Gamma \vdash \theta ; \theta' : t \Rightarrow t''}(\text{composition})$$

$$\frac{\Gamma \vdash \theta : t \Rightarrow t' \qquad \Gamma \vdash \kappa : u \Rightarrow u'}{\Gamma \vdash \theta\kappa : tu \Rightarrow t'u'}(\text{application})$$

$$\frac{\Gamma, x \vdash \theta : t \Rightarrow t'}{\Gamma \vdash \lambda x.\theta : \lambda x.t \Rightarrow \lambda x.t'}(\text{abstraction})$$

$$\frac{\Gamma, x \vdash \theta : t \Rightarrow t' \qquad \Gamma \vdash \kappa : u \Rightarrow u'}{\Gamma \vdash (\beta x.\theta)\kappa : (\lambda x.t)u \Rightarrow t' \{x \leftarrow u'\}}(\beta\text{-reduction})$$

Figure 1.4: The unityped λ-calculus reductions

1.2.10 The category **BinOp** has a terminal object: the singleton set $(\{\star\}, \rightarrow)$ such that $\star \rightarrow \star = \star$. The corresponding $\Lambda_\star$ operad corresponds to the unityped λ-calculus. As it is of paramount importance, we give an explicit description of it:

- $\Lambda_\star$ only has one object: $\star$;
- a multimorphism $\star^n \rightarrow \star$ is an id reduction built from the rules of Figure 1.4, modulo the equations of Figure 1.2.
- a 2-morphism is a general term built from the rules of Figure 1.4, modulo the equations of Figure 1.2.

$\Lambda_\star$ and $\Lambda^\eta_\star$ both have infinite reduction sequences.

1.2.11 Dually, we call *simply-typed* λ-calculus the **Cat**-operad $\Lambda_\rightarrow$ corresponding to the set[1] freely generated by an infinite countable set of atoms. $\Lambda_\rightarrow$ and $\Lambda^\eta_\rightarrow$ both only have finite reduction sequences.

1.3 The call-by-value λ-calculus as an operad

The *call-by-value* λ-calculus, introduced by Plotkin (1975), to formalize the evaluation strategy in ISWIM as given by the SECD machine, restricts the β-reduction of the usual λ-calculus to a particular class of arguments, the *values*. It too, can be described in an operadic fashion. We suppose that the set $\mathbb{T}$ is still closed by a binary operation $\rightarrow$, but that it is actually the reunion of two disjoint subsets, of *value types* and *term types*, exchanged by a fix-point-free involution. We will write the all types with roman capital letters $\mathrm{A}, \mathrm{B}, \ldots$.

The sequents will be of two forms:

$$\Gamma \vdash_{\mathsf{v}} \theta : t \Rightarrow t' : \mathrm{B}$$

$$\Gamma \vdash_{\mathsf{t}} \theta : t \Rightarrow t' : \mathrm{B}$$

The first kind meaning, that, under the context Γ (containing only value types), θ is a reduction from t to t', of value type B, the second that, under the context Γ, that also contains only value types, θ is a reduction from t to t', of term type B. The rules of reduction formation are given Figure 1.5, modulo the equation of Figure 1.6. We can remark that the reductions for values are all identities.

Definition 9

The **Cat**-operad $\Lambda_{v,\mathbb{T}}$ is defined as the operad with:

- as objects the elements of $\mathbb{T}$;
- as arrows

$$\begin{tikzcd} \Gamma \arrow[r] & \mathrm{B} \end{tikzcd}$$

  where

  - $\Gamma = \mathrm{A}_1, \ldots, \mathrm{A}_n$ are value types and B a term type, β_v-less terms of Figure 1.5, quotiented by the equations of Figure 1.6

$$x_1 : \mathrm{A}_1, \ldots, x_n : \mathrm{A}_n \vdash_{\mathsf{t}} t : t \Rightarrow t : \mathrm{B}$$

  that is, λ-terms t with free variables included in $\{x_1, \ldots, x_n\}$.

[1] They are all isomorphic.

$$\frac{}{\Gamma, x : \mathrm{A} \vdash_{\mathsf{v}} x : x \Rightarrow x : \mathrm{A}}\,\text{(variable)}$$

$$\frac{\Gamma, x : \mathrm{A} \vdash_{\mathsf{t}} t : t \Rightarrow t : \mathrm{B}}{\Gamma \vdash_{\mathsf{v}} \lambda x.t : \lambda x.t \Rightarrow \lambda x.t : \mathrm{A} \to \mathrm{B}}\,\text{(abstraction)}$$

$$\frac{\Gamma \vdash_{\mathsf{t}} \theta : t \Rightarrow t' : \mathrm{A} \qquad \Gamma \vdash_{\mathsf{t}} \theta' : t' \Rightarrow t'' : \mathrm{A}}{\Gamma \vdash_{\mathsf{t}} \theta; \theta' : t \Rightarrow t'' : \mathrm{A}}\,\text{(composition)}$$

$$\frac{\Gamma \vdash_{\mathsf{t}} \theta : t \Rightarrow t' : \mathrm{A} \to \mathrm{B} \qquad \Gamma \vdash_{\mathsf{t}} \kappa : u \Rightarrow u' : \mathrm{A}}{\Gamma \vdash_{\mathsf{t}} \theta\kappa : tu \Rightarrow t'u' : \mathrm{B}}\,\text{(application)}$$

$$\frac{\Gamma, \vdash_{\mathsf{v}} v : v \Rightarrow v : \mathrm{A}}{\Gamma \vdash_{\mathsf{t}} v : v \Rightarrow v : \mathrm{A}}\,\text{(values)}$$

$$\frac{\Gamma, x : \mathrm{A} \vdash_{\mathsf{t}} t : t \Rightarrow t : \mathrm{B} \qquad \Gamma \vdash_{\mathsf{v}} u : u \Rightarrow u : \mathrm{A}}{\Gamma \vdash_{\mathsf{t}} (\beta_v x.t)u : (\lambda x.t)u \Rightarrow t\{x \leftarrow u\} : \mathrm{B}}\,(\beta_v\text{-reduction})$$

Figure 1.5: The λ_v-calculus reductions

$$(\theta_{t \Rightarrow t'} \kappa_{u \Rightarrow u'})_{tu \Rightarrow t'u'}; (\theta'_{t' \Rightarrow t''} \kappa'_{u' \Rightarrow u''})_{t'u' \Rightarrow t''u''} = (\theta_{t \Rightarrow t'}; \theta'_{t' \Rightarrow t''})_{t \Rightarrow t''} (\kappa_{u \Rightarrow u'}; \kappa'_{u' \Rightarrow u''})_{u \Rightarrow u''}$$

Figure 1.6: The λ_v-calculus reduction equations.

  - $\Gamma = \mathrm{A}_1, \dots, \mathrm{A}_n$ are value types and B a value type, β_v-less terms of Figure 1.5, quotiented by the equations of Figure 1.6

    $$x_1 : \mathrm{A}_1, \dots, x_n : \mathrm{A}_n \vdash_{\mathsf{v}} v : v \Rightarrow v : \mathrm{B}$$

    that is, λ-values v with free variables included in $\{x_1, \dots, x_n\}$.
  - $\Gamma = \mathrm{B}$ is a term type, one identity arrow.
- as composition the substitution and renaming of variables;
- as 2-arrows the reductions.

1.3.1 We remark that the reductions between values are only identities, which correspond to the intuition that values are already in normal form, and shall not be reduced further.

We also need to remark that we added some identity arrows on every term type. This is because we are in a categorical framework, where every object is endowed with an identity. We will move to a more general semi-categorical framework starting Section 2.3.

1.4 Intuitionistic linear logic as an operad

1.4.1 In the exact same way than λ-calculus, we will present intuitionistic linear logic typed, and consider the untyped intuitionistic linear logic as a special case. We suppose given two infinite countable sets of variables, the *linear* variables, ranging over $a, b, \dots$ and the *cartesian* variables, ranging over $x, y, \dots$.

Let $\mathbb{T}$ be a set partitioned in two subsets $\mathbb{T}_{\mathsf{l}}$ and $\mathbb{T}_{\mathsf{c}}$, closed by a binary operation $\multimap$ and a unary operation ! that maps elements of $\mathbb{T}_{\mathsf{l}}$ to elements of $\mathbb{T}_{\mathsf{c}}$.

1.4.2 We define sequents of the form

$$\Gamma; \Delta \vdash \theta : t \Rightarrow t' : \mathrm{C}$$

where Γ is a list of the form $a_1 : \mathrm{A}_1, \dots, a_n : \mathrm{A}_n$ of linear variables and types in $\mathbb{T}_{\mathsf{l}}$, the *linear context* and Δ is a list of the form $x_1 : \mathrm{B}_1, \dots, x_m : \mathrm{B}_m$ of cartesian variables and types in $\mathbb{T}_{\mathsf{c}}$, the *cartesian context*, t and t' are two *terms* and C is a type. The rules of reduction formation are given in Figure 1.7, quotiented by the equations of Figure 1.8.

We remark that linear variables appear exactly once in a term.

1.4.3 We borrow the approach to reduction under explicit substitutions of (Accattoli 2012): substitutions acting at a distance avoids commuting conversions. Indeed, in a traditional definition of an explicit substitution calculus, a term such as $(\lambda a.a)\,[!x := u]\,t$ would not reduce to $t\,[!x := u]$ without the commutation $(\lambda a.a)\,[!x := u]\,t \rightsquigarrow ((\lambda a.a)t)\,[!x := u]$.

Definition 10

The **Cat**-operad $\Lambda_{!,\mathbb{T}}$ is defined as the operad with:

- as objects the elements of $\mathbb{T}$;
- as arrows

$$\Upsilon \longrightarrow \mathrm{B}$$

where Υ is an interleaving of Γ (containing only linear types) and Δ (containing only cartesian types), β-less terms $\Gamma; \Delta \vdash t : \mathrm{B}$ of Figure 1.1, quotiented by the equations of Figure 1.2

$$x_1 : \mathrm{A}_1, \dots, x_n : \mathrm{A}_n \vdash t : t \Rightarrow t : \mathrm{B}$$

that is, λ-terms t with free variables included in $\{x_1, \dots, x_n\}$;

- as compositions the substitution and renaming of variables;
- as 2-arrows between two terms the reductions;
- composition ; as composition of 2-arrows.

1.4.4 As for the λ-calculus, we denote by $\Lambda_{!,\star}$ the **Cat**-operad with the terminal set of types suitable for intuitionistic linear logic: the set of types is reduced to two types l and c satisfying:

$$\mathsf{l} \multimap \mathsf{l} = \mathsf{c} \multimap \mathsf{l} = \mathsf{l} \multimap \mathsf{c} = \mathsf{c} \multimap \mathsf{c} = \mathsf{l}$$
$$!\mathsf{l} = !\mathsf{c} = \mathsf{c}.$$

$$\frac{}{a : A; \Delta \vdash a : a \Rightarrow a : A}\text{(variable)}$$

$$\frac{\Gamma; \Delta \vdash \theta : t \Rightarrow t' : A \qquad \Gamma'; \Delta' \vdash \theta' : t' \Rightarrow t'' : A}{\Gamma, \Gamma'; \Delta, \Delta' \vdash \theta; \theta' : t \Rightarrow t'' : A}\text{(composition)}$$

$$\frac{\Gamma; \Delta \vdash \theta : t \Rightarrow t' : A \multimap B \qquad \Gamma'; \Delta' \vdash \kappa : u \Rightarrow u' : A}{\Gamma, \Gamma'; \Delta, \Delta' \vdash \theta\kappa : tu \Rightarrow t'u' : B}\text{(application)}$$

$$\frac{\Gamma, a : A; \Delta \vdash \theta : t \Rightarrow t' : B}{\Gamma; \Delta \vdash \lambda a.\theta : \lambda a.t \Rightarrow \lambda a.t' : A \multimap B}\text{(abstraction)}$$

$$\frac{\begin{array}{rcl} \Gamma, a : A; \vec{y} : \vec{C}, \Delta & \vdash & \theta : t \Rightarrow t' : B \\ \Gamma'; \Delta' & \vdash & \kappa : u \Rightarrow u' : A \end{array} \qquad \forall i \in I, \Gamma_i; \Delta_i \vdash \zeta_i : v_i \Rightarrow v'_i : !C_i}{\Gamma, \Gamma', \Gamma_i; \Delta, \Delta', \Delta_i \vdash (\beta a.\theta)\left[!\vec{y} := \vec{\zeta}\right]\kappa : (\lambda a.t)\left[!\vec{y} := \vec{v}\right]u \Rightarrow t'\{a \leftarrow u'\}\left[!\vec{x} := \vec{v'}\right] : B}\text{(β-reduction)}$$

$$\frac{}{; x : A, \Delta \vdash x : x \Rightarrow x : A}\text{(!-variable)}$$

$$\frac{; \Delta \vdash \theta : t \Rightarrow t' : A}{; \Delta \vdash !\theta : !t \Rightarrow !t' : !A}\text{(promotion)}$$

$$\frac{\Gamma; x : A, \Delta \vdash \theta : t \Rightarrow t' : B \qquad \Gamma'; \Delta' \vdash \kappa : u \Rightarrow u' : !A}{\Gamma, \Gamma'; \Delta, \Delta' \vdash \theta\,[!x := \kappa] : t\,[!x := u] \Rightarrow t'\,[!x := u'] : B}\text{(explicit)}$$

$$\frac{\begin{array}{rcl} \Gamma; x : A, \Delta & \vdash & \theta : t \Rightarrow t' : B \\ \Gamma'; \vec{y} : \vec{C}, \Delta' & \vdash & \kappa : u \Rightarrow u' : !A \end{array} \qquad \forall i \in I, \Gamma_i; \Delta_i \vdash \zeta_i : v_i \Rightarrow v'_i : !C_i}{\Gamma, \Gamma', \Gamma_i; \Delta, \Delta', \Delta_i \vdash \beta\theta\left[!x := \kappa\left[!\vec{y} := \vec{\zeta}\right]\right] : t\left[!x := u\left[!\vec{y} := \vec{v}\right]\right] \Rightarrow t'\{x \leftarrow u'\}\left[!\vec{y} := \vec{v'}\right] : B}\text{(explicit-β)}$$

Figure 1.7: The $\lambda_!$-calculus reductions

$$\begin{aligned}
(\theta_{t\Rightarrow t'}\kappa_{u\Rightarrow u'})_{tu\Rightarrow t'u'}; (\theta'_{t'\Rightarrow t''}\kappa'_{u'\Rightarrow u''})_{t'u'\Rightarrow t''u''} &= (\theta_{t\Rightarrow t'}; \theta'_{t'\Rightarrow t''})_{t\Rightarrow t''}(\kappa_{u\Rightarrow u'}; \kappa'_{u'\Rightarrow u''})_{u\Rightarrow u''} \\
(\lambda x.\theta_{t\Rightarrow t'}); (\lambda x.\theta'_{t'\Rightarrow t''}) &= \lambda x.(\theta; \theta')_{t\Rightarrow t''} \\
(\lambda x.\theta_{t\Rightarrow t'})\kappa_{u\Rightarrow u'}; (\beta x.\theta'_{t\Rightarrow t'})\kappa'_{u'\Rightarrow u''} &= (\beta x.(\theta; \theta')_{t\Rightarrow t''})(\kappa; \kappa')_{u\Rightarrow u''}
\end{aligned}$$

Figure 1.8: The $\lambda_!$-calculus reduction equations.

Example : $(\lambda a.xx\,[!x := a])!(\lambda a.xx\,[!x := a])$ is a closed term of type c (a multiarrow with the empty list as input type and c as output type).

2

Classical calculi as cyclic operads

2.1 Cyclic structures for classical reasoning

2.1.1 Classical calculi do not fit naturally in a categorical framework. Indeed, it is customary of classical reasoning to change conclusion in the course of a proof, corresponding to sending a result, not through the normal output, but through an error channel. Let us consider for instance the standard proof of excluded middle in the system **LK**:

$$\dfrac{\dfrac{\dfrac{}{A \vdash A}\,(\text{axiom})}{\vdash A, \neg A}\,(\vdash \neg)}{\vdash A \vee \neg A}\,(\vdash \vee)$$

It relies crucially on the possibility to have two conclusions at one point of the proof. We interpreted terms as multi-arrows, having many sources but exactly one target. Instead of allowing multiple targets, we will allow to swap one source with the target. In this way, in a sense, we consider the sources and the target on an equal footing (that is, to have only sources, or only targets, in a monolateral sequent calculus kind of way), with one that is temporarily focused.

Cyclic operads

2.1.2 Cyclic operads have been introduced in (Getzler and Kapranov 1995) to account abstractly for Connes (1985)'s cyclic homology. We present here a polychromatic, **Cat**-enriched version.

We consider $\mathfrak{S}_n$ to act on $\{1, \dots, n\}$, we view $\mathfrak{S}_{n+1}$ (writing explicitely a number as a successor) as acting on the set $\{0, \dots, n\}$.

Definition 11 (*cyclic* Cat*-operad*)

A *cyclic* **Cat**-*operad* $\mathscr{O}$ is the data of:

- a class C of *objects*;
- for each integer $n \in \mathbf{N}$ and all tuples $(c_1, \dots, c_n, c) \in \mathrm{C}^{n+1}$, a category
$$\mathscr{O}(c_1, \dots, c_n; c)$$
whose objects are *multi-morphisms* (or *multi-arrows*) from $(c_1, \dots, c_n)$ to c. Such a multi-arrow is written $f : c_1, \dots, c_n \to c$. Morphisms of $\mathscr{C}(c_1, \dots, c_n; c)$ are called 2-arrows;
- for each object $c \in \mathrm{C}$, a distinguished multimorphism
$$\mathrm{id}_c : c \to c,$$
the *identity* of c;
- for every tuple
$$(c_1, \dots, c_n, c),$$
together with n other tuples
$$(d_1^1, \dots, d_1^{k_1}, c_1), \dots, (d_n^1, \dots, d_n^{k_n}, c_n)$$
a morphism
$$\circ : \mathscr{C}(c_1, \dots, c_n; c) \times \mathscr{C}(d_1^1, \dots, d_1^{k_1}; c_1) \times \cdots \times \mathscr{C}(d_n^1, \dots, d_n^{k_n}; c_n) \to \mathscr{C}(d_1^1, \dots, d_n^{k_n}; c)$$
such that composition is associative:
$$\begin{aligned} &\theta \circ \left(\theta_1 \circ (\theta_1^1, \dots, \theta_1^{n_1}), \dots, \theta_k \circ (\theta_k^1, \dots, \theta_k^{n_k}\right) \\ =&\theta \circ (\theta_1, \dots, \theta_k) \circ (\theta_1^1, \dots, \theta_k^{n_k}) \end{aligned}$$
(whenever the mulimorphisms are composable) and unital with respect to the identities:
$$\forall \theta : c_1, \dots, c_n \to c, \qquad \theta \circ (\mathrm{id}_{c_1}, \dots, \mathrm{id}_{c_n}) = \theta = \mathrm{id}_c \circ \theta;$$
- for all object A_0, integer n, permutation $\sigma \in \mathfrak{S}_{n+1}$, and objects $\Gamma = \mathrm{A}_1 \dots \mathrm{A}_n$, a functor
$$\mathsf{exch}_\sigma^{\Gamma;\mathrm{A}} : \mathscr{O}(\mathrm{A}_1, \dots, \mathrm{A}_n; \mathrm{A}_0) \to \mathscr{O}(\mathrm{A}_{\sigma^{-1}(1)}, \dots, \mathrm{A}_{\sigma^{-1}(n)}; \mathrm{A}_{\sigma^{-1}(0)})$$
that satisfy compatibility laws: the action is compatible with the composition in $\mathfrak{S}_n$:
$$\begin{gathered} \forall \sigma, \sigma' \in \mathfrak{S}_{n+1}, \mathsf{exch}_{\sigma'}^{\sigma^{-1}\Gamma;\mathrm{A}} \circ \mathsf{exch}_\sigma^{\Gamma;\mathrm{A}} = \mathsf{exch}_{\sigma' \circ \sigma}^{\Gamma;\mathrm{A}} \\ \mathsf{exch}_{\mathrm{id}}^{\Gamma;\mathrm{A}} = \mathrm{id} \end{gathered}$$
and with the composition in $\mathscr{O}$.

It does not make much sense to try to define cyclic **Cat**-multicategories: indeed, by restricting the action of symmetric group $\mathfrak{S}_{n+1}$ into an action of $\mathfrak{S}_n$, a cyclic **Cat**-multicategory is endowed with a symmetric structure, making it a **Cat**-operad.

Example : We will give examples of fully fledged cyclic **Cat**-operad later in this chapter. For the time being, we only give a cyclic operad (that we trivially consider a **Cat**-operad by adding trivial 2-arrows).

Let us fix a field k. Consider the operad $\mathscr{L}$ of k-multi-linear maps: its objects are finite vector spaces over k, and an arrow in

$$\mathscr{L}(\mathrm{V}_1, \dots, \mathrm{V}_n; \mathrm{V}_0)$$

is a linear map

$$\mathrm{V}_1 \otimes \cdots \otimes \mathrm{V}_n \to \mathrm{V}_0$$

Linear dualization endows $\mathscr{L}$ with a structure of cyclic operad, with the isomorphisms

$$\mathscr{L}(\mathrm{V}_1, \dots, \mathrm{V}_n; \mathrm{V}_0) \simeq \mathscr{L}(\mathrm{V}_1, \dots, \mathrm{V}_n, \mathrm{V}_0^\star; k) \simeq \mathscr{L}(\mathrm{V}_1, \dots, \mathrm{V}_0^\star; \mathrm{V}_n^\star)$$

and, as the spaces are finite dimensional, a space is isomorphic to its dual, so

$$\mathscr{L}(\mathrm{V}_1, \dots, \mathrm{V}_n; \mathrm{V}_0) \simeq \mathscr{L}(\mathrm{V}_1, \dots, \mathrm{V}_0; \mathrm{V}_n).$$

This defines the action of a transposition, which generates all the symmetric group.

Semi-cyclic operad

2.1.3 In some cases, not all inputs can be used as a conclusion: among all inputs, some are outputs-in-waiting and some are real inputs. We nonetheless want to be able to compose outputs with inputs. This leads to the structure of *semi-cyclic operad* where only some inputs can be replaced by outputs. While we are at it, we do not consider identities on outputs.

Definition 12 (*semi-cyclic* Cat*-operad*)

A *cyclic* **Cat***-operad* $\mathscr{O}$ is the data of:

- three disjoint classes of *objects*,
  - C_1 of *input objects*, noted $c, d, \dots,$
  - C_2 of *outputs-in-waiting objects*, noted $\underline{\underline{c}}, \underline{\underline{d}}, \dots$
  - C_3 of *output objects*, noted $\underline{c}, \underline{d}, \dots.$

  such that C_1, C_2 and C_3 are canonically in bijection, which justifies us denoting the objects by input objects, more or less underlined. We write $\mathrm{C} = \mathrm{C}_1 \sqcup \mathrm{C}_2 \sqcup \mathrm{C}_3$. When we will want to consider an object without specifying its status, we will write it $\underset{-}{c}$;
- for each integer $n \in \mathbf{N}$ and all tuples $(c_1, \dots, c_n, c) \in \mathrm{C}^{n+1}$, a category

  $$\mathscr{O}(c_1, \dots, c_n; c)$$

  whose objects are *multi-morphisms* (or *multi-arrows*) from $(c_1, \dots, c_n)$ to c. Such a multi-arrow is written $f : c_1, \dots, c_n \to c$. Morphisms of $\mathscr{C}(c_1, \dots, c_n; c)$ are called 2-arrows;

- for each object $c \in C_1$, a distinguished multimorphism

  $$\mathrm{id}_c : c \to c,$$

  the *identity* of c;
- for every tuple

  $$(c_1, \dots, c_n, c),$$

  together with n other tuples

  $$(d_1^1, \dots, d_1^{k_1}, c_1), \dots, (d_n^1, \dots, d_n^{k_n}, c_n)$$

  a morphism

  $$\circ : \mathscr{C}(c_1, \dots, c_n; c) \times \mathscr{C}(d_1^1, \dots, d_1^{k_1}; c_1) \times \cdots \times \mathscr{C}(d_n^1, \dots, d_n^{k_n}; c_n) \to \mathscr{C}(d_1^1, \dots, d_n^{k_n}; c)$$

  such that composition is associative:

  $$\begin{aligned} &\theta \circ \left(\theta_1 \circ (\theta_1^1, \dots, \theta_1^{n_1}), \dots, \theta_k \circ (\theta_k^1, \dots, \theta_k^{n_k}\right) \\ =&\theta \circ (\theta_1, \dots, \theta_k) \circ (\theta_1^1, \dots, \theta_k^{n_k}) \end{aligned}$$

  (whenever the mulimorphisms are composable) and unital with respect to the identities:

  $$\forall \theta : c_1, \dots, c_n \to c, \qquad \theta \circ (\mathrm{id}_{c_1}, \dots, \mathrm{id}_{c_n}) = \theta = \mathrm{id}_c \circ \theta;$$
- for all object $\underline{\mathrm{A}_0}$, integer n, permutation $\sigma \in \mathfrak{S}_{n+1}$, and objects $\Gamma = \mathrm{A}_1, \dots, \underline{\underline{\mathrm{A}_{\sigma^{-1}(0)}}}, \dots, \mathrm{A}_n$, such that a functor

  $$\mathsf{exch}_\sigma^{\Gamma;\mathrm{A}} : \mathscr{O}(\mathrm{A}_1, \dots, \mathrm{A}_n; \mathrm{A}_0) \to \mathscr{O}(\mathrm{A}_{\sigma^{-1}(1)}, \dots, \underline{\underline{\mathrm{A}_0}}, \dots, \mathrm{A}_{\sigma^{-1}(n)}; \underline{\mathrm{A}_{\sigma^{-1}(0)}})$$

  that satisfy compatibility laws: the action is compatible with the composition in $\mathfrak{S}_n$:

  $$\begin{gathered} \forall \sigma, \sigma' \in \mathfrak{S}_n, \mathsf{exch}_{\sigma'}^{\sigma^{-1}\Gamma;\mathrm{A}} \circ \mathsf{exch}_\sigma^{\Gamma;\mathrm{A}} = \mathsf{exch}_{\sigma' \circ \sigma}^{\Gamma;\mathrm{A}} \\ \mathsf{exch}_{\mathrm{id}}^{\Gamma;\mathrm{A}} = \mathrm{id} \end{gathered}$$

  and with the composition in $\mathscr{O}$.

  The $\mathsf{exch}^{;}$ operation swaps an output with an output-in-waiting.

Remark 5

A semi-operad is, usually, an operad with objects lacking identities. We have chosen the modifier *semi-* as, in our case, the outputs-in-waiting lack identities, so a semi-cyclic operad is, when the cyclic structure is forgotten, a semi-operad.

Example : Every cyclic **Cat**-operad $\mathscr{O}$ defines a semi-cyclic **Cat**-operad, whose objects are the objects of $\mathscr{O}$ triplicated.

Example : Linear applications from vector spaces to a finite dimensional vector space.

2.2 Proof-nets as a cyclic operad

Proof-nets, informally

2.2.1 Proof-nets are a formalism introduced by Girard (1987) to represent proofs of Linear Logic in a more synthetic way than sequent calculus.

We suppose given a set $\mathbb{T}$ of MELL types closed by the unary operations $!, ?, (\cdot)^{\perp}$ and the binary operations $⅋, \otimes$, verifying:

$$\begin{aligned}(A \otimes B)^{\perp} &= B^{\perp} ⅋ A^{\perp}\\ (A ⅋ B)^{\perp} &= B^{\perp} \otimes A^{\perp}\\ (!A)^{\perp} &= ?A^{\perp}\\ (?A)^{\perp} &= !A^{\perp}.\end{aligned}$$

We will see that this parameter set is functorial.

Multiplicative exponential proof-nets are graphs generated by the *multiplicative linear logic cells*, depicted in Figure 2.1.

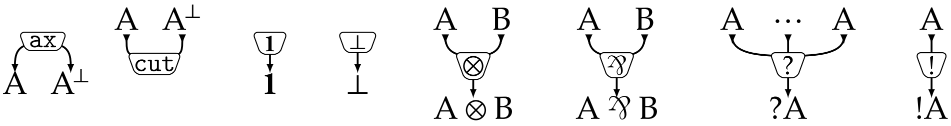


Figure 2.1: Multiplicative-exponential linear logic cells

Graphs

2.2.2 We follow the definition of a graph given by Borisov and Manin (2008)[1]: in particular, a graph is not the datum of a set of edges and a set of vertices, but the edges are split in halves, allowing for some of them to be hanging.

[1] The folklore attributes the definition of graphs with half-edges to Kontsevitch and Manin, but the idea can actually be traced back to Grothendieck's *dessins d'enfant*. The main contribution of (Borisov and Manin 2008) is to define morphisms for this definition of graphs – that we will not use here – that are of great generality.

Definition 13 (*graph*)

A (finite) **graph** τ is a quadruple $(F_\tau, V_\tau, \partial_\tau, j_\tau)$, where

- F_τ is a finite set, whose elements are called *flags* of τ;
- V_τ is a finite set, whose elements are called *vertices* of τ;
- $\partial_\tau : F_\tau \to V_\tau$ is a function associating to each flag its *boundary*;
- $j_\tau : F_\tau \to F_\tau$ is an involution.

A flag fixed by the involution is a *tail* of τ.

Two-elements orbits of the involution form a set E_τ of *edges* of τ. Elements of an edge e are called *halves* of e.

Given two graphs τ and τ', it is always possible to consider their disjoint union $\tau \sqcup \tau'$ defined as the disjoint union of the underlying sets and functions.

2.2.3 A one vertex graph with set of flags F and involution the identity on F is called the *corolla* with set of flags F. It is standardly written $*_F$. One corolla $*_5$ is depicted Figure 2.2.

Given a graph $\tau = (F_\tau, V_\tau, \partial_\tau, j_\tau)$, a vertex v defines a corolla τ_v, by, if we set $F_v = \partial_\tau^{-1}(v)$:

$$\tau_v = (F_v, v, \partial_\tau|_{F_v}, \mathrm{id}_{F_v}).$$

Every graph can be described as the set of corollas of its vertices, together with the involution glueing the flags in edges.

2.2.4 Different notions of morphisms exist between graphs. As we will mainly be interested in isomorphisms and subgraphs, we will use the most naive notion.

Definition 14 (*graph morphism*)

Let τ, σ be two graphs. A **graph morphism**

$$h : \tau \to \sigma$$

is a couple of functions $(h_F : F_\tau \to F_\sigma, h_V : V_\tau \to V_\sigma)$ such that $h_F \circ \partial_\tau = \partial_\sigma \circ h_V$ and $h_F \circ j_\tau = j_\sigma \circ h_F$.

A graph morphism is said to be *injective* if both its component functions are.

The category **Graph** has graphs as objects and morphisms of graphs as morphisms: indeed, graph morphisms compose (by composing the underlying functions) and the couple of identities (on vertices and flags) is neutral. It is a monoidal category, with disjoint union as a monoidal product.

Graphs with structure

2.2.5 Some structure can be put on top of a graph. For instance:

Definition 15

- A **labelled graph** (τ, ℓ) with labels in I is a graph τ together with a function $\ell : V_\tau \to I$;
- a **colored graph** (τ, c) with colors in a set C is a graph τ together with a function c :

$F_\tau \to C$ such that, for two halves f, f' of any edge of τ,

$$\mathsf{c}(f) = \mathsf{c}(f');$$

- an **oriented graph** (τ, o) is a graph τ together with a function $\mathsf{o} : F_\tau \to \{\mathbf{in}, \mathbf{out}\}$ such that, for two halves f, f' of any edge of τ,

$$\mathsf{o}(f) \neq \mathsf{o}(f');$$

  in-oriented tails of τ are called *inputs* of τ, **out**-oriented tails are called *outputs* of τ;
- an **ordered graph** $\tau, \leqslant_\tau$ is a graph together with an order on the flags.

The different structures on a graph combine.

Example: Let $\mathbf{5} = \{0, 1, 2, 3, 4\}$ be the finite cardinal endowed with the order $0 <_5 4$ and $1 <_5 2 <_5 3$, and

- o the orientation defined by

$$\mathsf{o}(0) = \mathsf{o}(4) = \mathbf{out}$$
$$\mathsf{o}(1) = \mathsf{o}(2) = \mathsf{o}(3) = \mathbf{in},$$

- ℓ defined by $\ell(*) = ✠$,
- $\mathsf{c} : \mathbf{5} \to \{a_0, \dots, a_4\}$ the coloring defined by

$$\forall i \in \mathbf{5}, \mathsf{c}(i) = a_i.$$

The ordered labelled oriented colored corolla $(*_5, \mathsf{o}, \ell, \mathsf{c}, <_5)$ will be depicted as in Figure 2.2.

2.2.6 Each enrichment of the structure of graphs of Definition 15 defines a notion of morphism that preserves it and an associated category.

2.2.7 We can depict graphs as two-dimensional figures. As a graph is just a disjoint union of corollas glued with the involution, we only need to show depictions of corollas (as in Figure 2.2) and how to depict an edge to be able to depict full graphs. We will always depict the inputs of a corolla above the corolla, and its outputs below, with arrows also indicating the orientation. The colors are written next to the arrows. If ordered, the different flags of a corolla are depicted increasing from left to right.

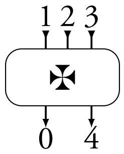


Figure 2.2: One depiction of a labelled oriented corolla

Trees

2.2.8 A graph can be realized geometrically. The geometric realization of a corolla $*_S$ is the disjoint union $\bigsqcup_S [0; \frac{1}{2}]$ with end-points 0 identified. The geometric realization of a graph is the disjoint union of the geometric realization of the corollas of all its vertices, with points $\frac{1}{2}$ of any two flags forming an orbit under the involution of the graph identified.

A graph is *(simply) connected* if its geometric realisation is (simply) connected.

A *tree* is a connected, simply connected graph.

2.2.9 A *rooted tree* is an oriented tree such that each vertex has exactly one **out** flag. A rooted tree only has one output tail. Its boundary is called the *root* of the rooted tree.

A special class of morphism is of interest for rooted trees.

Definition 16

> Let τ_1 and τ_2 be two rooted trees, and $h : \tau_1 \to \tau_2$ be a morphism of graphs.
>
> h is a **morphism of rooted trees** if h_V maps the root of τ_1 to the root of τ_2.

A *sub-rooted tree* of a tree τ is a tree τ' together with an injective morphism of rooted tree $\tau' \to \tau$.

2.2.10 It is sometimes useful to consider not a rooted tree, but its *reflexive-transitive closure* (as in Definition 17). An (oriented) *path* on a oriented graph τ is either empty or a sequence $f_0 f_1 \cdots f_{2n+1}$ such that the f_{2i} are outputs and the f_{2i+1} are inputs, and $\forall 0 \leqslant i < n, \partial(f_{2i+1}) = \partial(f_{2i+2})$. Such a path is said to be from $\partial(f_0)$ to $\partial(f_{2n+1})$. The empty path is a path from any vertex to itself.

The set of paths on a tree are finite. As such, given a tree τ, we define its *reflexive-transitive closure*, or *free category* $\tau^{\circlearrowleft}$ as the graph with same vertices and same tails as τ, and with edges oriented from v to v' the paths from v to v'.

2.2.11 Rooted trees and morphisms of rooted tree form a category **RootTree**. The reflexive-transitive closure operator extends to a functor $(\cdot)^{\circlearrowleft}$: **RootTree** $\to$ **Graph**.

Proof-nets

2.2.12 We adopt the **Cat**-operadic point of view also for proof-nets, which mean our fundamental objects will be reductions, and usual proof-nets will be particular cases of reductions: identities. Nonetheless, we will first define proof-nets with no reductions, then, pointed proof-nets with reductions, that is, with a distinguished conclusion, then, finally, proof-nets with reductions.

We formalize the intuition of Section 2.2 by equipping a graph by labels (which specify the types of the vertices, which are the connectives of MELL), colors (which specify the types of the flags, which are formulæ of MELL), and a function that specify the deepest box a flag is in, all subject to compatibility conditions.

Definition 17 (*proof-net*)

> A MELL *module* is a tuple $\mathrm{R} = (\tau, \ell, \mathsf{c}, \leqslant, \mathscr{T}, \mathsf{box})$, where
>
> - τ is a labelled ordered oriented colored graph $(\tau, \ell, \mathsf{c}, \leqslant)$, the *underlying graph* of R such that:
>   - $\ell : \mathrm{V}_\tau \to \{\mathtt{ax}, \mathtt{cut}, \mathbf{1}, \bot, \otimes, \parr, ?, !\}$;
>   - $\mathsf{c} : \mathrm{F}_\tau \to \mathscr{F}_{\mathsf{MELL}}$;
>   - let $v \in \mathrm{V}_\tau$.
>   - ∗ if $\ell(v) = \mathtt{cut}$, the corolla τ_v has only two flags i_1 and i_2 which are inputs, and

such that

$$\mathsf{c}(i_1) = \mathsf{c}(i_2)^{\perp};$$

- if $\ell(v) = \mathtt{ax}$, the corolla τ_v has only two flags o_1 and o_2 which are outputs, and such that

$$\mathsf{c}(o_1) = \mathsf{c}(o_2)^{\perp};$$

- if $\ell(v) = \mathbf{1}$, the corolla τ_v has no inputs and one output o, such that

$$\mathsf{c}(o) = \mathbf{1};$$

- if $\ell(v) = \perp$, the corolla τ_v has no inputs and one output o, such that

$$\mathsf{c}(o) = \perp;$$

- if $\ell(v) = \otimes$, the corolla τ_v has two inputs $i_1 < i_2$ and one output o, such that

$$\mathsf{c}(o) = \mathsf{c}(i_1) \otimes \mathsf{c}(i_2);$$

- if $\ell(v) = \parr$, the corolla τ_v has two inputs $i_1 < i_2$ and one output o, such that

$$\mathsf{c}(o) = \mathsf{c}(i_1) \parr \mathsf{c}(i_2);$$

- if $\ell(v) = ?$, the corolla τ_v has many inputs $i_1, \dots, i_n (n \leqslant 0)$ and one output o, such that

$$\forall 1 \leqslant j \leqslant n, \mathsf{c}(o) = ?\mathsf{c}(i_j);$$

- if $\ell(v) = !$, the corolla τ_v has one input i and one output o, such that

$$\mathsf{c}(o) = !\mathsf{c}(i).$$

These corollas are depicted Figure 2.1.

– $\leqslant$ is total.

- $\mathscr{T}$ is a rooted tree, the *box-tree* of R;
- $\mathsf{box} : \tau \to \mathscr{T}^{\circlearrowleft}$ is a morphism of graphs, the *box-function* of R such that:
  - box_V, induces a bijection between $\{v \in \mathrm{V}_\tau \mid \ell(v) = !\}$ and the non-root vertices of $\mathscr{T}$;
  - let $\{f, f'\}$ be an edge such that f is an output. If $\mathsf{box}_\mathrm{V}(\partial_\tau f) \neq \mathsf{box}_\mathrm{V}(\partial_\tau f')$, then $\ell(\partial_\tau f) \in \{!, ?\}$. If $\ell(\partial_\tau f) = !$ then $\mathsf{box}_\mathrm{V}(\partial_\tau f) \neq \mathsf{box}_\mathrm{V}(\partial_\tau f')$.

A MELL **proof-net** is a module with no input tails.

If R is a proof-net, we say that the vertices of τ_R (the underlying graph or R) are the *cells* of the proof-net, the edges of τ_R are the *wires* and the tails are the *conclusion* of R.

Lemma 2

Let $\mathrm{R} = (\tau, \ell, \mathsf{c}, \leqslant, \mathscr{T}, \mathsf{box})$ *be a proof-net. The function* c *is uniquely determined by* ℓ *and its* *image on the conclusions of* R.

2.2.13 The proof-nets we just defined are particularily rigid: a proof-net is depedent on its carrier-sets of cells and wires. We will only consider proof-nets modulo the isomorphim arising from its labelled graph structure.

Definition 18 (*isomorphism of proof-nets*)

A **isomorphism of proof-nets** $f : \mathrm{R} \simeq \mathrm{R}'$ is

- an isomorphism $f : \tau_{\mathrm{R}} \to \tau_{\mathrm{R}'}$ of the underlying graph
- an isomorphism $f_{\mathsf{box}} : \mathscr{T}_{\mathrm{R}} \to \mathscr{T}_{\mathrm{R}'}$ of the boxing trees

such that

$$\begin{tikzcd}
\tau_{\mathrm{R}} \arrow[r, "\mathsf{box}_{\mathrm{R}}"] \arrow[d, "f"] & \mathscr{T}_{\mathrm{R}} \arrow[d, "f_{\mathsf{box}}"] \\
\tau_{\mathrm{R}'} \arrow[r, "\mathsf{box}_{\mathrm{R}'}"] & \mathscr{T}_{\mathrm{R}'}
\end{tikzcd}$$

and such that, moreover, the restrictions of f on:

- the tails,
- the inputs of each cell of type $?, \otimes, ⅋$,

are all increasing.

We will actually always consider proof-nets quotiented by this isomorphism, and will implicitly verify that every construction we give factor through this quotient.

2.2.14 In order to fit in the operadic framwork we outlined, we need to specify which conclusions are to be seen as in the source of the multi-arrow and which conclusion is seen as in the target. We achieve this by specifying a distinguished conclusion on proof-nets.

Definition 19

A **pointed proof-net** is a proof-net R with a distinguished conclusion c.

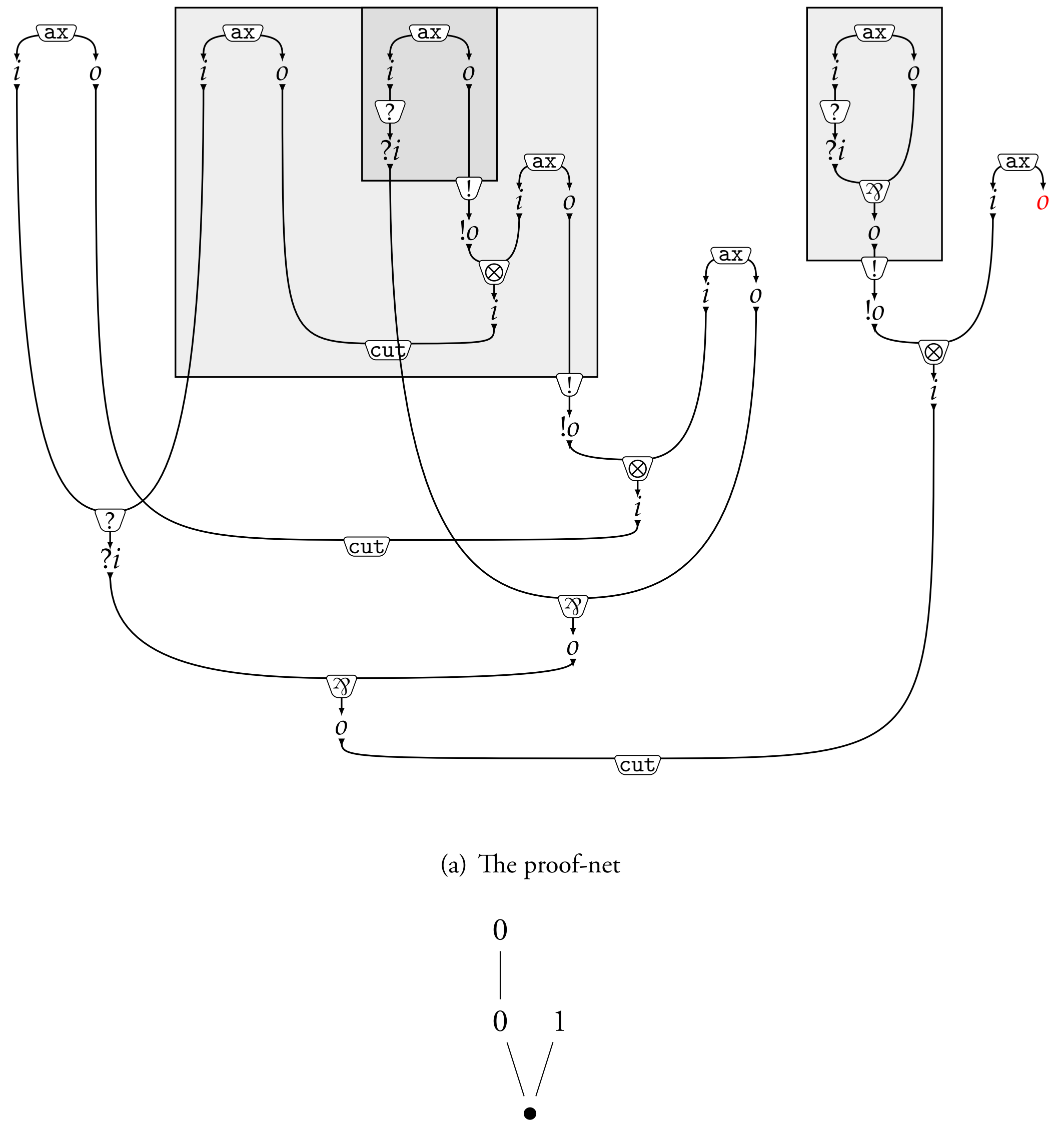
Example : The graph of Figure 2.3 is a proof-net, with only one conclusion, distinguished, and so written in red. The set of types is taken to be $\{i, o\}$, satisfying the equations $i = o^{\perp}$, $i = !o \otimes i$.

The greyed areas represent the inverse images of boxes.

As the non-distinguished conclusions are totally ordered, a pointed proof-net (R, c) can be seen as an arrow

$$\mathrm{A}_1, \dots, \mathrm{A}_n \to \mathrm{B}$$

where A_i is the type of the i^{th} non-distinguished conclusion, and B is the type of the distinguished conclusion. We will make this structure more precise, by defining the reductions.



(a) The proof-net

(b)
Its box-tree

Figure 2.3: A pointed proof-net.

2.2.15 The reduction proof-nets have a much more complicated structure than the proof-nets. Indeed, any cell of a reduction proof-net can be a reduction, carried in parallel, independently from the rest of the proof-net. We achieve this by defining *reduction cells*, so that reductions are represented by a structure akin to proof-nets, where some cells are reduction cells.

Definition 20

A **reduction cell** is a an oriented labelled colored corolla satisfying:

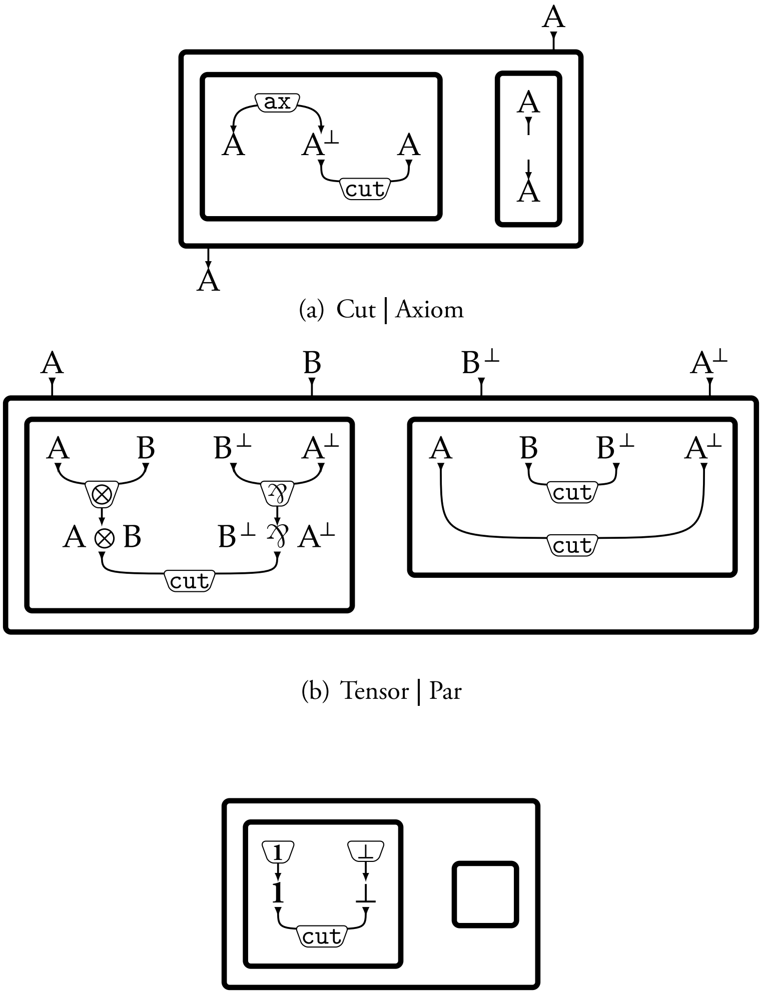
- the cell is labelled with a couple (m_1, m_2) of MELL-modules together with two increasing bijections between the sets of inputs (respectively outputs) of m_1 and m_2 that commute with c;

- the inputs (respectively outputs) of the reduction cells are in bijection with the inputs (respectively outputs) of m_1, and the bijection commutes with c.

The set of reduction cells is noted Red.

A reduction cell is endowed with two projections, the left and the right one, that associates with a cell (m_1, m_2) the left and the right module in it. The left and the right projection have potentially a different box-tree.

The reduction cells are depicted Figures 2.4 to 2.6, where π and π' are arbitrary MELL proof-nets. The set of reduction cells is noted Red.



(a) Cut | Axiom

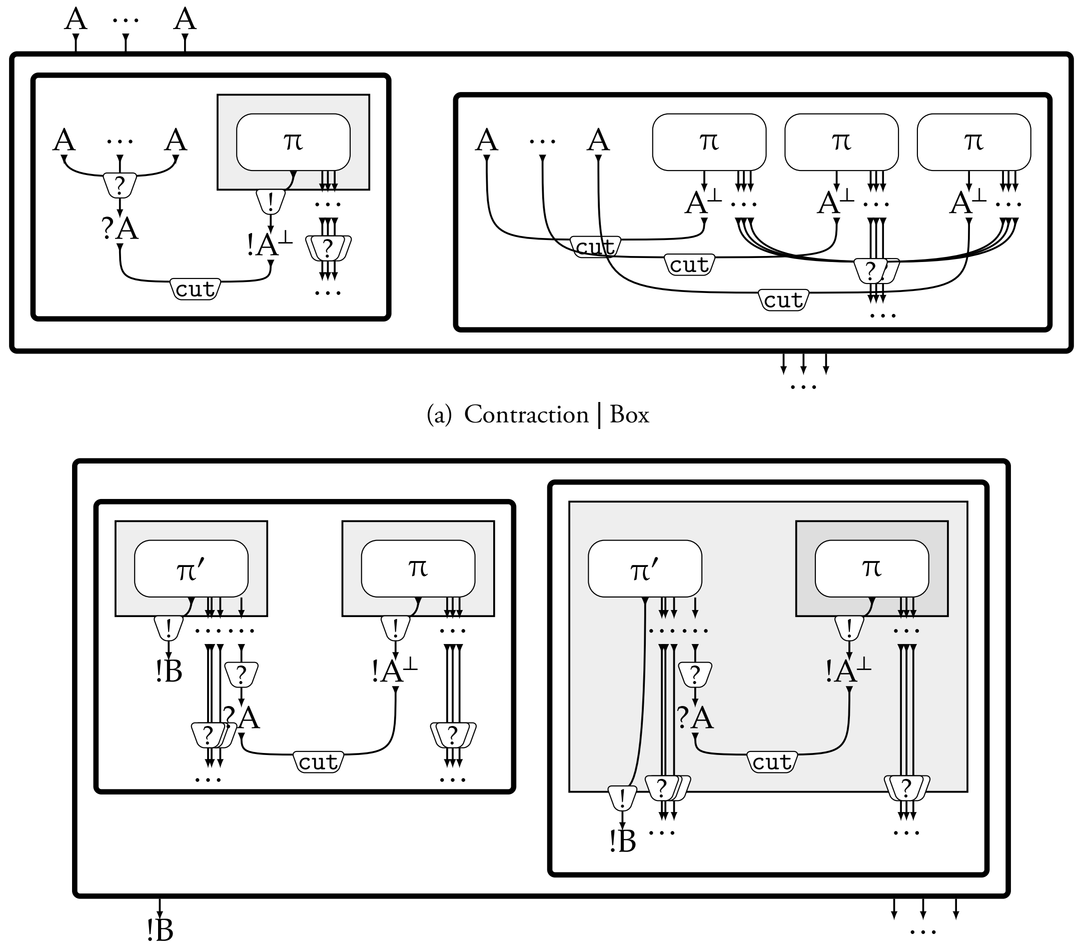
(b) Tensor | Par

(c) Tensor | Par

Figure 2.4: The MELL multiplicative reduction cells

Definition 21 (*reductive* MELL *proof-net*)

A *reductive* MELL *module* is a tuple $R = (\tau, \ell, c, \leqslant, \mathscr{T}, \mathsf{box})$, where τ is a labelled ordered oriented



(a) Contraction | Box

(b) Box | Box

Figure 2.5: The MELL exponential reduction cells

colored graph $(\tau, \ell, \mathtt{c}, \leqslant)$, the *underlying graph* of R such that:

- $\ell : V_\tau \to \{\mathtt{ax}, \mathtt{cut}, \mathbf{1}, \bot, \otimes, \parr, ?, !, \mathsf{Red}\}$;
- $\mathtt{c} : F_\tau \to \mathscr{F}_{\mathsf{MELL}}$.

It is endowed with two projections, whose underlying graph is obtained by grafting the projections of the reduction cells with the rest of the graph, while the box-trees are defined by the grafting of the box-tree of the projections of the reduction cells with the one of the underlying graph.

We require the two projected graphs to be MELL modules.

A reductive proof-net has two projections, the left and the right one, which are MELL proof-nets. In particular, the two proof-nets need not share the same box-tree.

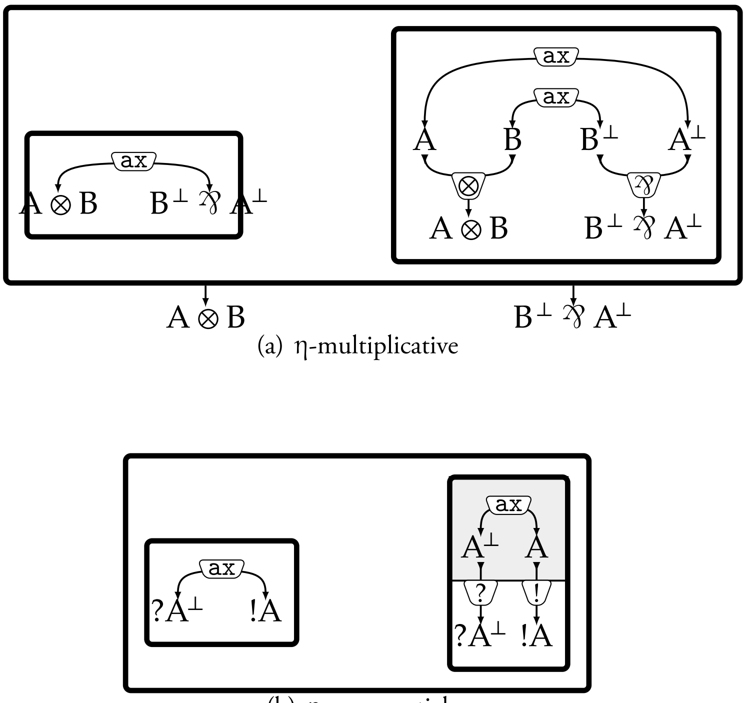
2.2.16 Let R and R' be two reductive proof-nets, such that the right projection of R is isomorphic to the left projection of R'. Each represent a reduction, and as such, $R; R'$ represent a reduction sequence. Just as we did for λ-calculi, we need to ensure permutation equivalence, which is done by:

- requiring a reductive cell to commute with any non-reductive proof-net;
- requiring any reductive cell with the same cell c in its left and right projection to be equal to the reductive proof-net with the reductive cell without c and c attached to it;
- requiring a reductive cell that n-plicates a box containing π followed by n reductive cells reducing π to π' to be equal with the reductive cell that reduce π into π' inside a box followed by the n-plication of π'.

2.2.17 The cyclic **Cat**-operad MELL of proof-nets is defined by:

- its objects are MELL formulæ;
- its multiarrows are pointed MELL proof-nets;
- its 2-arrows are sequences of composable reductive proof-nets modulo the equations described above.

2.2.18 Just like the λ-calculus, proof-nets have their kind of extensionality.



(a) η-multiplicative

(b) η-exponential

Figure 2.6: Extensionality

2.3 The λμ-calculus as a cyclic operad

The λμ-calculus (Parigot 1992) is a well-known extension of the λ-calculus that captures classical reasoning (and control operators). From the operadic point of view, this corresponds to being able to permute certain inputs and outputs. It is therefore our motivating example for introducing semi-cyclic operads.

Given two infinite disjoint sets, a set of *variables* ranging over $x, y, z, \ldots$ and a set of *names* ranging over $\alpha, \beta, \ldots$, and a set $\mathbb{T}$ closed by a binary operation $\to$, the λμ reductions are generated by the rules of Figure 2.7 where $\{\cdot /\!\!/ \cdot\}$ is recursively defined by:

$$\begin{aligned}
x\{u/\!\!/\alpha\} &= x \\
(\lambda x.t)\{u/\!\!/\alpha\} &= \lambda x.t\{u/\!\!/\alpha\} \\
(tu)\{u/\!\!/\alpha\} &= t\{u/\!\!/\alpha\}u\{u/\!\!/\alpha\} \\
(\mu\gamma.\ulcorner\alpha\urcorner t)\{u/\!\!/\alpha\} &= \mu\gamma.\ulcorner\alpha\urcorner(t\{u/\!\!/\alpha\})u \\
(\mu\gamma.\ulcorner\beta\urcorner t)\{u/\!\!/\alpha\} &= \mu\gamma.\ulcorner\beta\urcorner(t\{u/\!\!/\alpha\}), \beta \neq \alpha
\end{aligned}$$

modulo the equations of Figure 2.8

$$\frac{}{\Gamma, x : \mathrm{A}; \Delta \vdash x : x \Rightarrow x : \mathrm{A}}\,\text{(variable)}$$

$$\frac{\Gamma; \Delta \vdash \theta : t \Rightarrow t' : \mathrm{A} \qquad \Gamma; \Delta \vdash \theta' : t' \Rightarrow t'' : \mathrm{A}}{\Gamma; \Delta \vdash \theta; \theta' : t \Rightarrow t'' : \mathrm{A}}\,\text{(composition)}$$

$$\frac{\Gamma; \Delta \vdash \theta : t \Rightarrow t' : \mathrm{A} \to \mathrm{B} \qquad \Gamma; \Delta \vdash \kappa : u \Rightarrow u' : \mathrm{A}}{\Gamma; \Delta \vdash \theta\kappa : tu \Rightarrow t'u' : \mathrm{B}}\,\text{(application)}$$

$$\frac{\Gamma, x : \mathrm{A}; \Delta \vdash \theta : t \Rightarrow t' : \mathrm{B}}{\Gamma; \Delta \vdash \lambda x.\theta : \lambda x.t \Rightarrow \lambda x.t' : \mathrm{A} \to \mathrm{B}}\,\text{(abstraction)}$$

$$\frac{\Gamma; \Delta, \alpha : \mathrm{A}, \beta : \mathrm{B} \vdash \theta : t \Rightarrow t' : \mathrm{B}}{\Gamma; \Delta, \beta : \mathrm{B} \vdash \mu\alpha.\ulcorner\beta\urcorner\theta : \mu\alpha.\ulcorner\beta\urcorner t \Rightarrow \mu\alpha.\ulcorner\beta\urcorner t' : \mathrm{A}}\,\text{(name)}$$

$$\frac{\Gamma, x : \mathrm{A}; \Delta \vdash \theta : t \Rightarrow t' : \mathrm{B} \qquad \Gamma; \Delta \vdash \kappa : u \Rightarrow u' : \mathrm{A}}{\Gamma; \Delta \vdash (\beta x.\theta)\kappa : (\lambda x.t)u \Rightarrow t'\{x \leftarrow u'\} : \mathrm{B}}\,\text{(β-reduction)}$$

$$\frac{\Gamma; \Delta, \alpha : \mathrm{A} \to \mathrm{C} \vdash \theta : t \Rightarrow t' : \mathrm{B} \qquad \Gamma; \Delta \vdash \kappa : u \Rightarrow u' : \mathrm{A}}{\Gamma; \Delta, \beta : \mathrm{B} \vdash [(\mu\alpha.\ulcorner\beta\urcorner\theta)\kappa] : (\mu\alpha.\ulcorner\beta\urcorner t)u \Rightarrow \mu\alpha.\ulcorner\beta\urcorner t'\{u'/\!\!/\alpha\} : \mathrm{C}}\,\text{(μ-reduction)}$$

Figure 2.7: The λμ-calculus reductions

Definition 22 (λμ ***operad***)

The semi-cyclic **Cat**-semi-operad $\Lambda\mathrm{M}_{\mathbb{T}}$ is defined as the operad with:

- as objects three copies of the elements of $\mathbb{T}$, respectively as inputs, outputs-in-waiting, and outputs;
- as arrows, with $\Gamma = \mathrm{A}_1, \ldots, \mathrm{A}_n$ and $\underline{\underline{\Delta}} = \underline{\underline{\mathrm{C}_1}}, \ldots, \underline{\underline{\mathrm{C}_m}}$,

$$
\begin{aligned}
(\theta_{t\Rightarrow t'}\kappa_{u\Rightarrow u'})_{tu\Rightarrow t'u'};(\theta'_{t'\Rightarrow t''}\kappa'_{u'\Rightarrow u''})_{t'u'\Rightarrow t''u''} &= (\theta_{t\Rightarrow t'};\theta'_{t'\Rightarrow t''})_{t\Rightarrow t''}(\kappa_{u\Rightarrow u'};\kappa'_{u'\Rightarrow u''})_{u\Rightarrow u''}\\
(\lambda x.\theta_{t\Rightarrow t'});(\lambda x.\theta'_{t'\Rightarrow t''}) &= \lambda x.(\theta;\theta')_{t\Rightarrow t''}\\
(\lambda x.\theta_{t\Rightarrow t'})\kappa_{u\Rightarrow u'};(\beta x.\theta'_{t\Rightarrow t'})\kappa'_{u'\Rightarrow u''} &= (\beta x.(\theta;\theta')_{t\Rightarrow t''})(\kappa;\kappa')_{u\Rightarrow u''}\\
(\mu\alpha.\lceil\beta\rceil\theta_{t\Rightarrow t'})\kappa_{u\Rightarrow u'};[(\mu\alpha.\lceil\beta\rceil\theta'_{t'\Rightarrow t''})\kappa'_{u'\Rightarrow u''}] &= [(\mu\alpha.\lceil\beta\rceil(\theta;\theta')_{t\Rightarrow t''}\kappa_{u\Rightarrow u''}]
\end{aligned}
$$

Figure 2.8: The λμ-calculus reduction equations.

$$\Gamma,\underline{\underline{\Delta}} \xrightarrow{\qquad\qquad} \underline{\mathrm{B}}$$

β-less terms of Figure 2.7, quotiented by the equations of Figure 2.8

$$x_1 : \mathrm{A}_{\ 1},\dots,x_n : \mathrm{A}_{\ n},\alpha_1 : \mathrm{C}_1,\dots,\alpha_m : \mathrm{C}_m \vdash t : t \Rightarrow t : \mathrm{B}$$

that is, λ-terms t with free variables included in $\{x_1,\dots,x_n\}$ and names included in $\{\alpha_1,\dots,\alpha_m\}$;

- as arrows $\underline{\mathrm{A}} \to \mathrm{A}$ silent coercions;
- the only composable arrows are mediated through these coercions. We define the composition as the substitution and renaming of variables and names;
- as 2-arrows the reductions;
- composition ; as composition of 2-arrows;
- for a morphism

$$x_1,\dots,x_n,\alpha_1,\dots,\alpha_m \vdash t : \Gamma;\underline{\underline{\Delta}} \to \underline{\mathrm{B}},$$

with $\underline{\underline{\Delta}} = \underline{\underline{\mathrm{C}_1}},\dots,\underline{\underline{\mathrm{C}_m}}$ and $1 \leqslant i \leqslant m$, the action of the transposition $(0\ i)$ is the term

$$\mu\alpha_0.\lceil\alpha_i\rceil t\{\alpha_0/\alpha_i\} : \Gamma;\underline{\underline{\mathrm{C}_1}},\dots,\underline{\underline{\mathrm{B}}},\dots,\underline{\underline{\mathrm{C}_m}} \to \underline{\mathrm{C}_i},$$

which is enough to define the action of $\mathfrak{S}_{n+1}$ as the symmetric group is generated by the transposition with a fixed element.

It must be remarked that the potential outputs have no identities. Indeed, $;\alpha : \mathrm{A} \vdash \alpha : \mathrm{A}$ is not a term (or anything approaching it).

2.3.1 We can add extensionality to the λμ-calculus, by the reductions of Figure 2.9.

$$\frac{\Gamma \vdash \theta : t \Rightarrow t' : \mathrm{A} \to \mathrm{B}}{\Gamma \vdash (\eta x.\theta) : t \Rightarrow \lambda x.(t'x) : \mathrm{A} \to \mathrm{B}}\,(\eta\text{-expansion})$$

Figure 2.9: The λμ-calculus extensionality reductions

3

Morphisms of operads

3.0.1 For the time being, we have only been interested in calculi, and categorical structures representing them. As usual in mathematics, the morphisms of structures have an interest, and the morphisms of calculi are no exceptions. As we will see, we have three different ways of interpreting a morphism between two **Cat**-operads:

$$\begin{tikzcd} \mathscr{C} \arrow[r, "f"] & \mathscr{D} \end{tikzcd}$$

depending on how we view the two **Cat**-operads. Indeed, we will view either

- f as a translation of a calculus $\mathscr{C}$ into a calculus $\mathscr{D}$;
- f as a type system for the calculus $\mathscr{D}$, refining the types of $\mathscr{D}$ as types of $\mathscr{C}$, and typing the terms in $\mathscr{D}$ with derivations in $\mathscr{C}$;
- f as a semantics for the calculus $\mathscr{C}$, and $\mathscr{D}$ as an algebra for it.

3.1 CPS translations of the λ-calculus

3.1.1 Different flavors of λ-calculus can be translated thanks to so-called continuation-passing-style (CPS) translations. Invented for compiling functionnal languages such as Lisp, they enjoy a close relationship with various logical translation, such as Gödel $\neg\neg$-translation or forcing. These translations have first been given by Plotkin (1975), and then sharpened in a lot of studies.

From call-by-value to call-by-name

3.1.2 The translation from call-by-value to call-by-name we present is directly inspired by the one originally presented by Plotkin (1975, §6). The translation acts on the sequents of the call-by-value λ-calculus, and is the identity on types and contexts. So, omitting contexts and types, it is given by Figure 3.1. We remark, that, if u is a value, $\mathbf{P}(u) = \lambda\kappa.\kappa u^\circ$ (where u° is either x if u is a variable x or $(\lambda x.\mathbf{P}(t))$ if u is an abstraction $\lambda x.t$) and that, for all terms t and value u, $\mathbf{P}(t\,\{x \leftarrow u\}) = \mathbf{P}(t)\,\{x \leftarrow u^\circ\}$.

$$
\begin{aligned}
\mathbf{P}(x) &= \lambda\kappa.\kappa x\\
\mathbf{P}(\lambda x.t) &= \lambda\kappa.\kappa(\lambda x.\mathbf{P}(t))\\
\mathbf{P}(tu) &= \lambda\kappa.\mathbf{P}(t)(\lambda m.\mathbf{P}(u)(\lambda m'.mm'\kappa))\\
\mathbf{P}((\beta_v x.t)u) &= \lambda\kappa.(\beta\kappa'.\kappa'(\lambda x.\mathbf{P}(t)))(\lambda m.\mathbf{P}(u)(\lambda m'.mm'\kappa));\\
&\lambda\kappa.((\beta m.\mathbf{P}(u)(\lambda m'.mm'\kappa))(\lambda x.\mathbf{P}(t)));\\
&\lambda\kappa.(\mathbf{P}(u)(\lambda m'.(\beta x.\mathbf{P}(t))m'\kappa));\\
&\lambda\kappa.((\beta\kappa.\kappa u^\circ)(\lambda x.\mathbf{P}(t)\kappa));\\
&\lambda\kappa.(\beta x.\mathbf{P}(t)\kappa)u^\circ)\\
&:\lambda\kappa.(\lambda\kappa'.\kappa'(\lambda x.\mathbf{P}(t)))(\lambda m.\mathbf{P}(u)(\lambda m'.mm'\kappa)) \Rightarrow \lambda\kappa.(\mathbf{P}(t)\,\{x \leftarrow u^\circ\}\,\kappa)
\end{aligned}
$$

Figure 3.1: The CPS translation of call-by-value in call-by-name

So, $\mathbf{P}$ defines a morphism of **Cat**-operads $\Lambda_{v,\mathbb{T}} \to \Lambda_{\mathbb{T}}$ for all sets of types $\mathbb{T}$.

From call-by-name to call-by-value

3.1.3 In the other direction, the translation, given in Figure 3.2 is even arguably simpler. We remark that $\mathbf{P}(t\,\{x \leftarrow u\}) = \lambda\kappa.\mathbf{P}(t)\,\{x \leftarrow \mathbf{P}(u)\}\,\kappa$.

3.2 Embedding into intuitionnistic linear logic

The call-by-name Girard translation

$$\begin{aligned}
\mathbf{P}(x) =& \lambda\kappa.\kappa x\\
\mathbf{P}(\lambda x.t) =& \lambda\kappa.\kappa(\lambda x.\mathbf{P}(t))\\
\mathbf{P}(tu) =& \lambda\kappa.\mathbf{P}(t)(\lambda m.m\mathbf{P}(u)\kappa)\\
\mathbf{P}((\beta x.t)u) =& \lambda\kappa.(\beta_v\kappa'.\kappa'(\lambda x.\mathbf{P}(t)))(\lambda m.m\mathbf{P}(u)\kappa);\\
& \lambda\kappa.((\beta_v m.m\mathbf{P}(u)\kappa)(\lambda x.\mathbf{P}(t)));\\
& \lambda\kappa.((\beta_v x.\mathbf{P}(t))\mathbf{P}(u)\kappa)\\
& :\lambda\kappa.(\lambda\kappa'.\kappa'(\lambda x.\mathbf{P}(t)))(\lambda m.m\mathbf{P}(u)\kappa) \Rightarrow \lambda\kappa.\mathbf{P}(t)\,\{x \leftarrow \mathbf{P}(u)\}\,\kappa
\end{aligned}$$

Figure 3.2: The CPS translation of call-by-name in call-by-value

3.2.1 Linear Logic started as a decomposition of intuitionnistic logic, based on the type isomorphism

$$A \to B \simeq\ !A \multimap B,$$

meaning that intuitionistic implication is linear implication of a repetition. As such, for a set of types $\mathbb{T}_1$ suitable for intuitionnistic logic (that is, a set of types closed by a binary operation →) and a set of types $\mathbb{T}_2$ suitable for linear logic (that is, partitionned in two sub-set $\mathbb{T}_{\mathsf{l}}$ and $\mathbb{T}_{\mathsf{c}}$, closed by a binary operation $\multimap$ and a unary operation ! that maps elements of $\mathbb{T}_{\mathsf{l}}$ to elements of $\mathbb{T}_{\mathsf{c}}$) and a function

$$(\cdot)^\circ : \mathbb{T}_1 \to \mathbb{T}_{\mathsf{c}}$$

satisfying

$$\forall A, B \in \mathbb{T}_1, (A \to B)^\circ = (!A^\circ) \multimap B^\circ,$$

Girard's (call-by-name) embedding

$$\mathbf{G} : \Lambda_{\mathbb{T}_1} \longrightarrow \Lambda_{!,\mathbb{T}_2}$$

of the λ-calculus in intuitionnistic linear logic (Girard 1987) is defined as follows:

- on objects, $\mathbf{G}(A) := A^\circ$;
- on multimorphisms, given $\Gamma \vdash M : A_1, \dots, A_n \to B$, $\mathbf{G}(\Gamma \vdash M) \in \Lambda_{!,\mathbb{T}_2}(A^\circ{}_1, \dots, A^\circ{}_n; B^\circ)$ is defined by induction on M:
  - $\mathbf{G}(\Gamma, x \vdash x) := \Gamma, x \vdash x$;
  - $\mathbf{G}(\Gamma \vdash \lambda x.M) := \Gamma \vdash \lambda a.t\,[!x := a]$, where $\mathbf{G}(\Gamma, x \vdash M) = \Gamma, x \vdash t$;
  - $\mathbf{G}(\Gamma \vdash MN) := t!u$, where $\mathbf{G}(\Gamma \vdash M) = t$ and $\mathbf{G}(\Gamma \vdash N) = u$.

  It is straightforward to check that $\mathbf{G}(M\,\{N \leftarrow x\}) = \mathbf{G}(M)\,\{\mathbf{G}(N) \leftarrow x\}$.
- On 2-arrows, given

  $$(\beta x.\theta)\kappa : \Gamma \vdash (\lambda x.M)N \Rightarrow M'\,\{N' \leftarrow x\},$$

  $\mathbf{G}((\beta x.\theta)\kappa)$ is defined to be the following 2-step reduction:

  $$\begin{aligned}
  \mathbf{G}((\beta x.\theta)\kappa) =& (\beta a.\mathbf{G}(M)\,[!x := a])!\mathbf{G}(N); \beta\theta\,[!x := !\kappa]\\
  & :(\lambda a.\mathbf{G}(M)\,[!x := a])!\mathbf{G}(N) \to \mathbf{G}(M\,\{N \leftarrow x\})
  \end{aligned}$$

The call-by-value Girard translation

3.2.2 The call-by-value λ-calculus can also be encoded in linear logic (Maraist et al. 1999) by ways of a translation already present in (Girard 1987), where it is called the "boring translation". The resulting morphism $\mathbf{G}_v : \Lambda_v \to \Lambda_!$ is defined by:

- $\mathbf{G}_v(\mathsf{t}) = \mathsf{l}$ and $\mathbf{G}_v(\mathsf{v}) = \mathsf{c}$;
- on terms, $\mathbf{G}_v$ splits in two subcases, which we denote by $\mathbf{G}^{\mathsf{t}}_v$ (for terms) and $\mathbf{G}^{\mathsf{v}}_v$ (for values) and define by mutual induction: given $\mathrm{V} \in \Lambda_v(\mathsf{v}^n; \mathsf{v})$, $\mathbf{G}^{\mathsf{v}}_v(\mathrm{V}) \in \Lambda_!(\mathsf{c}^n; \mathsf{c})$ is
  - $\mathbf{G}^{\mathsf{v}}_v(x) := x$;
  - $\mathbf{G}^{\mathsf{v}}_v(\lambda x.\mathrm{M}) := \lambda a.\mathbf{G}_v(\mathrm{M})\,[!x := a]$

  and for $\mathrm{M} \in \Lambda_v(\mathsf{v}^n; \mathsf{t})$, $\mathbf{G}^{\mathsf{t}}_v(\mathrm{M}) \in \Lambda_!(\mathsf{c}^n; \mathsf{l})$ is
  - $\mathbf{G}^{\mathsf{t}}_v(\mathrm{V}) = !\mathbf{G}^{\mathsf{v}}_v(\mathrm{V})$
  - $\mathbf{G}^{\mathsf{t}}_v(\mathrm{MN}) = \xi\,[!\xi := \mathbf{G}^{\mathsf{t}}_v(\mathrm{M})]\,\mathbf{G}^{\mathsf{t}}_v(\mathrm{N})$

  One may check that $\mathbf{G}^{\mathsf{t}}_v(\mathrm{M}\{\mathrm{V}/x\}) = \mathbf{G}^{\mathsf{t}}_v(\mathrm{M})\{\mathbf{G}^{\mathsf{v}}_v(\mathrm{V})/x\}$.
- the 2-arrow $\beta_v : (\lambda x.\mathrm{M})\mathrm{V} \to \mathrm{M}\{\mathrm{V}/x\}$, $\mathbf{G}_v(\beta_v)$ is

$$\begin{aligned}\mathbf{G}_v((\lambda x.\mathrm{M})\mathrm{V}) &= \xi\,[!\xi := !\lambda a.\mathbf{G}_v(\mathrm{M})\,[!x := a]\,!\mathbf{G}_v(\mathrm{V})]\\ &\to \lambda a.\mathbf{G}_v(\mathrm{M})\,[!x := a]\,!\mathbf{G}_v(\mathrm{V}) \to \mathbf{G}_v(\mathrm{M})\,[!x := !\mathbf{G}_v(\mathrm{V})]\\ &\to \mathbf{G}_v(\mathrm{M})\{\mathbf{G}_v(\mathrm{V})/x\} = \mathbf{G}_v(\mathrm{M}\{\mathrm{V}/x\}).\end{aligned}$$

  which is enough, because β_v generates all 2-arrows of Λ_v.

3.3 Embedding into classical linear logic

Intuitionnistic to classical linear logic

3.3.1 The translation of intuitionnistic linear logic into classical logic is essentially vacuous, and is more of a change of syntax than anything. Figure 3.3 and Figure 3.5. The explicit-β step is not pictured and is a Contraction | Box from Figure 2.5(a) following the explicit-substitution translation.

3.3.2 The translation of the $\Lambda_!$-calculus we just presented does not validate the extensionality of λ-calculus, that is, the translations $\lambda x.(tx)$ and of t are not always reducible one to the other. In order for the translation to be extensional, we need to quotient MELL proof-net further, by making the ? cells commutative.

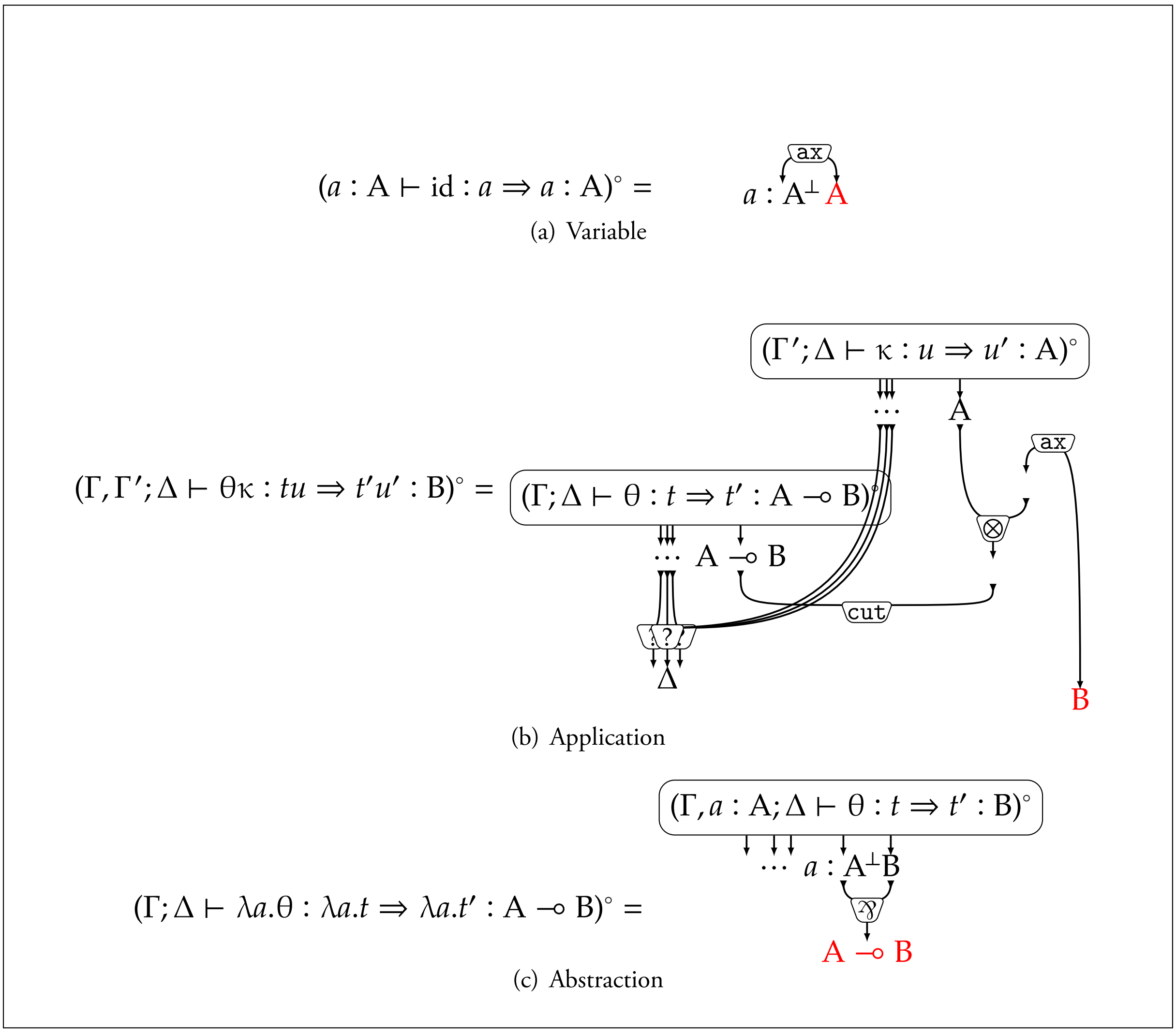
Definition 23 (*?-isomorphism of proof-nets*)

A **?-isomorphism of proof-nets** $f : \mathrm{R} \simeq_? \mathrm{R}'$ is

- an isomorphism $f : \tau_{\mathrm{R}} \to \tau_{\mathrm{R}'}$ of the underlying graph

- an isomorphism $f_{\mathsf{box}} : \mathscr{T}_{\mathrm{R}} \to \mathscr{T}_{\mathrm{R}'}$ of the boxing trees

such that


(a : A ⊢ id : a ⇒ a : A)° =
ax
a : A⊥ A
(Γ, Γ′; Δ ⊢ θκ : tu ⇒ t′u′ : B)° =
(Γ′; Δ ⊢ κ : u ⇒ u′ : A)°
A
ax
(Γ; Δ ⊢ θ : t ⇒ t′ : A ⊸ B)°
⊗
A ⊸ B
cut
?
Δ
B
(Γ, a : A; Δ ⊢ θ : t ⇒ t′ : B)°
a : A⊥ B
⅋
(Γ; Δ ⊢ λa.θ : λa.t ⇒ λa.t′ : A ⊸ B)° =
A ⊸ B

(a) Variable

(b) Application

(c) Abstraction

Figure 3.3: $\lambda_!$ calculus to MELL proof-nets, multiplicatives

$$\begin{tikzcd}
\tau_R \arrow[r, "\mathsf{box}_R"] \arrow[d, "f"] & \mathscr{T}_R^{\circlearrowright} \arrow[d, "f^{\circlearrowright}_{\mathsf{box}}"] \\
\tau_{R'} \arrow[r, "\mathsf{box}^{\circlearrowright}_{R'}"] & \mathscr{T}_{R'}^{\circlearrowright}
\end{tikzcd}$$

and such that, moreover, the restrictions of f on:

- the tails
- the inputs of each cell of type $\otimes, \parr$

are all increasing.

$$(\Gamma,\Gamma';\Delta,\Delta' \vdash (\beta a.\theta)\kappa : (\lambda a.t)u \Rightarrow t'\{a \leftarrow u'\} : B)^\circ$$

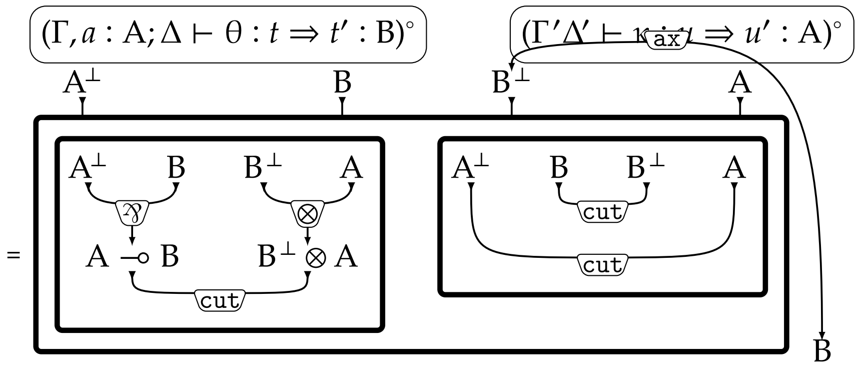

$(\Gamma, a : A; \Delta \vdash \theta : t \Rightarrow t' : B)^\circ$
$(\Gamma'\Delta' \vdash \kappa : u \Rightarrow u' : A)^\circ$
ax
$A^\perp$
B
$B^\perp$
A
=
$A^\perp$
B
$B^\perp$
A
⅋
⊗
$A \multimap B$
$B^\perp \otimes A$
cut
$A^\perp$
B
$B^\perp$
A
cut
cut
B

Figure 3.4: β-reduction

A $\simeq_?$-isomorphism class could be described directly by a definition close to the one of a proof-net, but with the order on flags kept partial and not defined on inputs of ? and ax cells, and neither on outputs of cut cells.

The Laurent-Girard translation

3.3.3 The Laurent-Girard translation translates λμ-calculus into some variant of MELL proof-nets (such as polarized MELL proof-nets). We represent it Figure 3.6, restricted on terms, as the translation of reductions is an adaptation of the translation of $\Lambda_!$ in $\mathsf{MELL}_?$.

3.4 Type systems

Type-systems as functors

3.4.1 What is a type system? (Melliès and Zeilberger 2015) recently suggested an amazingly simple answer to this question: a type system is a functor, mapping a category $\mathscr{D}$ of derivations to a monoid $\mathscr{C}$ of programs (in the simple case in which programs are untyped, otherwise $\mathscr{C}$ is also a category).

Typing reductions

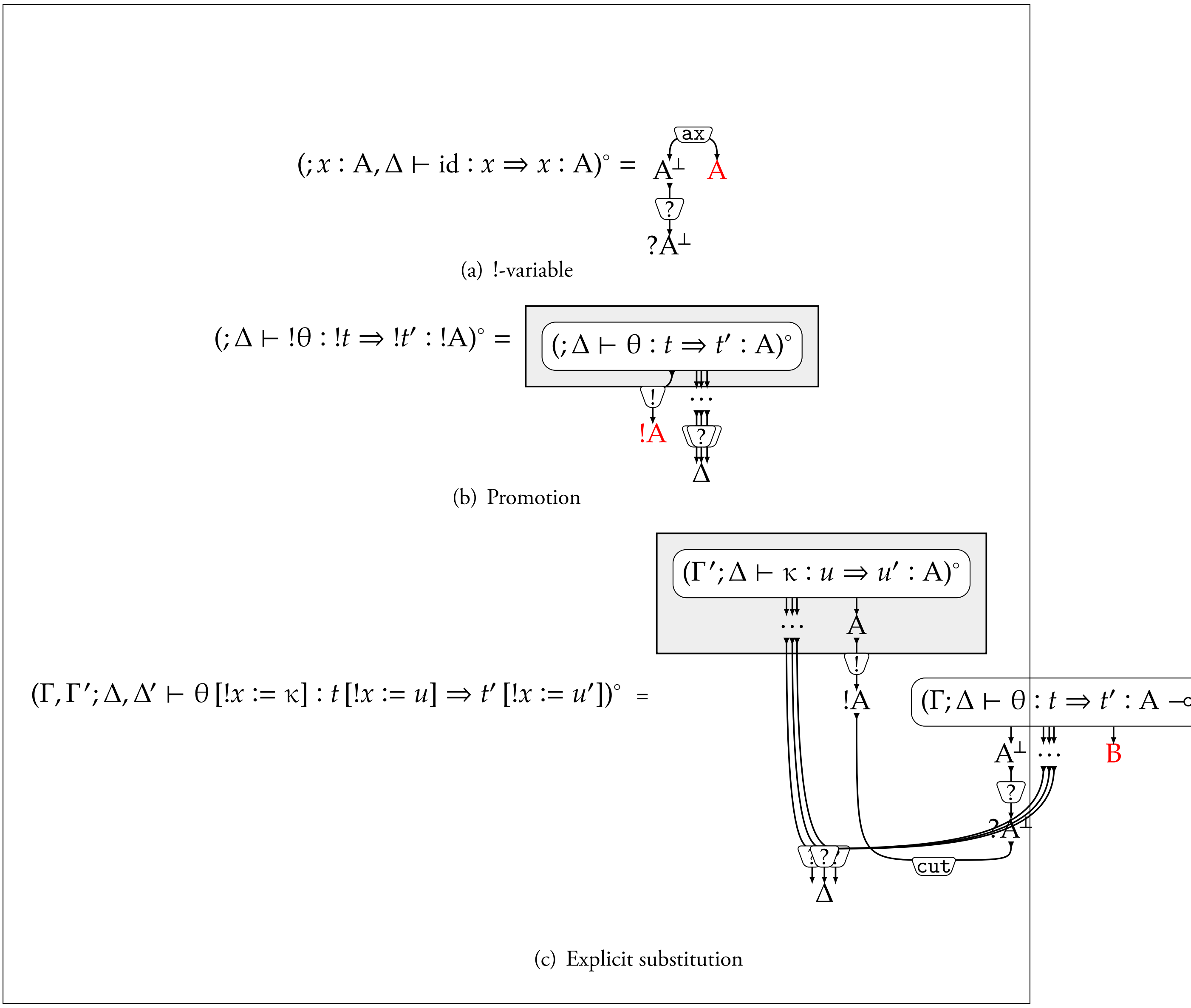
3.4.2 We introduce a class of morphisms of **Cat**-operads that are interesting as type systems.

Definition 24 (*(pointwise) Niefield fibration*)

Let $\mathscr{B}$ be a small category. A *Niefield fibration* on $\mathscr{B}$ is a functor $p : \mathscr{E} \to \mathscr{B}$, with $\mathscr{E}$ an arbitrary small category, verifying:

(identities) for every $b \in \mathscr{B}$ and arrow k of $\mathscr{E}$, $p(k) = \mathrm{id}_b$ implies $k = \mathrm{id}_e$ for some $e \in \mathscr{E}$;

(compositions) for every arrows k of $\mathscr{E}$ and f, f' of $\mathscr{B}$, $p(k) = f' \circ f$ implies that there exist


(; x : A, Δ ⊢ id : x ⇒ x : A)° =
ax
A⊥
A
?
?A⊥
(; Δ ⊢ !θ : !t ⇒ !t′ : !A)° =
(; Δ ⊢ θ : t ⇒ t′ : A)°
!
...
!A
?
Δ
(Γ, Γ′; Δ, Δ′ ⊢ θ [!x := κ] : t [!x := u] ⇒ t′ [!x := u′])° =
(Γ′; Δ ⊢ κ : u ⇒ u′ : A)°
...
A
!
!A
(Γ; Δ ⊢ θ : t ⇒ t′ : A ⊸
A⊥
...
B
?
?A⊥
?
cut
Δ

(a) !-variable

(b) Promotion

(c) Explicit substitution

Figure 3.5: $\lambda_!$ calculus to MELL proof-nets, exponentials

> (not necessarily unique) composable arrows g, g' of $\mathscr{E}$ such that $p(g) = f$ and $p(g') = f'$ and $k = g' \circ g$;
>
> We say that a morphism of small **Cat**-operads $p : \mathscr{E} \to \mathscr{B}$ is a *pointwise Niefield fibration* if, for all objects Γ, A of $\mathscr{E}$, the functor $p_{\Gamma;A}$ is a Niefield fibration.

The composition property has been studied by Niefield (2004) under the name *weak factorization lifting property*. For a Niefield fibration $p : \mathscr{E} \to \mathscr{B}$, the two properties together imply that, given an arrow f of $\mathscr{E}$, the structure of f (with regard to the compositions and the identities) is the same as that of $p(f)$.

In type-theoretic terms, pointwise Niefield fibrations correspond to type systems which

- do not type the untyped empty reduction with a non-empty reduction;

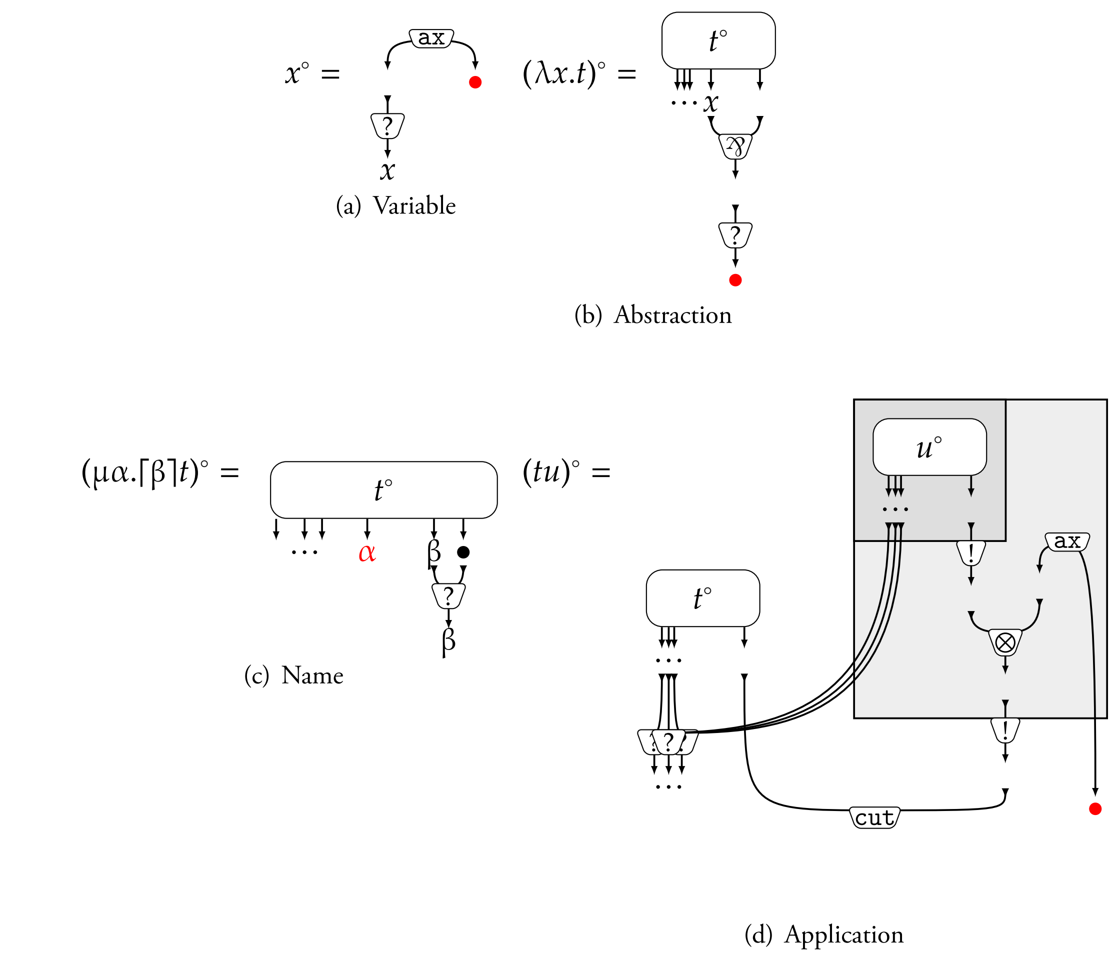


Figure 3.6: The Laurent translation of the λμ-calculus. A conclusion can me marked by the name or the variable it represents or by •, if it is the distinguished conclusion of a sub-net.

- type reductions "modularly": if a decomposable reduction is typed, then so are its components.

These seem to be reasonable requirements to ask of a type system. By the way, most common type systems do not even come with an explicit notion of "typing a reduction", so it does not even make sense to ask whether they comply with the above restrictions.

It is interesting to note that we have no similar requirements for terms, that is, the identity may be typed with a type of the form $A \to B$ ($A \neq B$), which corresponds to a case of subtyping, and we have no modularity requirements: if a term $t\{x/u\}$ is typed, we do not ask that t or u are typable.

Subject reduction, subject expansion

3.4.3 Of the properties of interest for type systems, the first ones are subject reduction and subject expansion.

A *weak discrete fibration* (or wd-fibration) is a Niefield fibration $p : \mathscr{E} \to \mathscr{B}$ which further satifies the *weak lifting property*: for every $f : b \to p(e')$ in $\mathscr{B}$, there exists $g : e \to e'$ in $\mathscr{E}$ such that $p(g) = f$.

A wd-opfibration has the dual property: for every $f : p(e) \to b'$ in $\mathscr{B}$, there exists $g : e \to e'$ in $\mathscr{E}$ such that $p(g) = f$. A wd-bifibration has both properties.

3.5 Semantics

3.5.1 Let $\mathscr{C}$ and $\mathscr{D}$ be two **Cat**-operads. If $\mathscr{C}$ is viewed as a calculus, but not $\mathscr{D}$, it is still possible to consider a morphism $f : \mathscr{C} \to \mathscr{D}$. In that case, if we view $\mathscr{D}$ as a mathematical universe, f associates

- to a type, a mathematical structure;
- to a term, a morphism of said structures;
- to a reduction, a transformation of these morphisms.

3.5.2 We can tame this structure by imposing restriction on what the reductions can be. For instance, it is possible to consider only **Cat**-operads $\mathscr{D}$ that are preorders locally, that is, for every objects $\mathrm{A}_1, \dots, \mathrm{A}_n$ and object A, the category $\mathscr{D}(\mathrm{A}_1, \dots, \mathrm{A}_n; \mathrm{A})$ is a pre-order.

Any partially ordered set $(\mathrm{S}, \leqslant)$ defines a **Cat**-operad $\mathscr{S}$:

- $\mathscr{S}$ has one object $\star$;
- a multimorphism $\star^n \to \star$ is a monotone function

$$\mathrm{S}^n \to \mathrm{S}$$

- the order between monotone functions of the same arity is the point-wise order: for $f, g : \star^n \to \star$,

$$f \leqslant g \iff \forall (x_1, \dots, x_n) \in \mathrm{S}^n, f(x_1, \dots, x_n) \leqslant g(x_1, \dots, x_n).$$

3.5.3 We will now introduce particular partially ordered sets whose associated **Cat**-operad is naturally an algebra for Λ, the *implicative structures*. They were introduced by Miquel to provide an uniform treatment of classical and intuitionnistic realizability. Without entering into details, implicative structures mix the interpretation of terms and of types, by allowing more flexibility in the structure of Heyting algebra.

Definition 25 (*implicative structure*)

An **implicative structure** $(\mathscr{A}, \leqslant, \to)$ is a complete lattice $(\mathscr{A}, \leqslant)$ endowed with a monotone application

$$(\to) : \mathscr{A}^{\mathrm{op}} \times \mathscr{A} \to \mathscr{A}$$

such that:

$$\forall a \in \mathscr{A}, \forall \mathrm{B} \subseteq \mathscr{A}, \bigwedge_{b \in \mathrm{B}} (a \to b) = a \to \bigwedge_{b \in \mathrm{B}} b$$

In particular, $\forall a \in \mathscr{A}, (a \to \top) = \top$.

Example : Implicative structures encompass many different structures of interest, including:

total Heyting algebras
total combinatory algebras
abstract Krivine structures

Proposition 1

Let $\mathscr{A}$ be an implicative structure and $\widehat{\mathscr{A}}$ the associated **Cat***-operad.*
The function f defined by:

- *for all objects* A *of* Λ, $f(\mathrm{A}) = \star$*;*
- *on the multi-arrows of* Λ*,*

$$\begin{aligned} f(\Gamma, x \vdash x) &= (a_1, \dots, a_n) \mapsto a_n : \star^n \to \star \\ f(\Gamma \vdash tu) &= (a_1, \dots, a_n) \mapsto \bigwedge \Big\{ c \in \mathscr{A}, f(\Gamma \vdash t)(a_1, \dots, a_n) \leqslant \big(f(\Gamma \vdash u)(a_1, \dots, a_n) \to c \big) \Big\} \\ f(\Gamma \vdash \lambda x.t) &= (a_1, \dots, a_n) \mapsto \bigwedge_{a \in \mathscr{A}} (a \to f(\Gamma, x \vdash t)(a_1, \dots, a_n, a)) \end{aligned}$$

where the sequents are written in unityped style, as the types are forgotten here.

- *as there are at most one 2-arrow between arrows in $\widehat{\mathscr{A}}$, f is already determined on 2-arrows*

is a morphism of **Cat***-operads*

$$f : \Lambda \to \widehat{\mathscr{A}}.$$

Proof : The only thing to prove is that if there exists a 2-arrow $\theta : t \Rightarrow t'$ in Λ, the image of t is less than the image of t'. ■

3.5.4 We can go a step further and ask for the reductions to be trivial. Every denotational model of the λ-calculus is an algebra for $\Lambda_\star$.

4 Linear approximations

❦

4.1 Linear approximation of the λ-calculus of linear logic

The polyadic λ-calculus

4.1.1 We fix two disjoint, countably infinite sets of *linear* and *polyadic* variables, ranged over by a, b, c and x, y, z, respectively. As usual, we also fix a set of types $\mathbb{T}$, closed by a binary operation $\multimap$, and, for each arity n, an n-ary operation $\langle\rangle$. We will approximate the ! box constructor of linear λ-calculus by a polyadic term, paving the way to linear approximations.

Polyadic terms are terms defined by Figure 4.1. Terms are considered up to renaming of variables, bound by λ and $\cdot\,[\langle\cdot\rangle := \cdot]$.

4.1.2 Four flavors of calculi are obtained by including or excluding the structural rules of weakening and contraction: a term is

- *affine* if, modulo Barendregt's convention, every polyadic variable appears in it at most once. This is achieved by allowing only weakening;
- *relevant* if, for every of its subterms of the form $t\,[\langle x_1, \dots, x_n\rangle := u]$, each x_i appears free in t. This is achieved by allowing only contraction;
- *linear* if it is both affine and relevant. This is achieved by allowing neither.

Unconstrained terms are also called *cartesian*, which is achieved by allowing both rules.

We want to stress that the type system of Figure 4.1 is far from original: if one erases all term annotations, one obtains a natural deduction formulation of a logical system such that

$$\frac{}{a : \mathrm{A}; \Delta \vdash a : a \Rightarrow a : \mathrm{A}}\,\text{(variable)}$$

$$\frac{\Gamma; \Delta \vdash \theta : t \Rightarrow t' : \mathrm{A} \qquad \Gamma'; \Delta' \vdash \theta' : t' \Rightarrow t'' : \mathrm{A}}{\Gamma, \Gamma'; \Delta, \Delta' \vdash \theta; \theta' : t \Rightarrow t'' : \mathrm{A}}\,\text{(composition)}$$

$$\frac{\Gamma; \Delta \vdash \theta : t \Rightarrow t' : \mathrm{A} \multimap \mathrm{B} \qquad \Gamma'; \Delta' \vdash \kappa : u \Rightarrow u' : \mathrm{A}}{\Gamma, \Gamma'; \Delta, \Delta' \vdash \theta\kappa : tu \Rightarrow t'u' : \mathrm{B}}\,\text{(application)}$$

$$\frac{\Gamma, a : \mathrm{A}; \Delta \vdash \theta : t \Rightarrow t' : \mathrm{B}}{\Gamma; \Delta \vdash \lambda a.\theta : \lambda a.t \Rightarrow \lambda a.t' : \mathrm{A} \multimap \mathrm{B}}\,\text{(abstraction)}$$

$$\frac{\begin{array}{rcl} \Gamma, a : \mathrm{A}; \vec{y} : \vec{\mathrm{C}}, \Delta & \vdash & \theta : t \Rightarrow t' : \mathrm{B} \\ \Gamma'; \Delta' & \vdash & \kappa : u \Rightarrow u' : \mathrm{A} \end{array} \qquad \Gamma_i; \Delta_i \vdash \zeta_i : v_i \Rightarrow v_i' : \mathrm{C}_i}{\Gamma, \Gamma', \Gamma_i; \Delta, \Delta', \Delta_i \vdash (\beta a.\theta)\left[\langle\vec{y}\rangle := \vec{\zeta}\right]\kappa : (\lambda a.t)\left[\langle\vec{y}\rangle := \vec{v}\right]u \Rightarrow t'\,\{a \leftarrow u'\}\left[\langle\vec{y}\rangle := \vec{v'}\right] : \mathrm{B}}\,\text{(β-reduction)}$$

$$\frac{}{; x : \mathrm{A}, \Delta \vdash x : x \Rightarrow x : \mathrm{A}}\,\text{(!-variable)}$$

$$\frac{\Gamma_1; \vdash \theta_1 : t_1 \Rightarrow t_1' : \mathrm{A}_1 \quad \cdots \quad \Gamma_n; \vdash \theta_n : t_n \Rightarrow t_n' : \mathrm{A}_n}{\Gamma_1 \ldots \Gamma_n; \vdash \langle\theta_1, \ldots, \theta_n\rangle : \langle t_1, \ldots, t_n\rangle \Rightarrow \langle t_1', \ldots, t_n'\rangle : \langle\mathrm{A}_1, \ldots, \mathrm{A}_n\rangle}\,\text{(promotion)}$$

$$\frac{\Gamma; x_1 : \mathrm{A}_1, x_n : \mathrm{A}_n, \Delta \vdash \theta : t \Rightarrow t' : \mathrm{B} \qquad \Gamma'; \vdash \kappa : u \Rightarrow u' : \langle\mathrm{A}_1, \ldots, \mathrm{A}_m\rangle}{\Gamma, \Gamma'; \Delta \vdash \theta\,[\langle x_1, \ldots, x_n\rangle := \kappa] : t\,[\langle x_1, \ldots, x_n\rangle := u] \Rightarrow t'\,[\langle x_1, \ldots, x_n\rangle := u'] : \mathrm{B}}\,\text{(explicit)}$$

$$\frac{\begin{array}{rcl} \Gamma; x_1 : \mathrm{A}_1, x_n : \mathrm{A}_n, \Delta & \vdash & \theta : t \Rightarrow t' : \mathrm{B} \\ \Gamma'; \vec{y} : \vec{\mathrm{C}}, \Delta' & \vdash & \kappa : \langle u_1, \ldots, u_m\rangle \Rightarrow \langle u_1', \ldots, u_m'\rangle : \langle\mathrm{A}_1, \ldots, \mathrm{A}_m\rangle \\ \Gamma''; & \vdash & \zeta : v \Rightarrow v' : \langle\mathrm{C}_1, \ldots, \mathrm{C}_k\rangle \end{array}}{\begin{array}{rl} \Gamma, \Gamma'; \Delta, \Delta' & \vdash \beta\theta\left[\langle x_1, \ldots, x_n\rangle := \kappa\left[\langle\vec{y}\rangle := \vec{\zeta}\right]\right] \\ & : t\left[\langle x_1, \ldots, x_n\rangle := u\left[\langle\vec{y}\rangle := \vec{v}\right]\right] \Rightarrow t'\left\{x_i \leftarrow u_j'\right\}\left[\langle\vec{y}\rangle := \vec{v'}\right] : \mathrm{B} \end{array}}\,\text{(explicit-β)}$$

Figure 4.1: Polyadic simple types derivations.

- if one reads $\langle\mathrm{A}_1, \ldots, \mathrm{A}_n\rangle$ as $\mathrm{A}_1 \otimes \cdots \otimes \mathrm{A}_n$, all rules except weakening and contraction are derivable in the $\otimes/\multimap$ fragment of multiplicative linear logic;
- if one reads $\langle\mathrm{A}_1, \ldots, \mathrm{A}_n\rangle$ as $(\mathrm{A}_1 \,\&\, 1) \otimes \cdots \otimes (\mathrm{A}_n \,\&\, 1)$, then weakening too is derivable in multiplicative additive linear logic;
- if one reads $\langle\mathrm{A}_1, \ldots, \mathrm{A}_n\rangle$ as $!\mathrm{A}_1 \otimes \cdots \otimes !\mathrm{A}_n$, then every rule is derivable in multiplicative exponential linear logic.

The simply-typed polyadic calculus are obtained as term calculi for these fragments of linear logic, in the standard Curry-Howard sense.

$$\frac{\Gamma;\Delta \vdash \theta : t \Rightarrow t' : A}{\sigma_1\Gamma;\sigma_2\Delta \vdash \theta : t \Rightarrow t' : A}\text{(exchange)}$$

$$\frac{\Gamma;\Delta \vdash t : C}{\Gamma;x : A,\Delta \vdash t : C}\text{(weakening)}$$

$$\frac{\Gamma;\Delta,x : A,y : A \vdash t : C}{\Gamma;\Delta,x : A \vdash t\left\{y \leftarrow x\right\} : C}\text{(contraction)}$$

Figure 4.2: The structural rules

4.1.3 We are ready to define four operads, each corresponding to a flavor of polaydic calculi.

Definition 26

The **Cat**-operad $\mathbf{Poly}_{\mathbb{T}}$ of (cartesian) polyadic terms is defined as follows:

- as objects the elements of $\mathbb{T}$;
- as arrows

$$\Upsilon \longrightarrow B$$

where Υ is an interleaving of Γ (containing only linear types) and Δ (containing only cartesian types), β-less terms $\Gamma;\Delta \vdash t : B$ of Figure 4.1;

- as composition substitution and renaming;
- as 2-arrows between two terms the reductions;
- as composition of 2-arrows, composition ;.

We will, when stressing the importance of it, write $\mathbf{CartPoly}_{\mathbb{T}}$ this **Cat**-operad.

The **Cat**-operads $\mathbf{AffPoly}_{\mathbb{T}}$, $\mathbf{RelPoly}_{\mathbb{T}}$ and $\mathbf{LinPoly}_{\mathbb{T}}$ are the sub-**Cat**-operads of $\mathbf{CartPoly}_{\mathbb{T}}$ that are restricted to terms and reductions that do not use a particular structural rule. The inclusions are schematised Figure 4.3.

$$\begin{tikzcd}
\mathbf{LinPoly}_{\mathbb{T}} \arrow[r] \arrow[d] & \mathbf{RelPoly}_{\mathbb{T}} \arrow[d] \\
\mathbf{AffPoly}_{\mathbb{T}} \arrow[r] & \mathbf{CartPoly}_{\mathbb{T}}
\end{tikzcd}$$

Figure 4.3: The inclusions of operads of polyadic calculi

4.1.4 As usual, we single out two sets of types $\mathbb{T}$ and the **Cat**-operads attached to them:

- the terminal set, that only contains two elements l and c. We denote the associated **Cat**-operads **AffPoly**$_\star$, **RelPoly**$_\star$, **LinPoly**$_\star$ and **CartPoly**$_\star$.
- the set freely generated by $\multimap$ and $\langle\rangle$ and an infinite set of atomic terms. We denote the associated **Cat**-operads **AffPoly**, **RelPoly**, **LinPoly** and **CartPoly**.

There are obvious morphisms from the simply-typed versions to their terminal versions, which we view as type systems.

Proposition 2

Every multimorphism of **Poly** *is strongly normalizable.*

Proof: As observed above, **Poly** may be embedded in propositional multiplicative exponential linear logic, whose strong normalization is well known. ■

It is worth mentioning that, for **LinPoly** or **AffPoly**, Proposition 2 is actually immediate because linear or affine polyadic terms strongly normalize even without types: the absence of duplication makes the size of terms strictly decrease at every reduction step. In that case, the only property ensured by simple types is that typed terms cannot get stuck (this is proved as customary).

4.1.5 We can remark that, given a context Γ and a type A, the set **Poly**$(\Gamma;A)$ can be endowed with a partial order that makes it a lattice.

It lacks some limits, so it is not a dcpo. We will return to this subject in Chapter 8.

Approximations of the λ-calculus

4.1.6 Although it can be studied on its own, the polyadic λ-calculus just presented is mainly interesting for approximating λ-calculus and λ-calculus reductions. The approximation relation $\sqsubset$ is defined inductively on the structure of the λ-calculus reduction (which justifies our decision to describe the reductions by a term language) and is the same for all flavors of approximation, as represented Figure 4.4, where we omit the types, for brevity: indeed, the approximation relation is defined on sequents $\Gamma;\Delta \vdash \rho : t \Rightarrow t' : A \sqsubset \Gamma' \vdash \pi : T \Rightarrow T' : A'$. We suppose given a function from the polyadic types $\mathbb{T}_{\mathsf{p}}$ to the linear types $\mathbb{T}_{\mathsf{l}}$ and all the judgements are understood closed with a context coherent with this function.

4.1.7 Let us consider a pure λ-term of $\Lambda_{!,\star}$ $M : \star^n \to \star$. For a given list of types Γ of length n and a type A, we can consider the set $\mathscr{M}$ of polyadic terms of type $\Gamma \to A$ that approximates M. This set $\mathscr{M}$ is naturally endowed with an order relation, defined by induction on the term:

$$\frac{}{a \leqslant a} \qquad \frac{}{x \leqslant x} \qquad \frac{t \leqslant t' \qquad u \leqslant u'}{tu \leqslant t'u'} \qquad \frac{t_1 \leqslant t_1' \qquad \cdots t_n \leqslant t_n'}{\langle t_1, \dots, t_n\rangle \langle t_1', \dots, t_n', \dots, t_m\rangle}$$

The ordered set $(\mathscr{M}, \leqslant)$ falls short of being a dcpo, and thus a model of the λ-calculus. We will come back to this shortcoming Section 8.1.

4.2 Linear approximations of proof-nets

Polyadic proof-nets

4.2.1 The analog of polyadic λ-terms are polyadic proof-nets.

$$\frac{}{a : a \Rightarrow a \sqsubset a : a \Rightarrow a}\,\text{(variable)}$$

$$\frac{\rho : t \Rightarrow t' \sqsubset \pi : \mathrm{T} \Rightarrow \mathrm{T}' \qquad \rho' : t' \Rightarrow t'' \sqsubset \pi' : \mathrm{T}' \Rightarrow \mathrm{T}''}{\rho;\rho' : t \Rightarrow t'' \sqsubset \pi;\pi' : \mathrm{T} \Rightarrow \mathrm{T}''}\,\text{(composition)}$$

$$\frac{\rho : t \Rightarrow t' \sqsubset \pi : \mathrm{T} \Rightarrow \mathrm{T}' \qquad \rho' : u \Rightarrow u' \sqsubset \pi' : \mathrm{U} \Rightarrow \mathrm{U}'}{\rho\rho' : tu \Rightarrow t'u' \sqsubset \pi\pi' : \mathrm{TU} \Rightarrow \mathrm{T}'\mathrm{U}'}\,\text{(application)}$$

$$\frac{\rho : t \Rightarrow t' \sqsubset \pi : \mathrm{T} \Rightarrow \mathrm{T}'}{\lambda a.\rho : \lambda a.t \Rightarrow \lambda a.t' \sqsubset \lambda a.\pi : \lambda a.\mathrm{T} \Rightarrow \lambda a.\mathrm{T}'}\,\text{(abstraction)}$$

$$\frac{\rho : t \Rightarrow t' \sqsubset \pi : \mathrm{T} \Rightarrow t' \qquad \kappa : u \Rightarrow u' \sqsubset \chi : \mathrm{U} \Rightarrow \mathrm{U}' \qquad \zeta_i : v_i \Rightarrow v_i' \sqsubset \theta_i : \mathrm{V}_i \Rightarrow \mathrm{V}_i}{\begin{array}{ll} & (\beta a.\rho)\left[\langle\vec{y}\rangle := \vec{\zeta}\right]\kappa : (\lambda a.t)\left[\langle\vec{y}\rangle := \vec{v}\right]u \Rightarrow t'\,\{a \leftarrow u'\}\left[\langle\vec{y}\rangle := \vec{v'}\right] \\ \sqsubset & (\beta a.\pi)\left[\langle\vec{y}\rangle := \vec{\theta}\right]\chi : (\lambda a.\mathrm{T})\left[\langle\vec{y}\rangle := \vec{\mathrm{V}}\right]\mathrm{U} \Rightarrow \mathrm{T}'\,\{a \leftarrow \mathrm{U}'\}\left[\langle\vec{y}\rangle := \vec{\mathrm{V}'}\right] \end{array}}\,\text{(β-reduction)}$$

$$\frac{}{x_i : x_i \Rightarrow x_i \sqsubset x : x \Rightarrow x}\,\text{(!-variable)}$$

$$\frac{\rho_1 : t_1 \Rightarrow t_1' \sqsubset \pi : \mathrm{T} \Rightarrow \mathrm{T}' \quad \cdots \quad \rho_n : t_n \Rightarrow t_n' \sqsubset \pi : \mathrm{T} \Rightarrow \mathrm{T}'}{\langle\rho_1, \dots, \rho_n\rangle : \langle t_1, \dots, t_n\rangle \Rightarrow \langle t_1', \dots, t_n'\rangle \sqsubset\, !\pi : !\mathrm{T} \Rightarrow !\mathrm{T}'}\,\text{(promotion)}$$

$$\frac{\rho : t \Rightarrow t' \sqsubset \pi : \mathrm{T} \Rightarrow \mathrm{T}' \qquad \kappa : u \Rightarrow u' \sqsubset \chi : \mathrm{U} \Rightarrow \mathrm{U}'}{\begin{array}{ll} & \rho\,[\langle x_1, \dots, x_n\rangle := \kappa] : t\,[\langle x_1, \dots, x_n\rangle := u] \Rightarrow t'\,[\langle x_1, \dots, x_n\rangle := u'] \\ \sqsubset & \pi\,[!x := \chi] : \mathrm{T}\,[!x := \mathrm{U}] \Rightarrow \mathrm{T}'\,[!x := \mathrm{U}'] \end{array}}\,\text{(explicit)}$$

$$\frac{\begin{array}{l} \rho : t \Rightarrow t' \sqsubset \pi : \mathrm{T} \Rightarrow \mathrm{T}' \\ \kappa : \langle u_1, \dots, u_m\rangle \Rightarrow \langle u_1', \dots, u_m'\rangle \sqsubset \chi : \mathrm{U} \Rightarrow \mathrm{U}' \\ \zeta_i : v_i \Rightarrow v_i' \sqsubset \theta_i : \mathrm{V}_i \Rightarrow \mathrm{V}_i \end{array}}{\begin{array}{ll} & \beta\rho\left[\langle x_1, \dots, x_n\rangle := \kappa\left[\langle\vec{y}\rangle := \vec{\zeta}\right]\right] : t\left[\langle x_1, \dots, x_n\rangle := u\left[\langle\vec{y}\rangle := \vec{v}\right]\right] \Rightarrow t'\left\{x_i \leftarrow u_i'\right\}\left[\langle\vec{y}\rangle := \vec{v'}\right] \\ \sqsubset & \beta\rho\left[!x := \kappa\left[!\vec{y} := \vec{\zeta}\right]\right] : \mathrm{T}\left[!x := \mathrm{U}\left[!\vec{y} := \vec{\mathrm{V}}\right]\right] \Rightarrow \mathrm{T}'\,\{x \leftarrow \mathrm{U}'\}\left[!\vec{y} := \vec{\mathrm{V}'}\right] \end{array}}\,\text{(explicit-β)}$$

Figure 4.4: The polyadic approximations of λ-calculus reductions

Definition 27 (*polyadic proof-net*)

A *polyadic module* is a labelled ordered oriented colored graph $(\tau, \ell, \mathsf{c}, \leqslant)$ (the orientation is left implicit) such that:

- $\ell : \mathrm{V}_\tau \to \{\mathtt{ax}, \mathtt{cut}, \mathbf{1}, \bot, \otimes, \parr, ?, !\}$;
- $\mathsf{c} : \mathrm{F}_\tau \to \mathscr{F}_{\mathsf{MELL}}$;
- let $v \in \mathrm{V}_\tau$.
  - if $\ell(v) = \mathtt{ax}$, the corolla τ_v has only two flags i_1 and i_2 which are inputs, and such that
  $$\mathsf{c}(i_1) = \mathsf{c}(i_2)^\perp;$$
  - if $\ell(v) = \mathtt{cut}$, the corolla τ_v has only two flags o_1 and o_2 which are outputs, and such that
  $$\mathsf{c}(o_1) = \mathsf{c}(o_2)^\perp;$$
  - if $\ell(v) = \mathbf{1}$, the corolla τ_v has no inputs and on output o, such that
  $$\mathsf{c}(o) = \mathbf{1};$$
  - if $\ell(v) = \bot$, the corolla τ_v has no inputs and on output o, such that
  $$\mathsf{c}(o) = \bot;$$
  - if $\ell(v) = \otimes$, the corolla τ_v has two inputs $i_1 < i_2$ and one output o, such that
  $$\mathsf{c}(o) = \mathsf{c}(i_1) \otimes \mathsf{c}(i_2);$$
  - if $\ell(v) = \parr$, the corolla τ_v has two inputs $i_1 < i_2$ and one output o, such that
  $$\mathsf{c}(o) = \mathsf{c}(i_1) \parr \mathsf{c}(i_2);$$
  - if $\ell(v) = ?$, the corolla τ_v has many inputs $i_1, \dots, i_n (n \leqslant 0)$ and one output o, such that
  $$\forall 1 \leqslant j \leqslant n, \mathsf{c}(o) = ?\mathsf{c}(i_j);$$
  - if $\ell(v) = !$, the corolla τ_v has many inputs $i_1, \dots, i_n (n \leqslant 0)$ and one output o, such that
  $$\forall 1 \leqslant j \leqslant n, \mathsf{c}(o) = !\mathsf{c}(i_j);$$

  These corollas are depicted Figure 4.5.
- $\leqslant$ is total.

A **resource proof-net** is a module with no input tails.

4.2.2 There are reductive polyadic proof-nets, too, depicted in Figure 4.6.

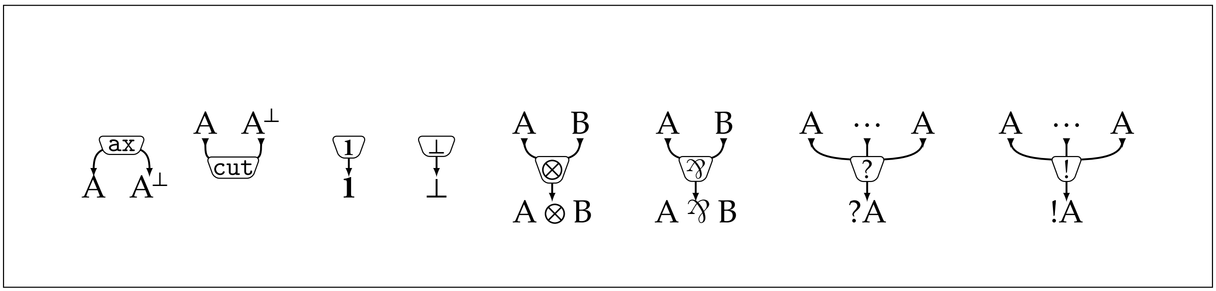


Figure 4.5: Resource linear logic cells

Definition 28

> A **reduction cell** is a an oriented labelled colored corolla satisfying:
>
> - the cell is labelled with a couple (m_1, m_2) of MELL-modules together with two increasing bijections between the sets of inputs (respectively outputs) of m_1 and m_2 that commute with c;
> - the inputs (respectively outputs) of the reduction cells are in bijection with the inputs (respectively outputs) of m_1, and the bijection commutes with c.

Definition 29

> A *reductive resource module* is a tuple $R = (\tau, \ell, c, \leqslant)$, where τ is a labelled ordered oriented colored graph $(\tau, \ell, c, \leqslant)$, the *underlying graph* of R such that:
>
> - $\ell : V_\tau \to \{\mathtt{ax}, \mathtt{cut}, \mathbf{1}, \perp, \otimes, ⅋, ?, !, \mathsf{Red}\}$;
> - $c : F_\tau \to \mathscr{F}_{\mathsf{MELL}}$.
>
> It is endowed with two projections, whose underlying graph is obtained by grafting the projections of the reduction cells with the rest of the graph.
>
> We require the two projected graphs to be resource modules.
>
> A **reductive proof-net** is a reductive module without inputs.

Remark 6

> All the MELL reduction cells without internal boxes are also polyadic reduction cells.

4.2.3 Proof-nets have a tree structure that is made explicit through their boxing function. We actually define the linear approximations on these tree structures, and we will pull the approximation back to the underlying graph.

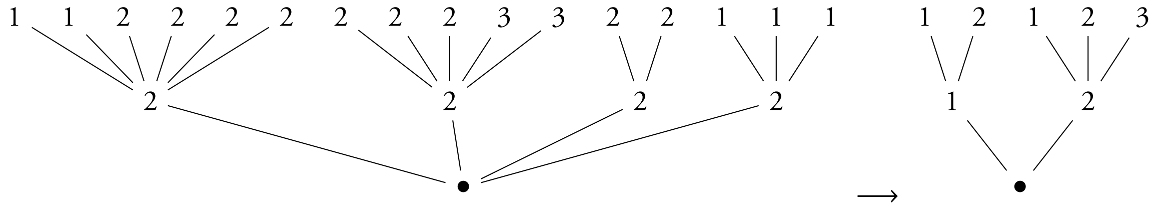
Given a rooted tree $\mathscr{T}$, it is possible to define a subtree with many copies of each node of $\mathscr{T}$ (Boudes 2009).

Definition 30 (*thick subtree*)

> A **thick subtree** of a rooted tree τ is a rooted tree σ together with a morphism of rooted trees $\sigma \to \tau$.

Example : The left tree σ is a thick sub-tree of the right one τ. The nodes of τ are numbered, and the one of

σ are marked with their image through the morphism of tree.



Definition 31 (*proto-Taylor expansion*)

Let R be a proof-net.

The set of thick subtrees of its box-tree is called the **proto-Taylor** expansion of R and noted $\mathscr{T}_{\mathrm{R}}^{\mathsf{proto}}$.

Example : These are elements of the proto-Taylor expansion of Figure 2.3.

4.2.4 The point of defining the expansion of trees is to pull back this expansion to proof-nets.

Definition 32

Let R be a proof-net and $\mathscr{T}$ its box-tree.

Let $p \in \mathscr{T}_{\mathrm{R}}^{\mathsf{proto}}$ be an element of the proto-Taylor expansion of R. The t-expansion of R is the pullback:

$$\begin{tikzcd}
\mathrm{R}_t \arrow[r] \arrow[d] \arrow[dr, phantom, "\lrcorner", very near start] & t^{\circlearrowleft} \arrow[d, "p^{\circlearrowleft}"] \\
\mathrm{R} \arrow[r, "\mathsf{box}"] & \mathscr{T}^{\circlearrowleft}
\end{tikzcd}$$

computed in the category of ordered oriented graphs.

4.2.5 Let t be an element of the proto-Taylor expansion of a proof-net R. R_t inherits a structure of labelled colored graph, by composition with the graph morphism $\mathrm{R}_t \to \mathrm{R}$.

Given this structure, one can check that R_t is a polyadic proof-net.

Definition 33 (*polyadic Taylor expansion*)

Let R be a proof-net. The **Taylor expansion** of R is the set

$$\mathscr{T}_{\mathrm{R}} = \{\mathrm{R}_t \mid t \in \mathscr{T}_{\mathrm{R}}^{\mathsf{proto}}\}$$

of polyadic proof-nets.

4.2.6 The element of the Taylor expansion that takes exactly one copy of each box is particular: it is a version of the original MELL proof-net, but without the boxes.

Lemma 3

Let R *be a* MELL *proof-net, $t \in \mathscr{T}_{\mathrm{R}}^{\mathsf{proto}}$ such that the morphism of trees $t \to \mathscr{T}$ is an isomorphism.*

> *The projection* $\mathrm{R}_t \to \mathrm{R}$ *is an isomorphism of labelled colored oriented graphs.*

Approximations of the reduction

4.2.7 The MELL reduction cells are not all polyadic reduction cells. As such, for the pullback of a reductive MELL proof-net and a thick subtree of its boxing tree to be a reductive polyadic proof-net, we need to define the action, on a reductive MELL cell c, of a thick subtree of the boxtree of the left projection of c.

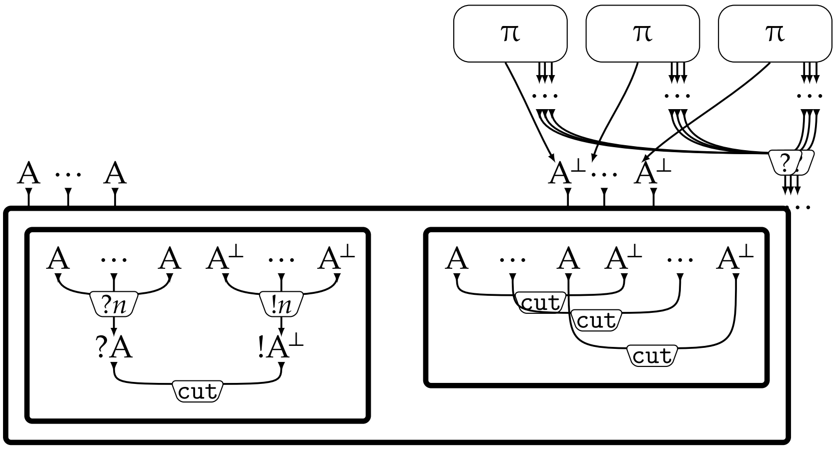
Remark 7

> Contrarily to what we do with proof-nets, the reductive proof-nets are not computed via a pullback. It is because one reduction cell is actually a pair of MELL modules, and for each of them, a pullback is to be computed.

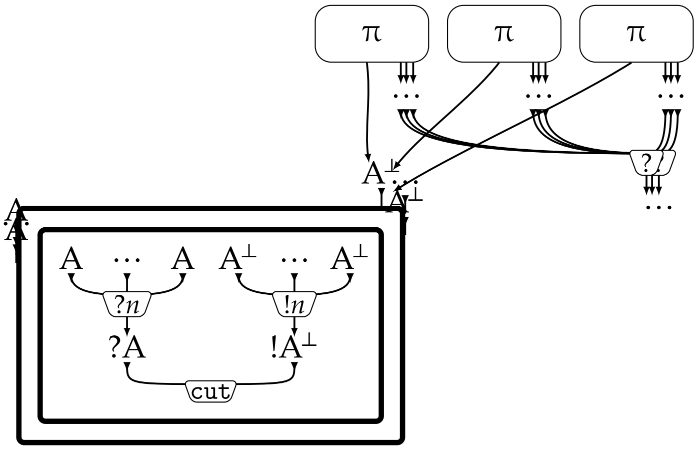
Definition 34 (*expansion of a reduction cell*)

Let c be a MELL reductive cell, and $\mathscr{T}$ be the box-tree of its left projection. Let τ be a thick subtree $\tau \to \mathscr{T}$ of $\mathscr{T}$. The *expansion* of c along τ is

- if c is of type *Box* | *Contraction* (Figure 2.5(a)) with its contraction of arity n and the box inside c called π, if π has n edges in τ with same image in $\mathscr{T}$,



- if c is of type *Box* | *Contraction* (Figure 2.5(a)) with its contraction of arity n and the box inside c called π, if π has a number of edges in τ with same image in $\mathscr{T}$ m different of n,



and so, its right projection is collapsed to the empty polyadic proof-net.

- if c is of type *Box* | *Box* (Figure 2.5(b)), and the arity of π and π' in τ are compatible, the expansion of c is the identity.

Let R be a MELL reductive proof-net, and let $\mathscr{T}$ be the box-tree of the left projection of R. Let τ be a thick subtree $\tau \to \mathscr{T}$ of $\mathscr{T}$. The **expansion** of R along τ is the reductive proof-net obtained by grafting:

- the pullback of the restriction of R to its non-reductive cells and its cells without internal boxes;
- for each reduction cell c with an internal box, the expansion of c along the sub-tree of τ corresponding to π.

Quotients and the Taylor expansion

4.2.8 We only defined the Taylor expansion of a MELL proof-net. We can check that, given an element t of the proto-Taylor expansion of a proof-net R, $t \leftarrow \mathrm{R}_t \to \mathrm{R}$ can be quotiented by $\simeq_?$.

Definition 35 (*?-equivalence*)

A **?-isomorphism of polyadic proof-nets** $f : \mathrm{R} \simeq_? \mathrm{R}'$ is an isomorphism $f : \tau_\mathrm{R} \to \tau_{\mathrm{R}'}$ of the underlying graph such that, moreover, the restrictions of f on:

- the tails
- the inputs of each cell of type $\otimes, ⅋, !$

are all increasing.

4.2.9 It is not the case, if by analogy with ?, we also quotient !.

Definition 36 (*!?-equivalence*)

A **!?-isomorphism of polyadic proof-nets** $f : \mathrm{R} \simeq_? \mathrm{R}'$ is an isomorphism $f : \tau_\mathrm{R} \to \tau_{\mathrm{R}'}$ of the underlying graph such that, moreover, the restrictions of f on:

- the tails
- the inputs of each cell of type $\otimes, ⅋$

are all increasing. A **resource proof-net** is an $\simeq_{!?}$-equivalence class of polyadic proof-nets.

Definition 37 (*resource Taylor expansion*)

The resource Taylor expansion of a MELL proof-net R is defined as the set

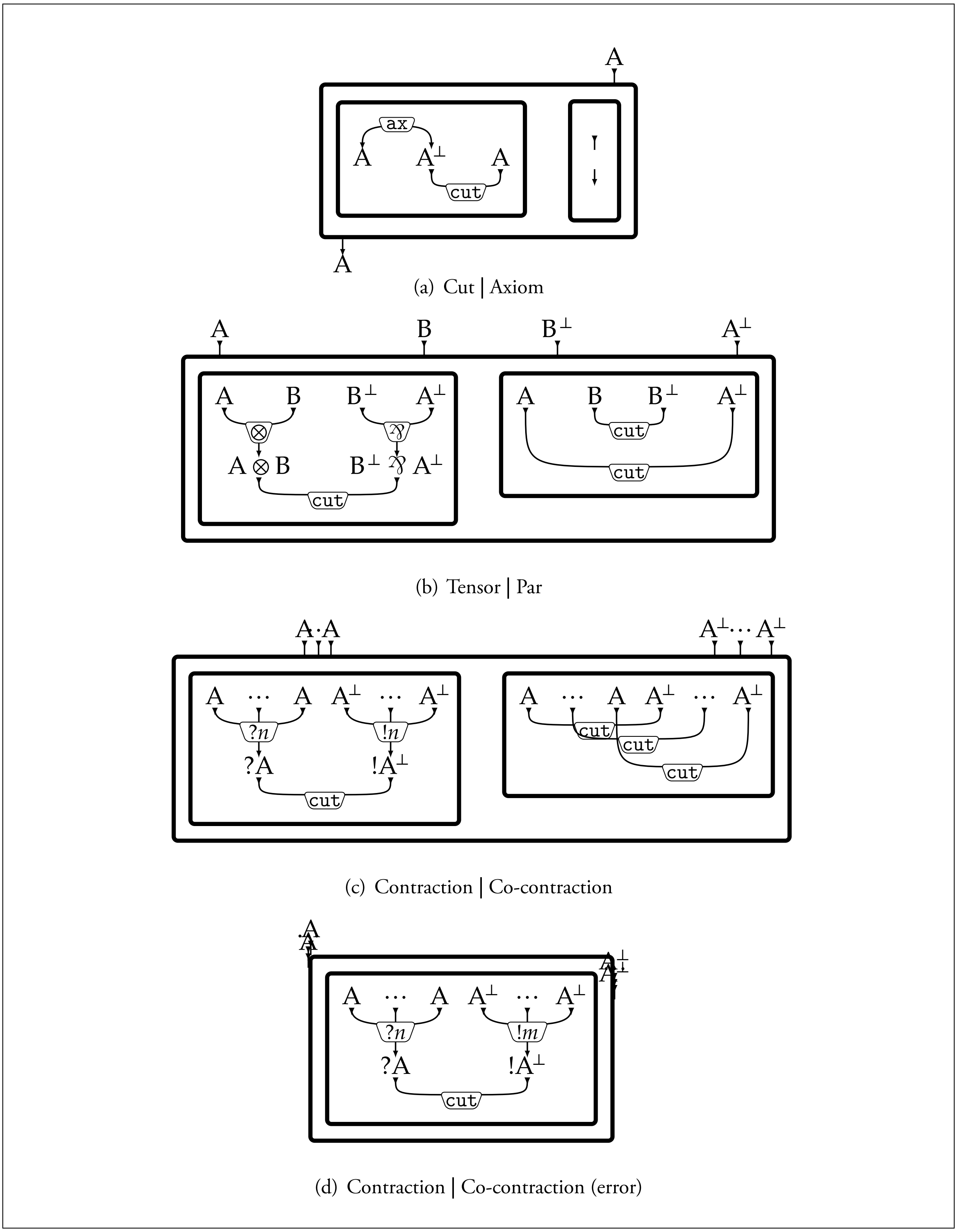
$$\mathscr{T}_\mathrm{R}^{!?} = \{[t] \mid t \in \mathscr{T}_\mathrm{R}\}$$

and the same for R a MELL$_?$ proof-net.

4.2.10 It is indispensable to note that the polyadic Taylor expansion and the resource Taylor expansion are fundamentally different. An element of the Taylor expansion is a polyadic proof-net endowed with two projections, while an element of the quotiented Taylor expansion is a resource proof-net.

The non-quotiented Taylor expansion, consisting of polyadic proof-net, with their rigidity and their determinism, will be our main tool to study the resource Taylor expansion, which, alone, has the innocence that makes it the analogue of the polyadic (or resource) Taylor expansion of λ-terms.

Indeed, the polyadic Taylor expansion contains objects that are already formatted for their use. We will come back to their study in Section 7.1.



(a) Cut | Axiom

(b) Tensor | Par

(c) Contraction | Co-contraction

(d) Contraction | Co-contraction (error)

Figure 4.6: The polyadic reduction cells

5

Approximations and intersection types

5.1 The Grothendieck construction

The Grothendieck construction for discrete fibrations

5.1.1 The Grothendieck construction gives an equivalence between discrete fibrations overs a base category $\mathscr{B}$ and presheaves on $\mathscr{B}$. Let's unravel the definitions.

For pedagogical purposes, we first present a particular case, that will be generalized in § 5.1.7.

Definition 38 (*discrete fibration*)

Let $\mathscr{B}$ be a small category.

A **discrete fibration** over $\mathscr{B}$ is a functor

$$p : \mathscr{C} \to \mathscr{B},$$

where $\mathscr{C}$ is any small category, such that, for all object $c : \mathscr{C}$, and arrow $f : b' \to pc : \mathscr{B}$ there exists a unique $g : c' \to c : \mathscr{C}$ such that $pg = f$.

The category of discrete fibration over $\mathscr{B}$ is written $\mathbf{DiscFib}(\mathscr{B})$.

Definition 39 (*presheaf*)

Let $\mathscr{B}$ be a small category.

A **presheaf** on $\mathscr{B}$ is a functor $\mathscr{B} \to Set^{\mathrm{op}}$. The category of presheaves on $\mathscr{B}$ is noted $\widehat{\mathscr{B}}$.

Theorem 3

There is an equivalence of categories

$$\int : \widehat{\mathscr{B}} \simeq \mathbf{DiscFib}(\mathscr{B}) : \partial$$

Proof: This is a corollary of Theorem 5. ■

Distributors

5.1.2 Functors can be generalized, in the same way that functions can be generalized to relations.

Definition 40 (*distributor*)

Let $\mathscr{V}$ be a monoidal category. Let $\mathscr{C}$ and $\mathscr{D}$ be two categories.

A $\mathscr{V}$-enriched **distributor** D from $\mathscr{C}$ to $\mathscr{D}$, noted

$$\mathrm{D} : \mathscr{C} \nrightarrow \mathscr{D}$$

is a functor

$$\mathrm{D} : \mathscr{C} \times \mathscr{D}^{\mathrm{op}} \to \mathscr{V}.$$

Example: Let $\mathscr{C}$ be a category enriched over a monoidal category $\mathscr{V}$. The functor

$$\begin{aligned} \mathrm{Hom} : &\mathscr{C} \times \mathscr{C}^{\mathrm{op}} \to \mathscr{V} \\ &(c, c') \mapsto \mathscr{C}(c', c) \end{aligned}$$

is a distributor

$$\mathrm{Hom} : \mathscr{C} \nrightarrow \mathscr{C}$$

Remark 8

The terminology on distributors is unclear.

When first defining them, Bénabou called them *profunctors* as they are more than functors. The name is unfortunate, as the prefix 'pro' often denotes the pro-completion of a category, i.e. the collection of all formal cofiltered limits of objects of a category, and *profunctors* are not objects of the pro-completion of some category.

He later switched to *distributors*, following an analogy with functional analysis: just as all functions define a distribution, and some distributions are representable by a function, all functors define a distributor, and some distributors are representable by a functor.

> Other names used are bi-modules, making the above example paradigmatic; or stress the status of distributors as generalized relations and call them *correspondences* or even *relators* (Loregian 2015).

As we will only use **Set**-enriched distributors, we will only give the next definitions and propositions in the **Set**-enriched case, although they generalize.

5.1.3 In the same way as relations, distributors compose.

Definition 41 (*(co-)end*)

Let $\mathscr{C}$ and $\mathscr{E}$ be two categories. Let $\mathrm{H} : \mathscr{C}^{\mathrm{op}} \times \mathscr{C} \to \mathscr{E}$ be a functor.
Every objects $c, c' : \mathscr{C}$ and arrow $f : c \to c' : \mathscr{C}$ induces a contravariant arrow

$$f^* : \mathrm{H}(c', c') \to \mathrm{H}(c, c')$$

and a covariant arrow

$$f_* : \mathrm{H}(c, c) \to \mathrm{H}(c, c').$$

The **end** of H, noted $\int_{c:\mathscr{C}} \mathrm{H}$ is the universal object endowed with projections (in $\mathscr{E}$) $\int_{c:\mathscr{C}} \mathrm{H} \to \mathrm{H}(c, c)$, for all objects $c : \mathscr{C}$, making the diagram

$$\begin{tikzcd}
\int_{c:\mathscr{C}} \mathrm{H} \arrow[r] \arrow[d] & \mathrm{H}(c', c') \arrow[d, "f^*"] \\
\mathrm{H}(c, c) \arrow[r, "f_*"] & \mathrm{H}(c, c')
\end{tikzcd}$$

commute, for all $f : c \to c' : \mathscr{C}$.
Dually, the **co-end** of H, noted $\int^{c:\mathscr{C}} \mathrm{H}$ is the universal object endowed with projections (in $\mathscr{E}$) $\mathrm{H}(c, c) \to \int^{c:\mathscr{C}} \mathrm{H}$, for all objects $c : \mathscr{C}$, making the diagram

$$\begin{tikzcd}
\mathrm{H}(c, c') \arrow[r, "f_*"] \arrow[d, "f^*"] & \mathrm{H}(c', c') \arrow[d] \\
\mathrm{H}(c, c) \arrow[r] & \int^{c:\mathscr{C}} \mathrm{H}
\end{tikzcd}$$

Although co-ends are also denoted by an integral sign, they have nothing to do with the Grothendieck construction applied to a functor.

Theorem 4 (*Fubini theorem for (co-)ends*)

Let $\mathscr{C}$, $\mathscr{D}$ and $\mathscr{E}$ be three small categories and

$$\mathrm{T} : \mathscr{C}^{\mathrm{op}} \times \mathscr{C} \times \mathscr{D}^{\mathrm{op}} \times \mathscr{D} \to \mathscr{E}$$

be a functor. If all sides exist, there are canonical isomorphisms:

$$\int_{c:\mathscr{C}}\int_{d:\mathscr{D}} \mathrm{T}(c,c,d,d) \simeq \int_{(c,d):\mathscr{C}\times\mathscr{D}} \mathrm{T}(c,c,d,d) \simeq \int_{d:\mathscr{D}}\int_{c:\mathscr{C}} \mathrm{T}(c,c,d,d)$$

And the same for co-ends:

$$\int^{c:\mathscr{C}}\int^{d:\mathscr{D}} \mathrm{T}(c,c,d,d) \simeq \int^{(c,d):\mathscr{C}\times\mathscr{D}} \mathrm{T}(c,c,d,d) \simeq \int^{d:\mathscr{D}}\int^{c:\mathscr{C}} \mathrm{T}(c,c,d,d)$$

5.1.4 Alas, a (co-)end is not defined unequivocally: as it is a universal object, it is only defined modulo isomorphism. As such, the composition of distributors can not be associative, and there can not exist a category of small categories and distributors.

We need to relax the definition of a category, to allow for composition and identities to hold only up to isomorphism.

Definition 42 (*bicategory*)

A **bicategory** $\mathscr{C}$ is the data of:

- a class C_0 of *objects*;
- for all couples (A,B) of objects, a category
$$\mathscr{C}(\mathrm{A};\mathrm{B})$$
of morphisms from A to B.
- for every objects A, B, C, a functor
$$\mathscr{C}(\mathrm{A};\mathrm{B}) \times \mathscr{C}(\mathrm{B};\mathrm{C}) \to \mathscr{C}(\mathrm{A};\mathrm{C})$$
- for every object A, a distinguished morphism
$$\mathrm{id}_\mathrm{A} \in \mathscr{C}(\mathrm{A};\mathrm{A})$$
- for each triple $f:\mathrm{A}\to\mathrm{B}$, $g:\mathrm{B}\to\mathrm{C}$, $h:\mathrm{C}\to\mathrm{D}$, a natural isomorphism
$$\alpha_{h,g,f} : h\circ(g\circ f) \Rightarrow (h\circ g)\circ f$$
- for each morphism $f:\mathrm{A}\to\mathrm{B}$, natural isomorphisms
$$\lambda_f : \mathrm{id}_\mathrm{B}\circ f \Rightarrow f$$
$$\rho_f : f\circ \mathrm{id}_\mathrm{A} \Rightarrow f$$
that satisfy the pentagon and triangle coherence diagrams.

Definition 43 (*bicategory of distributors*)

Given three small categories $\mathscr{C},\mathscr{D},\mathscr{E}$, and two distributors $\mathrm{F}:\mathscr{C} \nrightarrow \mathscr{D}$, $\mathrm{G}:\mathscr{D}\nrightarrow\mathscr{E}$, we define the composite $G \circ F : \mathscr{C} \nrightarrow \mathscr{E}$ by:

$$\forall c \in \mathscr{C}, \forall e \in \mathscr{E}, G \circ F(e,c) = \int^{d:\mathscr{D}} G(e,d) \times F(d,c)$$

The composition is associative up to isomorphism (due to Fubini theorem for co-ends and the lax associativity of the cartesian product in **Set**). As such, we can define the bicategory **Dist** of distributors in the following way:

- objects are small categories;
- morphisms are distributors with the aforementioned composition;
- 2-morphisms are natural transformation (of the underlying functors).

5.1.5 The bicategory **Dist** enjoys a close relationship with the bicategory **Cat** of small categories (whose arrows are functors and 2-arrows natural transformations). Indeed, any functor $F : \mathscr{C} \to \mathscr{D}$ induce a pair of adjoints distributors $F_* \dashv F^*$ in the following way:

$$F_* : \mathscr{C} \nrightarrow \mathscr{D} \text{ defined by } F_*(d,c) = \mathscr{D}(d, Fc)$$
$$F^* : \mathscr{D} \nrightarrow \mathscr{C} \text{ defined by } F^*(c,d) = \mathscr{D}(c, Fd)$$

Moreover, the functor $\cdot \mapsto (\cdot)_* : \mathbf{Cat} \to \mathbf{Dist}$ is locally fully faithful, that is,

$$\mathbf{Cat}(f,g) \simeq \mathbf{Dist}(f_*, g_*)$$

This situation between **Cat** and **Dist** is called a *pro-arrow equipment* (Wood 1982). It is a actually the prototypical exemple.

It is also possible to describe the same situation with double categories.

5.1.6 The bicategory **Dist** has a monoidal structure, inherited from **Set**.

The Grothendieck construction for Cat*-operads*

5.1.7 In order to define an analogue of the Grothendieck construction, building an equivalence between some functors on a base **Cat**-operad $\mathscr{B}$ and type systems (that is, Niefield fibrations) over $\mathscr{B}$, we need to consider morphisms for type systems. Niefield fibrations and these morphisms are organized in a bioperad, which is a relaxation of the notion of **Cat**-operad, replacing some identities with natural isomorphisms.

Definition 44 (*bioperad*)

A (small) **bioperad** $\mathscr{C}$ is given by the following:

- a set $\mathscr{C}_0$ of *objects*;
- for every objects $C_1, \dots, C_n, A$, a category $\mathscr{C}(C_1, \dots, C_n; A)$ whose objects are called *multimorphisms* and whose morphisms are called 2-*arrows*;
- for every object A, sequence of objects $\Delta := B_1, \dots, B_n$ and sequences of objects $\Gamma_1, \dots, \Gamma_n$, a functor $\circ_{\Gamma_1,\dots,\Gamma_n;\Delta;A}$ from $\mathscr{C}(\Delta; A) \times \mathscr{C}(\Gamma_1; B_1) \times \cdots \times \mathscr{C}(\Gamma_n; B_n)$ to $\mathscr{C}(\Gamma_1, \dots, \Gamma_n; A)$;
- for every object A, a multimorphism $\mathrm{id}_A \in \mathscr{C}(A; A)$;

- for each $n \in \mathbf{N}$, $\sigma \in \mathfrak{S}_n$, object A and objects $\Gamma := C_1, \dots, C_n$, a functor $\mathsf{exch}_{\sigma}^{\Gamma;A} : \mathscr{C}(C_1, \dots, C_n; A) \to \mathscr{C}(C_{\sigma^{-1}(1)}, \dots, C_{\sigma^{-1}(n)}; A)$ such that $\mathsf{exch}_{\sigma'}^{\sigma^{-1}(\Gamma);A} \circ \mathsf{exch}_{\sigma}^{\Gamma;A} = \mathsf{exch}_{\sigma' \circ \sigma}^{\Gamma;A}$;

such that the composition functors satisfy the obvious associativity, neutrality and compatibility laws with respect to exch up to (coherent) isomorphism.

A *lax morphism* of bioperads commute with the operations up to (coherent) natural transformations in the right-to-left direction, which are not necessarily invertible.

Definition 45 (*pointwise relational morphism*)

A *pointwise relational morphism* F between two pointwise Niefield fibrations $p_1 : \mathscr{E}_1 \to \mathscr{B}$ and $p_2 : \mathscr{E}_2 \to \mathscr{B}$ is

- a relation $F \subseteq \mathscr{E}_1 \times \mathscr{E}_2$ between the objects of $\mathscr{E}_1$ and the objects of $\mathscr{E}_2$;
- for every list $\Gamma_1 = B_1^1 \cdots B_1^n$ of objects of $\mathscr{E}_1$ and object A_1 of $\mathscr{E}_1$, and for every list $\Gamma_2 = B_2^1 \cdots B_2^n$ of objects of $\mathscr{E}_2$ and object A_2 of $\mathscr{E}_2$, such that

$$\forall 1 \leq i \leq n, (B_1^i, B_2^i) \in F \text{ and } (A_1, A_2) \in F,$$

  a relation $F_{\Gamma_2,A_2}^{\Gamma_1,A_1} \subseteq \mathscr{E}_1(\Gamma_1; A_1) \times \mathscr{E}_2(\Gamma_2; A_2)$ such that:

  - $(e_1, e_2) \in F_{\Gamma_2,A_2}^{\Gamma_1,A_1}$ implies $p_1(\Gamma_1; A_1)(e_1) = p_2(\Gamma_2; A_2)(e_2)$;
  - for every 2-arrow $f : b \to b'$ of $\mathscr{B}$ and every $e_1 \in \mathscr{E}_1$ such that $p_1(\Gamma_1; A_1)(e_1) = b$ and $e_2' \in \mathscr{E}_2$ such that $p_2(\Gamma_2; A_2)(e_2') = b'$, the following conditions are equivalent:
    - ∗ there exists $g_2 : e_2 \to e_2'$ such that $p_2(\Gamma_2; A_2)(g_2) = f$ and $(e_1, e_2) \in R$;
    - ∗ there exists $g_1 : e_1 \to e_1'$ such that $p_1(\Gamma_1; A_1)(g_1) = f$ and $(e_1', e_2') \in R$.

Pointwise Niefield fibrations over $\mathscr{B}$ and pointwise relational morphisms between them form a category, which we denote $\mathbf{NieFib}(\mathscr{B})$.

Interpreted as a morphism of type systems, a pointwise relational morphism ensures a property of inter-typability between the two systems it is a morphism of.

5.1.8 Let $\mathfrak{Dist}$ be the following (large) bioperad:

- objects are small categories;
- multimorphisms $A_1 \dots A_n \nrightarrow B$ are distributor-valued distributors, that is functors (of bicategories)

$$F : A_1 \times \cdots \times A_n \times B^{\mathrm{op}} \to \mathbf{Dist}_s,$$

  where $A_1, \dots, A_n, B$ are trivially viewed as bicategories, and $\mathbf{Dist}_s$ is the bicategory whose objects are semi-categories, arrows distributors between them and 2-arrows natural transformations;
- composition of multimorphisms is defined as the composition of distributors;
- 2-arrows $\theta : F \Rightarrow G : A_1 \dots A_n \to B$ are natural transformations of the underlying functors: a family of **Set**-valued functions indexed by $A_1 \times \cdots \times A_n \times B^{\mathrm{op}}$:

$$\forall (a_1, \dots, a_n, b) \in A_1 \times \cdots \times A_n \times B^{\mathrm{op}}, \theta_{a_1,\dots,a_n,b} : F(a_1, \dots, a_n, b) \times G(a_1, \dots, a_n, b) \to \mathbf{Set}$$

  satisfying naturality conditions:

$$\begin{tikzcd}
\mathrm{F}(a_1, \dots, a_n, b) \times \mathrm{G}(a_1, \dots, a_n, b) \arrow[rr, "\mathrm{F}(f_1, \dots, f_n, f) \times \mathrm{G}(f_1, \dots, f_n, f)"] \arrow[dr, "\theta_{a_1, \dots, a_n, b}"] & & \mathrm{F}(a'_1, \dots, a'_n, b') \times \mathrm{G}(a'_1, \dots, a'_n, b' \arrow[dl, "\theta_{a'_1, \dots, a'_n, b'}"] \\
& \mathbf{Set} &
\end{tikzcd}$$

5.1.9 Let now $\mathfrak{Dist}_*$ be the following (large) bioperad:

- objects are small pointed categories, that is couples (A, a) where A is a small category and $a : \mathrm{A}$;
- multimorphisms $(\mathrm{A}_1, a_1) \dots (\mathrm{A}_n, a_n) \nrightarrow (\mathrm{B}, b)$ are pointed distributor-valued distributors (F, f), that is functors
$$\mathrm{F} : \mathrm{A}_1 \times \cdots \times \mathrm{A}_n \times \mathrm{B}^{\mathrm{op}} \to \mathbf{Dist}_s,$$
together with a point $f : \mathrm{F}(a_1, \dots, a_n, b)$.
- composition of multimorphisms is defined as the composition of distributors: given
$$(\mathrm{G}, g) : (\mathrm{B}_1, b_1) \dots (\mathrm{B}_m, b_m) \nrightarrow (\mathrm{C}, c),$$
$$(\mathrm{F}, f) : (\mathrm{A}_1, a_1) \dots (\mathrm{A}_n, a_n) \nrightarrow (\mathrm{B}_i, b_i),$$
the composite $(\mathrm{G}, g) \circ_i (\mathrm{F}, f)$ is defined as the couple $(\mathrm{G} \circ_i \mathrm{F}, (g, f))$ ((g, f) here denotes the image of (g, f) in the coend defining $\mathrm{G} \circ_i \mathrm{F}(b_1, \dots, b_{i-1}, a_1, \dots, a_n, b_{i+1}, \dots, b_m; c)$);
- 2-arrows $\theta : (\mathrm{F}, f) \Rightarrow (\mathrm{G}, g) : (\mathrm{A}_1, a_1) \dots (\mathrm{A}_n, a_n) \to (\mathrm{B}, b)$ are natural transformations of the underlying functors such that, moreover,
$$(f, g) \in \theta_{a_1, \dots, a_n, b}.$$

There is an obvious forgetful functor $\mathrm{U} : \mathfrak{Dist}_* \to \mathfrak{Dist}$.

5.1.10 Let us recall the definition of lax natural transformation between two lax functors $\theta : \mathrm{F} \Rightarrow \mathrm{G} : \mathscr{B} \to \mathfrak{Rel}$:

- for each A in $\mathscr{B}$ a distributor $\theta_\mathrm{A} : \mathrm{FA} \nrightarrow \mathrm{GA}$;
- for each $f : \mathrm{A}_1 \dots \mathrm{A}_n \to \mathrm{B}$ in $\mathscr{B}$, a 2-arrow $\theta_f : \mathrm{G}f \circ (\theta_{\mathrm{A}_1}, \dots, \theta_{\mathrm{A}_n}) \Rightarrow \theta_\mathrm{B} \circ \mathrm{F}f$, that is a family of relations indexed on the objects of $\mathrm{FA}_1 \times \cdots \times \mathrm{FA}_n \times \mathrm{GB}^{\mathrm{op}}$,
$$\forall (a_1, \dots, a_n, b) \in \mathrm{FA}_1 \times \cdots \times \mathrm{FA}_n \times \mathrm{GB}^{\mathrm{op}}, (\theta_f)_{a_1, \dots, a_n, b} \subseteq (\mathrm{G}f \circ \theta_\mathrm{A})(a_1, \dots, a_n; b) \times (\theta_\mathrm{B} \circ \mathrm{F}f)(a_1, \dots, a_n;$$
that satisfy naturality conditions.

We say that a lax natural transformation is *relational* if, for every object A, the distributor θ_A is a relation, that is, it is valued in a subsingleton.

Theorem 5 (*Grothendieck construction*)

> *Let $\mathscr{B}$ be a small* **Cat***-operad. The category* **NieFib**$\mathscr{B}$ *is equivalent to the category* $\mathfrak{Dist}^{\mathscr{B}}$ *of lax morphisms* $\mathscr{B} \to \mathfrak{Dist}$ *and lax relational natural transformations.*

Proof: $\int : \mathfrak{Dist}^{\mathscr{B}} \to \mathbf{NieFib}(\mathscr{B})$ is defined by:

- given a lax functor $\mathrm{F} : \mathscr{B} \to \mathfrak{Dist}$, $\int \mathrm{F}$ is the pullback of the forgetful functor $\mathrm{U} : \mathfrak{Dist}_* \to \mathfrak{Dist}$ along F. More explicitly, we denote by $\mathscr{E}\ell(\mathrm{F})$ the pullback category:
  - the objects are pairs (B, x), where B is an object of $\mathscr{B}$ and $x : \mathrm{FB}$;
  - a multimorphism $(\mathrm{B}_1, b_1) \dots (\mathrm{B}_n, b_n) \to (\mathrm{B}', b')$ is a pair (f, a) where $f : \mathrm{B}_1 \dots \mathrm{B}_n \to \mathrm{B}'$ is a multimorphism in $\mathscr{B}$ and $a \in \mathrm{F}f(b_1, \dots, b_n, b')$;
  - given

    $$\begin{aligned}(g, b) &: (\mathrm{B}_1, b_1) \dots (\mathrm{B}_m, b_m) \to (\mathrm{C}, c),\\ (f, a) &: (\mathrm{A}_1, a_1) \dots (\mathrm{A}_n, a_n) \to (\mathrm{B}_i, b_i),\end{aligned}$$

    the composite $(g, b) \circ_i (f, a)$ is the couple $(g \circ_i f, (b, a))$.
  - a 2-arrow $\theta : (f, a) \Rightarrow (g, a') : (\mathrm{A}_1, a_1) \dots (\mathrm{A}_n, a_n) \to (\mathrm{B}, b)$ is a family of **Set**-valued functions indexed by $\mathrm{A}_1 \times \dots \times \mathrm{A}_n \times \mathrm{B}$:

    $$\forall(\alpha^1, \dots, \alpha^n, \beta) \in \mathrm{A}_1 \times \dots \times \mathrm{A}_n \times \mathrm{B}, \theta_{\alpha^1, \dots, \alpha^n, \beta} \subseteq \mathrm{F}(\alpha^1, \dots, \alpha^n, \beta) \times \mathrm{G}(\alpha^1, \dots, \alpha^n, \beta).$$

    such that

    $$\theta_{a_1, \dots, a_n, b}(a, a') \neq \varnothing.$$

  $\int \mathrm{F}$ is just the first projection. It is a pointwise Niefield fibration.
- Given a relational lax natural transformation $\theta : \mathrm{F} \Rightarrow \mathrm{G}$, we define the relation $\int \theta$ from the objects of $\mathscr{E}\ell\mathrm{F}$ to the objects of $\mathscr{E}\ell(\mathrm{G})$ by $((b, x), (b', y)) \in \left(\int \theta\right)$ if $b = b'$ and $(x, y) \in \theta_b$. The relational morphism of $\mathscr{E}\ell\mathrm{F}$ to the morphisms of $\mathscr{E}\ell\mathrm{G}$ by:

  $$\left(\int \theta\right)_{\Gamma;(\mathrm{B},b)} : \mathscr{E}\ell\mathrm{F}(\Gamma; (\mathrm{B}, b)) \to \mathscr{E}\ell\mathrm{G}((\mathrm{A}_1, \theta_{\mathrm{A}_1} a_1), \dots, (\mathrm{A}_n, \theta_{\mathrm{A}_n} a_n); (\mathrm{B}, \theta_{\mathrm{B}} b))$$

  for every list $\Gamma = (\mathrm{A}_1, a_1), \dots, (\mathrm{A}_n, a_n)$ of objects of $\mathscr{E}\ell\mathrm{F}$ and object (B, b) of $\mathscr{E}\ell\mathrm{F}$, and list $\Gamma' = (\mathrm{A}_1, a'_1), \dots, (\mathrm{A}_n, a'_n)$ of objects of $\mathscr{E}\ell\mathrm{G}$ and object (B, b') of $\mathscr{E}\ell\mathrm{G}$ the relation $\left(\int \theta\right)^{\Gamma;(\mathrm{B},b)}_{\Gamma';(\mathrm{B}',b')}$ is defined, for

  $$\begin{aligned}(f, d) &: (\mathrm{A}_1, a_1), \dots, (\mathrm{A}_n, a_n) \to (\mathrm{B}, b)\\ (g, d') &: (\mathrm{A}_1, a'_1), \dots, (\mathrm{A}_n, a'_n) \to (\mathrm{B}, b')\end{aligned}$$

  by $((f, d), (g, d')) \in \left(\int \theta\right)^{\Gamma;(\mathrm{B},b)}_{\Gamma';(\mathrm{B}',b')}$ if $(a_1, a'_1) \in \theta_{\mathrm{A}_1}, \dots, (a_n, a'_n) \in \theta_{\mathrm{A}_n}, (b, b') \in \theta_{\mathrm{B}}$, and $\varnothing$ else. Hence $\int \theta$ is a pointwise relational morphism.

∂ : **NieFib**$(\mathscr{B}) \to \mathfrak{Dist}^{\mathscr{B}}$ is defined by:

- given a pointwise Niefield fibration $p : \mathscr{E} \to \mathscr{B}$, we set, for $b : \mathscr{B}$,
  - $\partial p(\mathrm{B})$ equal to the semi-category whose objects are the pre-images of B through p and the morphims are the pre-images of the identity over B through p;
  - for $f : \mathrm{B}_1 \dots \mathrm{B}_n \to \mathrm{B}$ in $\mathscr{B}$, $\partial p(f)$ is the distributor $\partial p(f) : p^{-1}(\mathrm{B}_1) \times \dots \times p^{-1}(\mathrm{B}_n) \nrightarrow p^{-1}(\mathrm{B})$ defined by:

    $$\forall(e_1, \dots, e_n, e) \in p^{-1}(\mathrm{B}_1) \times \dots \times p^{-1}(\mathrm{B}_n) \times p^{-1}(\mathrm{B})^{\mathrm{op}}, \partial p(f)(e_1, \dots, e_n; e) = \left\{g : e_1, \dots, e_n \to e : \mathscr{E} \mid pg = f\right\}$$

  - for $\theta : f \Rightarrow g : \mathrm{B}_1 \dots \mathrm{B}_n \to \mathrm{B}$ in $\mathscr{B}$, $\partial p(\theta)_{b_1, \dots, b_n; b}$ is the distributor

    $$\partial p(\theta)_{b_1, \dots, b_n; b}(\phi, \gamma) = \{\rho : \phi \Rightarrow \gamma, p(\rho) = \theta\}.$$

This defines a lax functor $\partial p : \mathscr{B} \to \mathfrak{Dist}$.

- given a pointwise relational morphism F between Niefield fibrations $p_1 : \mathscr{E}_1 \to \mathscr{B}$ and $p_2 : \mathscr{E}_2 \to \mathscr{B}$,
  - for B an object of $\mathscr{B}$, we set, for $e_1 \in p_1^{-1}(\mathrm{B})$, $e_2 \in p_2^{-1}(\mathrm{B})$, $(e_1, e_2) \in (\partial \mathrm{F})_\mathrm{B}$ if $(e_1, e_2) \in \mathrm{F}$ and, for $f : e_1 \to e_1'$ in $p_1^{-1}(\mathrm{B})$, $g : e_2 \to e_2'$ in $p_2^{-1}(\mathrm{B})$, $(f, g) \in (\partial \mathrm{F})_\mathrm{B}$ if $(f, g) \in \mathrm{F}_{e_2, e_2'}^{e_1, e_1'}$.

    $$(\partial \mathrm{F})_\mathrm{B}(f; g) = \mathrm{F}(f; g)$$

    As F is a relation, $(\partial \mathrm{F})_\mathrm{B}$ can be seen as a distributor $(\partial \mathrm{F})_\mathrm{B} : \partial p_1(\mathrm{B}) \nrightarrow \partial p_2(\mathrm{B})$ which is a relation.
  - for $f : \mathrm{B}_1 \dots \mathrm{B}_n \to \mathrm{B}' : \mathscr{B}$, we set $(\partial \mathrm{F})_f$, for $(b_1, \dots, b_n; b') \in \partial p_1(\mathrm{B}_1) \times \cdots \times \partial p_1(\mathrm{B}_n) \times \partial p_2(\mathrm{B}')$,

    $$((\partial \mathrm{F})_f)_{b_1, \dots, b_n; b'} \subseteq (\partial p_1(f) \circ ((\partial \mathrm{F})_{\mathrm{B}_1}, \dots, (\partial \mathrm{F})_{\mathrm{B}_n}))(b_1, \dots, b_n; b') \times ((\partial \mathrm{F})_{\mathrm{B}'} \circ \partial p_2)(b_1, \dots, b_n; b')$$

    as the diagonal relation.

Hence $\partial \mathrm{F}$ is a relational lax natural transformation between ∂p_1 and ∂p_2. ■

All the ideas behind this theorem are present in the following special case: a Niefield fibration $p : \mathscr{E} \to \mathscr{B}$ such that:

- for each object $b : \mathscr{B}$, $p^{-1}(b)$ is a set;
- for each 2-arrow $\theta : f \to f'$, $p^{-1}(\theta)$ is either empty or a singleton,

that is, in type theoretic terms, the type system p does not have subtyping and a reduction can only be typed by at most one typed reduction, then p can be faithfully represented by:

- for each object in $\mathscr{B}$, its set of pre-images (the refined types)
- for each arrow in $\mathscr{B}$, and refined context, the set of type derivations of the arrow in the context;
- for each reduction, a relation linking the type derivations together.

The cyclic case

5.1.11 In the cyclic case (that is, when $\mathscr{B}$ is a cyclic **Cat**-operad), the theorem has to be adapted: indeed, $\mathfrak{Dist}$ is not a cyclic **Cat**-operad. It can be adapted by just changing $\mathfrak{Dist}$ to the following cyclic operad:

Let $\mathfrak{Dist}_s$ be the following (large) bioperad:

- objects are sets;
- multimorphisms $\mathrm{A}_1 \dots \mathrm{A}_n \nrightarrow \mathrm{B}$ are distributor-valued distributors, that is functors (of bicategories)

  $$\mathrm{F} : \mathrm{A}_1 \times \cdots \times \mathrm{A}_n \times \mathrm{B}^{\mathrm{op}} \to \mathbf{Dist}_s,$$

  where $\mathrm{A}_1, \dots, \mathrm{A}_n, \mathrm{B}$ are trivially viewed as bicategories, and $\mathbf{Dist}_s$ is the bicategory whose objects are semi-categories, arrows distributors between them and 2-arrows natural transformations;
- composition of multimorphisms is defined as the composition of distributors;

- 2-arrows $\theta : \mathrm{F} \Rightarrow \mathrm{G} : \mathrm{A}_1 \dots \mathrm{A}_n \to \mathrm{B}$ are natural transformations of the underlying functors: a family of **Set**-valued functions indexed by $\mathrm{A}_1 \times \cdots \times \mathrm{A}_n \times \mathrm{B}^{\mathrm{op}}$:

  $$\forall (a_1, \dots, a_n, b) \in \mathrm{A}_1 \times \cdots \times \mathrm{A}_n \times \mathrm{B}^{\mathrm{op}}, \theta_{a_1,\dots,a_n,b} : \mathrm{F}(a_1, \dots, a_n, b) \times \mathrm{G}(a_1, \dots, a_n, b) \to \mathbf{Set}$$

  satisfying naturality conditions:

  $$\begin{tikzcd} \mathrm{F}(a_1, \dots, a_n, b) \times \mathrm{G}(a_1, \dots, a_n, b) \arrow[rr, "\mathrm{F}(f_1, \dots, f_n, f) \times \mathrm{G}(f_1, \dots, f_n, f)"] \arrow[dr, "\theta_{a_1,\dots,a_n,b}"] & & \mathrm{F}(a'_1, \dots, a'_n, b') \times \mathrm{G}(a'_1, \dots, a'_n, b \arrow[dl, "\theta_{a'_1,\dots,a'_n,b'}"] \\ & \mathbf{Set} & \end{tikzcd}$$

5.1.12 As subtyping appears in our framework as an arrow $f : \mathrm{A} \to \mathrm{B}$ that types the identity, our framework is able to handle:

- type systems with subtyping for intuitionistic calculi;
- type systems without subtyping for classical calculi,

but no type systems with subtyping for classical calculi, which is a bit puzzling.

5.1.13 We will not consider type systems with subtyping (there exist intersection type systems with subtyping, such as in (Terui 2012)), so the difference will not matter much.

5.2 Theorems of intersection types

5.2.1 Most reduction strategies can not be presented as **Cat**-operads: indeed, reductions in a **Cat**-operad have to compose both sequentially and when applied to subterms, and most strategies are not stable by composition into a subterm. So strategies have to be singled out, not as sub-operads, but as sets of reductions of an operad. As we will see, depending on the kind of properties we are interested in, we will need to define a varying notion of strategy.

5.2.2 We have already defined type systems for linear logic. Indeed, the systems of Chapter 4 define pointwise Niefield fibrations:

$$\begin{aligned} \mathbf{LinPoly}_{\mathbb{T}} &\to \Lambda_{!,\star} \\ \mathbf{RelPoly}_{\mathbb{T}} &\to \Lambda_{!,\star} \\ \mathbf{AffPoly}_{\mathbb{T}} &\to \Lambda_{!,\star} \\ \mathbf{CartPoly}_{\mathbb{T}} &\to \Lambda_{!,\star} \end{aligned}$$

and

$$\begin{aligned} \mathbf{LinPolyLL}_{\mathbb{T}} &\to \mathsf{MELL}_{?,\star} \\ \mathbf{RelPolyLL}_{\mathbb{T}} &\to \mathsf{MELL}_{?,\star} \\ \mathbf{AffPolyLL}_{\mathbb{T}} &\to \mathsf{MELL}_{?,\star} \\ \mathbf{CartPolyLL}_{\mathbb{T}} &\to \mathsf{MELL}_{?,\star}. \end{aligned}$$

Through the Grothendieck construction of the last Section, for each operad **D** of derivations, we have (for instance, in the case of $\mathsf{MELL}_{?,\star}$) an approximation functor $\vec{\mathrm{A}}[\mathbf{D}] : \mathsf{MELL}_{?,\star} \to \mathfrak{Dist}_s$.

So, the general setting in which the work takes place is the following: given a (possibly cyclic) **Cat**-operad $\mathscr{C}$, together with a translation into linear logic $\mathbf{G} : \mathscr{C} \to \mathsf{MELL}_{?,\bigstar}$, and a choice of an operad of derivations, we can define an approximation functor for $\mathscr{C}$:

$$\begin{tikzcd}
\mathscr{C} \arrow[r, "\mathbf{G}"] \arrow[rr, bend right, "{\vec{\mathrm{A}}[\mathbf{D},\mathbf{G}]}"'] & \mathsf{MELL}_{?,\bigstar} \arrow[r, "{\vec{\mathrm{A}}[\mathbf{D}]}"] & \mathfrak{Dist}_s
\end{tikzcd}$$

In turn, by applying the Grothendieck construction to it, we get a type system for $\mathscr{C}$:

$$\begin{tikzcd}
\mathscr{E}\ell(\mathbf{D},\mathbf{G}) \arrow[r] \arrow[d, "{\mathbf{p}[\mathbf{D},\mathbf{G}]}"] & \mathfrak{Dist}_{s,*} \arrow[d] \\
\mathscr{C} \arrow[r, "{\vec{\mathrm{A}}[\mathbf{D},\mathbf{G}]}"] & \mathfrak{Dist}_s
\end{tikzcd}$$

So, the main interest for us of the Grothendieck construction is to allow to compose type system along translations. All the work is therefore to find the right translations and the right type systems for linear logic to capture the properties we are interested in.

Strong-style normalization properties

5.2.3 Suppose we are interested in a calculus presented by a **Cat**-operad $\mathscr{C}$. If we are interested in proving that strong normalization holds for a reduction strategy, we have to prove that every reduction accepted by the strategy enjoys a certain property. In other words, every set of reductions in $\mathscr{C}$ have enough structure to carry a strong normalization proof.

Theorem 6

*Let $\mathscr{C}$ be a **Cat**-operad together with the morphisms*

$$\begin{tikzcd}
\mathscr{C} \arrow[r, "\mathbf{G}"] \arrow[rr, bend right, "{\vec{\mathrm{A}}[\mathbf{D},\mathbf{G}]}"'] & \mathsf{MELL}_{?,\bigstar} \arrow[r, "{\vec{\mathrm{A}}[\mathbf{D}]}"] & \mathfrak{Dist}
\end{tikzcd}$$

Let Φ be a set of 2-arrows in $\mathscr{C}$.
If

- *for all $\phi : t \Rightarrow t' \in \Phi$ and for all list Γ and element B of $\vec{\mathrm{A}}[\mathbf{D}](\mathsf{I})$, $\vec{\mathrm{A}}[\mathbf{D},\mathbf{G}]_{\Gamma,\mathrm{A}}(\phi)$ is entire: for all $\tau \in \vec{\mathrm{A}}[\mathbf{D},\mathbf{G}]_{\Gamma,\mathrm{A}}(t)$, there exists a $\tau' \in \vec{\mathrm{A}}[\mathbf{D},\mathbf{G}]_{\Gamma,\mathrm{A}}(t')$ such that $\vec{\mathrm{A}}[\mathbf{D},\mathbf{G}]_{\Gamma,\mathrm{A}}(\phi)(\tau,\tau')$ is non-empty;*
- *if $\phi \neq \mathrm{id}$, then $\vec{\mathrm{A}}[\mathbf{D},\mathbf{G}]_{\Gamma,\mathrm{A}}(\phi) \neq \mathrm{id}$,*

then, if M is $\mathbf{p}[\mathbf{D},\mathbf{G}]$-typable, then there is no infinite reduction in Φ starting from M.

Proof: The proof is a rewrite of the standard proof in the language of **Cat**-operads: let $\tau \in \vec{A}[\mathbf{D},\mathbf{G}]_{\Gamma,A}(M)$. For any reduction sequence

$$M \Rightarrow M_1 \Rightarrow \cdots \Rightarrow M_n$$

we can define a sequence $\tau \Rightarrow \tau_1 \Rightarrow \cdots \Rightarrow \tau_n$ (as reductions in $\mathbf{D}$ such that $\forall 1 \leqslant i \leqslant n, \tau_i \in \vec{A}[\mathbf{D},\mathbf{G}]_{\Gamma,A}(M_i)$ (by the first condition) that is not stationary (by the second condition). As there is no non-stationary infinite such sequence (by strong normalization of polyadic linear logic), it concludes. ■

Weak-style normalization properties

5.2.4 For weak normalization, we want to ensure that a reduction sequence starting from M ends in a set of terms already chosen. So we have to specify two things: a set of valid reductions and a set of terms already reduced.

Theorem 7

Let $\mathscr{C}$ be a **Cat**-*operad together with the morphisms*

$$\begin{tikzcd} \mathscr{C} \arrow[r, "\mathbf{G}"] \arrow[rr, bend right, "{\vec{A}[\mathbf{D},\mathbf{G}]}"] & \mathsf{MELL}_{?,\star} \arrow[r, "{\vec{A}[\mathbf{D}]}"] & \mathfrak{Dist} \end{tikzcd}$$

Let R *be a set of reductions in* $\mathscr{C}$.
Let $\mathscr{N}$ *be a set of terms in* $\mathscr{C}$.
If

- *for all* ρ *in* R, *and for all list* Γ *and element* B *of* $\vec{A}[\mathbf{D}](\mathsf{I})$, $\vec{A}[\mathbf{D},\mathbf{G}]_{\Gamma,B}(\rho)$ *is surjective: for all* $\tau' \in \vec{A}[\mathbf{D},\mathbf{G}]_{\Gamma,A}(t')$, *there exists a* $\tau \in \vec{A}[\mathbf{D},\mathbf{G}]_{\Gamma,A}(t)$ *such that* $\vec{A}[\mathbf{D},\mathbf{G}]_{\Gamma,A}(\phi)(\tau,\tau')$ *is non-empty,*
- *for all* $n \in \mathscr{N}$, n *is* $\mathbf{p}[\mathbf{D},\mathbf{G}]$-*typable*

then, if there is a succession of reductions in R *from* M *to a term in* $\mathscr{N}$, *then* M *is* $\mathbf{p}[\mathbf{D},\mathbf{G}]$-*typable.*

Proof: This is also just a reformulation of the standard proof: by induction on a reduction sequence from M to a term in $\mathscr{N}$. ■

The fundamental theorem of intersection types

5.2.5 The two above theorems might seem a bit puzzling: we proved that typability implies strong normalization, and that weak normalization implies typability. So the full theorem, a characterization of normalization by typability, requires strong normalization to be equivalent to weak normalization, which is not true for non-deterministic reductions.

The solution to this riddle comes from the fact that the parameter in the last two theorems can be chosen independently in a way that makes them agree. So, this gives the full theorem, dubbed "the fundamental theorem of intersection types".

Theorem 8

Let $\mathscr{C}$ be a **Cat***-operad together with the morphisms*

$$\begin{tikzcd}
\mathscr{C} \arrow[r, "\mathbf{G}"] \arrow[rr, bend right, "{\vec{\mathrm{A}}[\mathbf{D},\mathbf{G}]}"'] & \mathsf{MELL}_{?,\bigstar} \arrow[r, "{\vec{\mathrm{A}}[\mathbf{D}]}"] & \mathfrak{Dist}
\end{tikzcd}$$

Let R *and* Φ *be sets of reductions in $\mathscr{C}$, and let $\mathscr{N}$ be a set of terms in $\mathscr{C}$.*
If

- *for all* $\phi : t \Rightarrow t' \in \Phi$ *and for all list* Γ *and element* B *of* $\vec{\mathrm{A}}[\mathbf{D}](\mathsf{I})$, $\vec{\mathrm{A}}[\mathbf{D},\mathbf{G}]_{\Gamma,\mathrm{A}}(\phi)$ *is entire: for all* $\tau \in \vec{\mathrm{A}}[\mathbf{D},\mathbf{G}]_{\Gamma,\mathrm{A}}(t)$*, there exists a* $\tau' \in \vec{\mathrm{A}}[\mathbf{D},\mathbf{G}]_{\Gamma,\mathrm{A}}(t')$ *such that* $\vec{\mathrm{A}}[\mathbf{D},\mathbf{G}]_{\Gamma,\mathrm{A}}(\phi)(\tau,\tau')$ *is non-empty;*
- *for all* $\phi : t \Rightarrow t' \in \Phi$*, if* $\phi \neq \mathrm{id}$*, then* $\vec{\mathrm{A}}[\mathbf{D},\mathbf{G}]_{\Gamma,\mathrm{A}}(\phi) \neq \mathrm{id}$*;*
- *for all* ρ *in* R*, and for all list* Γ *and element* B *of* $\vec{\mathrm{A}}[\mathbf{D}](\mathsf{I})$, $\vec{\mathrm{A}}[\mathbf{D},\mathbf{G}]_{\Gamma,\mathrm{B}}(\rho)$ *is surjective: for all* $\tau' \in \vec{\mathrm{A}}[\mathbf{D},\mathbf{G}]_{\Gamma,\mathrm{A}}(t')$*, there exists a* $\tau \in \vec{\mathrm{A}}[\mathbf{D},\mathbf{G}]_{\Gamma,\mathrm{A}}(t)$ *such that* $\vec{\mathrm{A}}[\mathbf{D},\mathbf{G}]_{\Gamma,\mathrm{A}}(\phi)(\tau,\tau')$ *is non-empty,*
- *for all* $n \in \mathscr{N}$*,* n *is* $\mathbf{p}[\mathbf{D},\mathbf{G}]$*-typable*
- *for all terms* M*, if there is no infinite reduction in* Φ *starting from* M*, then there is a succession of reductions in* R *from* M *to a term in* $\mathscr{N}$*,*

then for a term M *in* $\mathscr{C}$*, the following conditions are equivalent:*

- M *is* $\mathbf{p}[\mathbf{D},\mathbf{G}]$*-typable,*
- *there is no infinite reduction from* M *in* Φ*,*
- *there is a sequence of reductions from* M *to a term in* $\mathscr{N}$*.*

As we will show, the well-known intersection type systems and their characterization of normalization are instances of this construction and this theorem.

This theorem captures the essence of intersection types as a bridge between a strong normalization property for a certain notion of reduction (represented by the set Φ) and a weak normalization property for an other notion of reduction (represented by the sets R and $\mathscr{N}$). When these sets are harmonious, intersection types characterize exactly normalization for these notions.

5.3 Intersection type systems for the λ-calculus

Recovering Coppo, Dezani and Venneri's original intersection type system

5.3.1 We are now ready to reconstruct Coppo, Dezani-Ciancaglini, and Venneri (1981)'s original intersection type system. Consider the **Cat**-operad $\Lambda_\bigstar$, that we will translate, for simplicity, into $\Lambda_{!,\bigstar}$ through Girard's translation. We will then consider the simply typed derivations of **CartPoly** without empty sequences.

$$\begin{tikzcd}
\mathscr{E}\ell(\mathbf{CartPoly},\mathbf{G}) \arrow[rr] \arrow[d, "\mathbf{p}[\mathbf{CartPoly},\mathbf{G}]"] & & \mathfrak{Dist}_* \arrow[d] \\
\Lambda \arrow[r, "\mathbf{G}"] \arrow[rr, bend right, "\vec{\mathrm{A}}[\mathbf{CartPoly},\mathbf{G}]"'] & \Lambda_{!,\star} \arrow[r, "\vec{\mathrm{A}}[\mathbf{CartPoly}]"] & \mathfrak{Dist}
\end{tikzcd}$$

Objects in $\mathscr{E}\ell(\mathbf{CartPoly},\mathbf{G})$ are the objects in **CartPoly**, so given by the grammar:

$$\mathrm{A} ::= \langle \mathrm{A}_1, \dots, \mathrm{A}_n \rangle | \mathrm{A} \multimap \mathrm{B}$$

and a term M and a reduction $\phi : \mathrm{M} \Rightarrow \mathrm{M}'$ in $\Lambda_\star$ are typed by a simply-typed term in **CartPoly** following the approximation relation with eventually some structural rules applied. So, the rules of the obtained system have shape:

$$\frac{}{x : \mathrm{A} \vdash x : \mathrm{A}}\,(\text{variable})$$

$$\frac{\Gamma, x_1 : \mathrm{A}_1, \dots, x_n : \mathrm{A}_n \vdash t : \mathrm{B}}{\Gamma \vdash \lambda x.t\,\{x_1, \dots, x_n \leftarrow x\} : \langle \mathrm{A}_1, \dots, \mathrm{A}_n \rangle \multimap \mathrm{B}}\,(\text{abstraction})$$

$$\frac{\Gamma \vdash t : \langle \mathrm{A}_1, \dots, \mathrm{A}_n \rangle \multimap \mathrm{B} \qquad \Delta_1 \vdash u : \mathrm{A}_1 \qquad \cdots \qquad \Delta_n \vdash u : \mathrm{A}_n}{\Gamma, \Delta_1, \dots, \Delta_n \vdash tu}\,(\text{application})$$

Figure 5.1: Coppo, Dezani and Venneri's original intersection type system

with all structural rules on the context, which means that the applicative lists behave like sets.

5.3.2 We can moreover use the fundamental theorem of intersection types for this system, by using the following parameters:

- as Φ all reductions in $\Lambda_\star$;
- as R the set of non-erasing reductions;
- as $\mathscr{N}$ all normal forms.

This proves:

Theorem 9 (*Coppo, Dezani and Venneri*)

The system of Figure 5.1 charaterizes strong normalization: a term M *in* $\Lambda_\star$ *is equivalently:*

- *typable in this system;*
- *with no infinite sequences of reduction starting from it;*
- *with a finite sequence of non-erasing reduction starting from it and ending in a normal form.*

Recovering de Carvalho's System R

5.3.3 Carvalho (2009)'s System R is a non-idempotent intersection type system. It is obtained by translating $\Lambda_\star$ into $\Lambda_{!,\star}$ and considering linear derivations **LinPoly**. Its rules are represented Figure 5.2 with only the exchange rule on contexts, which means that the applicative lists behave like multisets.

$$\frac{}{x : \langle \mathrm{A} \rangle \vdash x : \mathrm{A}}\,\text{(variable)}$$

$$\frac{\Gamma, x_1 : \mathrm{A}_1, \dots, x_n : \mathrm{A}_n \vdash t : \mathrm{B}}{\Gamma \vdash \lambda x.t\,\{x_1, \dots, x_n \leftarrow x\} : \langle \mathrm{A}_1, \dots, \mathrm{A}_n \rangle \multimap \mathrm{B}}\,\text{(abstraction)}$$

$$\frac{\Gamma \vdash t : \langle \mathrm{A}_1, \dots, \mathrm{A}_n \rangle \multimap \mathrm{B} \qquad \Delta_1 \vdash u : \mathrm{A}_1 \qquad \cdots \qquad \Delta_n \vdash u : \mathrm{A}_n}{\Gamma, \Delta_1, \dots, \Delta_n \vdash tu}\,\text{(application)}$$

Figure 5.2: de Carvalho's System R

5.3.4 We can also use the fundamental theorem of intersection types for this system, by using the following parameters:

- as Φ the head reductions in $\Lambda_\star$;
- as R the set of all reductions;
- as $\mathscr{N}$ all head-normal forms.

This proves:

Theorem 10 (*de Carvalho*)

The system of Figure 5.2 charaterizes head normalization: a term M *in* $\Lambda_\star$ *is equivalently:*

- *typable in this system;*
- *with no infinite sequences of head reductions starting from it;*
- *with a finite sequence of reduction starting from it and ending in a head-normal form.*

5.4 Intersection type systems for the call-by-value λ-calculus

5.4.1 In the same way, we can consider intersection type systems for the call-by-value λ-calculus, computed thanks to the translation $\mathbf{G}_v$.

$$\begin{tikzcd}
\mathscr{E}\ell(\mathbf{LinPoly},\mathbf{G}_v) \arrow[rr] \arrow[d, "{\mathbf{p}[\mathbf{LinPoly},\mathbf{G}_v]}"] & & \mathfrak{Dist}_* \arrow[d] \\
\Lambda_v \arrow[r, "\mathbf{G}_v"] \arrow[rr, bend right, "{\vec{\mathrm{A}}[\mathbf{LinPoly},\mathbf{G}_v]}"'] & \Lambda_{!,\star} \arrow[r, "{\vec{\mathrm{A}}[\mathbf{LinPoly}]}"] & \mathfrak{Dist}
\end{tikzcd}$$

For instance, computing the system for the approximation derivations **LinPoly**, we get the system shown in Figure 5.3. Its types are given by

$$\mathrm{A},\mathrm{B} ::= \langle \mathrm{A}_1 \multimap \mathrm{B}_1, \ldots, \mathrm{A}_n \multimap \mathrm{B}_n \rangle$$

($n = 0$ is allowed, which gives the base case of the inductive definition). The shape of types is justified by observing that Girard's call-by-value translation is based on the recursive type $\mathrm{D} = !(\mathrm{D} \multimap \mathrm{D})$, and remembering that $\langle - \rangle$ approximates $!(-)$, with only the exchange rule on contexts, which means that the lists behave like multisets.

$$\frac{}{x : \mathrm{A} \vdash x : \mathrm{A}}\,(\text{variable})$$

$$\frac{\Gamma_1, x : \mathrm{A}_1 \vdash \mathrm{N} : \mathrm{B}_1 \qquad \cdots \qquad \Gamma_n, x : \mathrm{A}_n \vdash \mathrm{N} : \mathrm{B}_n}{\Gamma_1 \cdots \Gamma_n \vdash \lambda x.\mathrm{N} : \langle \mathrm{A}_1 \multimap \mathrm{B}_1, \ldots, \mathrm{A}_n \multimap \mathrm{B}_n \rangle}\,(\text{abstraction})$$

$$\frac{\Gamma \vdash \mathrm{M} : \langle \mathrm{A} \multimap \mathrm{B} \rangle \qquad \Delta \vdash \mathrm{N} : \mathrm{A}}{\Gamma \cdot \Delta \vdash \mathrm{MN} : \mathrm{B}}\,(\text{application})$$

Figure 5.3: Non-idempotent intersection types for the call-by-value λ-calculus.

5.4.2 By applying Theorem 8, with the following parameters:

- as Φ and R the set of all reductions in $\Lambda_{v,\star}$;
- as $\mathscr{N}$ all normal forms,

we prove:

Theorem 11

The system of Figure 5.3 charaterizes call-by-value normalization: a term M *in* $\Lambda_{v,\star}$ *is equivalently:*

- *typable in this system;*
- *with no infinite sequences of reductions starting from it;*
- *with a finite sequence of reduction starting from it and ending in a normal form.*

5.5 Intersection type systems for the λμ-calculus

5.5.1 The situation of the λμ-calculus caused us to move from the world of **Cat**-operads to the world of semi-cyclic **Cat**-operads. So, by embedding λμ-calculus into $\mathsf{MELL}_{?,\bigstar}$, we get intersection type systems for the λμ-calculus.

$$\begin{tikzcd}
\mathscr{E}\ell(\mathbf{LinPolyLL},\mathbf{L}) \arrow[rr] \arrow[d, "{\mathbf{p}[\mathbf{LinPolyLL},\mathbf{L}]}"] & & \mathfrak{Dist}_{s,*} \arrow[d] \\
\Lambda M \arrow[r, "\mathbf{L}"] \arrow[rr, bend right, "{\vec{\mathrm{A}}[\mathbf{LinPolyLL},\mathbf{L}]}"'] & \mathsf{MELL}_{?,\bigstar} \arrow[r, "{\vec{\mathrm{A}}[\mathbf{LinPolyLL}]}"] & \mathfrak{Dist}_{s}
\end{tikzcd}$$

The types of the resulting system, ranged over by A, B, are as follows:

$$\begin{aligned} \mathrm{A},\mathrm{B} &::= \langle\!\langle \mathrm{P}_1,\dots,\mathrm{P}_n \rangle\!\rangle, \\ \mathrm{P} &::= \langle \mathrm{A}_1,\dots,\mathrm{A}_n \rangle \multimap \mathrm{B}, \end{aligned}$$

where $\langle\!\langle - \rangle\!\rangle$ is the dual of $\langle - \rangle$, just like the modality $?(-)$ is dual to $!(-)$ in classical linear logic. The shape of the types is immediately justified by noting that Laurent's translation uses the recursive type $\mathrm{D} = ?(!\mathrm{D} \multimap \mathrm{D})$, keeping in mind that $\langle - \rangle$ approximates $!(-)$ and $\langle\!\langle - \rangle\!\rangle$ approximates $?(-)$. In the literature on intersection types for the λμ-calculus, these latter are known as *union types* (Laurent 2004). The typing judgments resulting from the translation are of the form

$$x_1 : \vec{\mathrm{A}}_1,\dots,x_m : \vec{\mathrm{A}}_m, \alpha_1 : \mathrm{B}_1,\dots,\alpha_n : \mathrm{B}_n \vdash \mathrm{M} : \mathrm{C},$$

where by $\vec{\mathrm{A}}$ we denote sequences of the form $\langle \mathrm{A}_1,\dots,\mathrm{A}_n \rangle$, for which we write $\vec{\mathrm{A}}(i)$ to denote A_i. The typing rules are given in Figure 5.4. Interestingly, this system turns out to coincide with a system for the λμ-calculus recently introduced by Kesner and Vial (2017). In their work, the authors go further and characterize strong normalization; we may of course adapt the above to strong normalization, obtaining however a system that is slightly different than Kesner and Vial's.

$\Gamma_{\langle\rangle}$ denotes a context where all variables are typed by $\langle\rangle$ or $\langle\!\langle\rangle\!\rangle$ (as appropriate). In the bottom rule, $\cdot$ denotes the concatenation of sequences (of both forms). The notation $\Gamma \cdot \Delta$ denotes concatenation of lists. Sequences in the context (of both forms) may be permuted at will.

$$\frac{}{\Gamma_{\langle\rangle}, x : \langle A\rangle \vdash x : A}\text{(axiom)}$$

$$\frac{\Gamma, x : \vec{A} \vdash M : B}{\Gamma \vdash \lambda x.M : \langle\!\langle \vec{A} \multimap B \rangle\!\rangle}\text{(abstraction)}$$

$$\frac{\Gamma, \alpha : A, \beta : B \vdash M : C}{\Gamma, \alpha : C \cdot A \vdash \mu\beta.[\alpha]M : B}(\mu)$$

$$\frac{\Gamma \vdash M : \langle\!\langle \overrightarrow{A}_1 \multimap B_1, \dots, \overrightarrow{A}_n \multimap B_n \rangle\!\rangle \qquad (\Delta_i^j \vdash \overrightarrow{A}_i(j))_{1 \leq i \leq n, 1 \leq j \leq k_i}}{\Gamma \cdot \Delta_1^1 \cdots \Delta_1^{k_1} \cdots \Delta_n^1 \cdots \Delta_n^{k_n} \vdash MN : \langle\!\langle B_1, \dots, B_n \rangle\!\rangle}\text{(application)}$$

Figure 5.4: System characterizing λμ head normalizing terms.

6

Approximations in box-connected linear logic

❦

6.1 The case of the λ-calculus

6.1.1 Associating to a term in a calculus whose dynamic can be very complicated (such as the λ-calculus or MELL proof-nets) a set of terms in a simpler calculus (such as a linear or affine calculus) can help study (and define operations on) the first language. For instance, a language such as **LinPoly** is linear and strongly normalizing. To any term in the pure λ-calculus is associated a set of terms of **LinPoly**, the set of its approximations, as elaborated in Chapter 5. So, it can be tempting to define a notion of normal form of any λ-term (even a non-normalizing one) by taking the set of normal forms of its linear approximations.

As we will see, the situation is very different between the intutionnistic world of λ-calculus (and intutionnistic linear logic) and the classical world of proof-nets: the fact that classic calculi allow for multiple conclusion make them lack any tree-like structure.

6.1.2 We will not compare polaydic approximations of the λ-calculus with polyadic approximations of proof-nets: given our definitions, an element of the polyadic Taylor expansion is attached with two projections ($t \leftarrow \mathrm{R}_t \rightarrow \mathrm{R}$), and so contain all the informations on how it was built, and which MELL proof-net it approximates. For the comparison to be interesting, it must be between resource λ-terms and resource proof-nets. Nonetheless, we will use polyadic proof-nets as a tool to understand resource proof-nets. We actually believe that polyadic proof-nets are a syntax for resource proof-nets: non-canonical, but easier to manipulate.

6.1.3 The goal of this Chapter is to introduce and motivate a geometric restriction on MELL proof-nets, *box-connexity* which brings some of the clarity of the λ-calculus' situation to proof-nets.

Coherence

6.1.4 In a language of approximations, not all terms approximate a term in the approximated language. A notion of uniformity (Ehrhard and Regnier 2008) or well-formedness (Kfoury 2000) can capture it, in the case of the λ-calculus.

Definition 46 (*Coherence for resource λ-terms*)

Let t and t' be resource λ-terms. We say that t and t' are **coherent**, written

$$t \frown t',$$

if, by induction on the structure of t,

- $t = t' = x$;
- $t = \lambda x.s$, $t' = \lambda x.s'$ and $s \frown s'$;
- $t = s\mathrm{T}$, $t' = s'\mathrm{T}'$, and $s \frown s'$ and for all $t_1, t_2 \in \mathrm{T} + \mathrm{T}'$, $t_1 \frown t_2$.

A term that is coherent with itself is called *uniform*.

Uniform terms are exactly the terms that approximate λ-terms. Finite cliques (for the coherence relations) are included in the Taylor expansion of a λ-term.

6.1.5 It is impossible to do so for proof-nets: indeed, consider the example, slightly adapted from (Tasson 2009, Example V.9) (in the original example, all the (co-)contractions are replaced by (co-)weakenings), of Figure 6.1. By pairs, the resource proof-nets depicted come from the Taylor expansion of a MELL proof-net, but not the three together.

So, there does not exist a binary notion of coherence, that captures the relationship of a set of polyadic proof-nets and an arbitrary MELL proof-net. This example can be extended to show that there is no n-ary notion of coherence capturing it, for any $n \in \mathbf{N}$. In Section 6.3, we will introduce a notion of coherence for polyadic proof-nets that (translated to resource proof-nets) extends coherence for the resource λ-calculus, and shares its properties but only with respect to box-coherent ones: two resource proof-nets will be coherent if and only if they are in the resource Taylor expansion of the same box-connected MELL proof-net.

Disjointness of reducts

6.1.6 To a term t, we associate its Taylor expansion $\mathscr{T}_t$. There is a reduction on the approximation language, that is moreover strongly terminating. Can we use it to compute a normal form for the term t, that would coincide with the usual notion of normal form for the original calculus?

6.1.7 In the general case, the Taylor expansion is not a set, but the elements of the Taylor expansion are weighted by coefficients in a semi-ring. This allows to capture quantitative behavior. So, the reduction of a formal sum $s = \sum a_i t_i$, weighted by coefficients, is another formal sum $s' = \sum a_i' t_i'$.

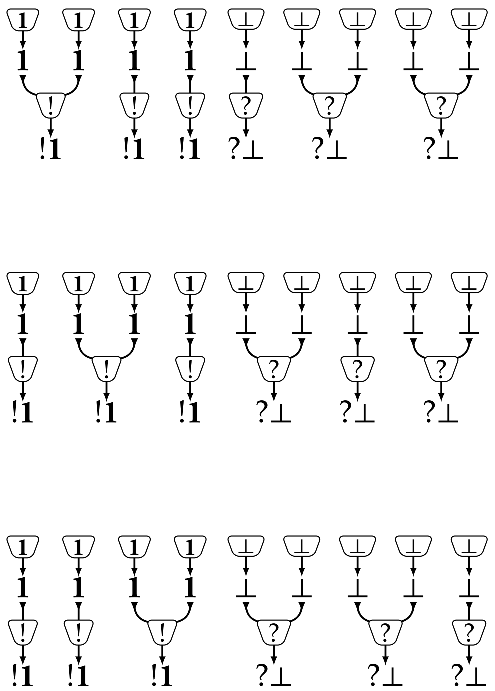


Figure 6.1: Three incoherent resource proof-nets

A priori, some of the coefficients in s' are not finite (as one term in s can reduce to many terms, and the formal sums considered are in general infinite), which means that s' is not well-defined (in absence of a topology on the space of formal sums). So, a simplification of the first question is: are the set of reducts of approximations of the same term disjoint?

6.1.8 It is the case in the λ-calculus (Ehrhard and Regnier 2008, Theorem 22). This result uses coherence and uniformity in a crucial way.

In proof-nets, it is not the case, as shown by the above example.

Example : This counter-example is due to Mazza and Pagani. The proof-net of Figure 6.2 has the two resource proof-nets of Figure 6.3 and Figure 6.3 in its Taylor expansion. Both reduce to the the resource proof-net of Figure 6.5.

6.1.9 In this Chapter, we will not tackle this problem, but the definition of coherence we give can be an important tool.

Computing a term from its Taylor expansion

6.1.10 Given an approximation with enough information, we can compute back the approximated term in the λ-calculus: indeed, any approximation with no empty application has the same tree-like structure of applications and abstraction than the original term, only fattened. It is impossible to do so in proof-nets. Indeed, we can exhibit two MELL proof-nets with an element in the resource Taylor expansion of both. Even worse, we can exhibit two MELL proof-nets whose resource Taylor expansions have an infinite intersection.

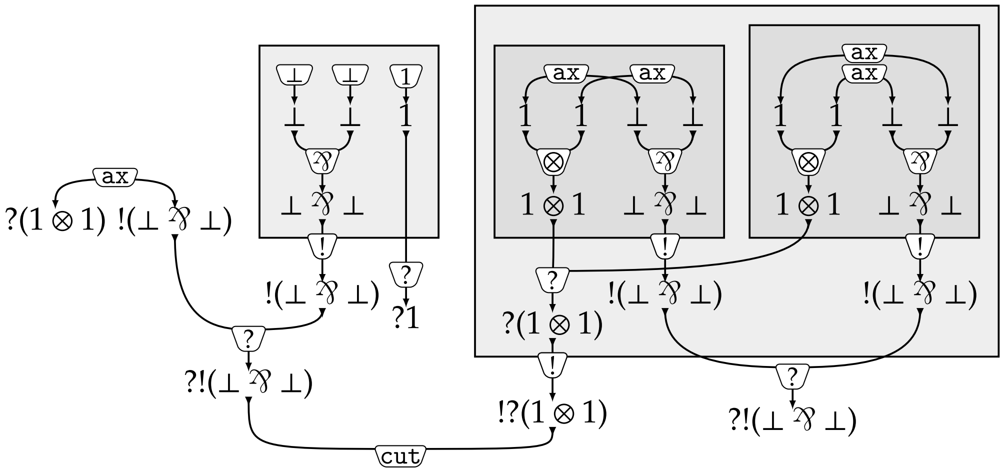
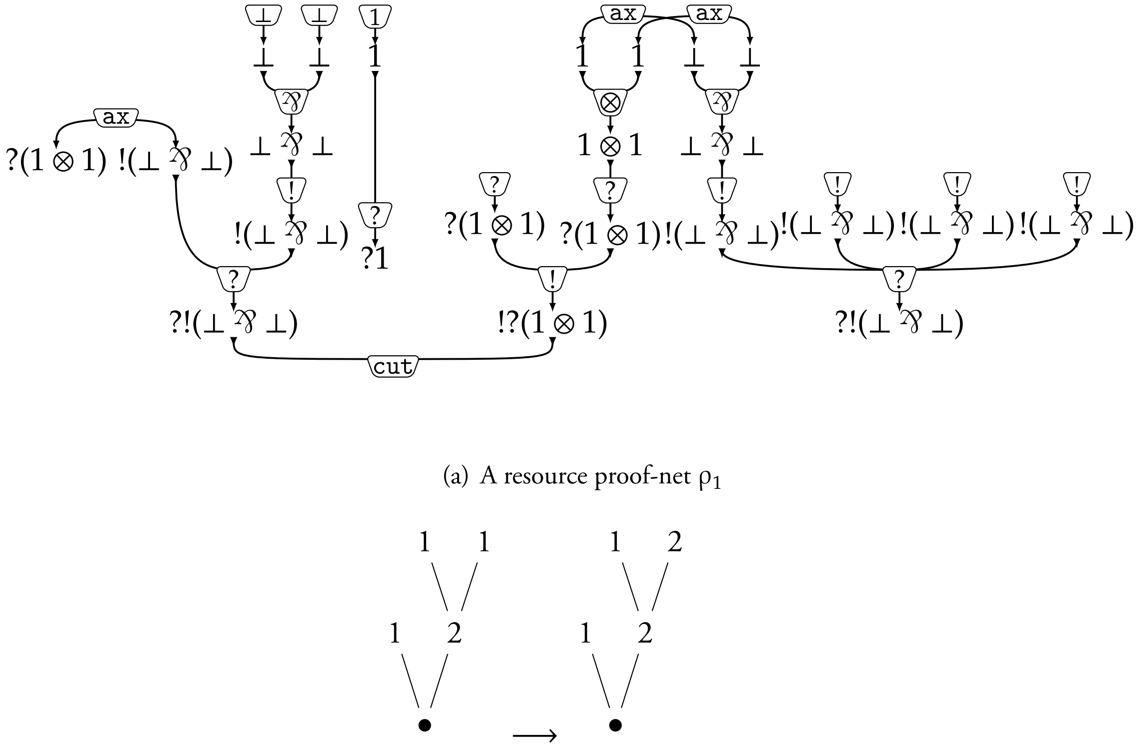


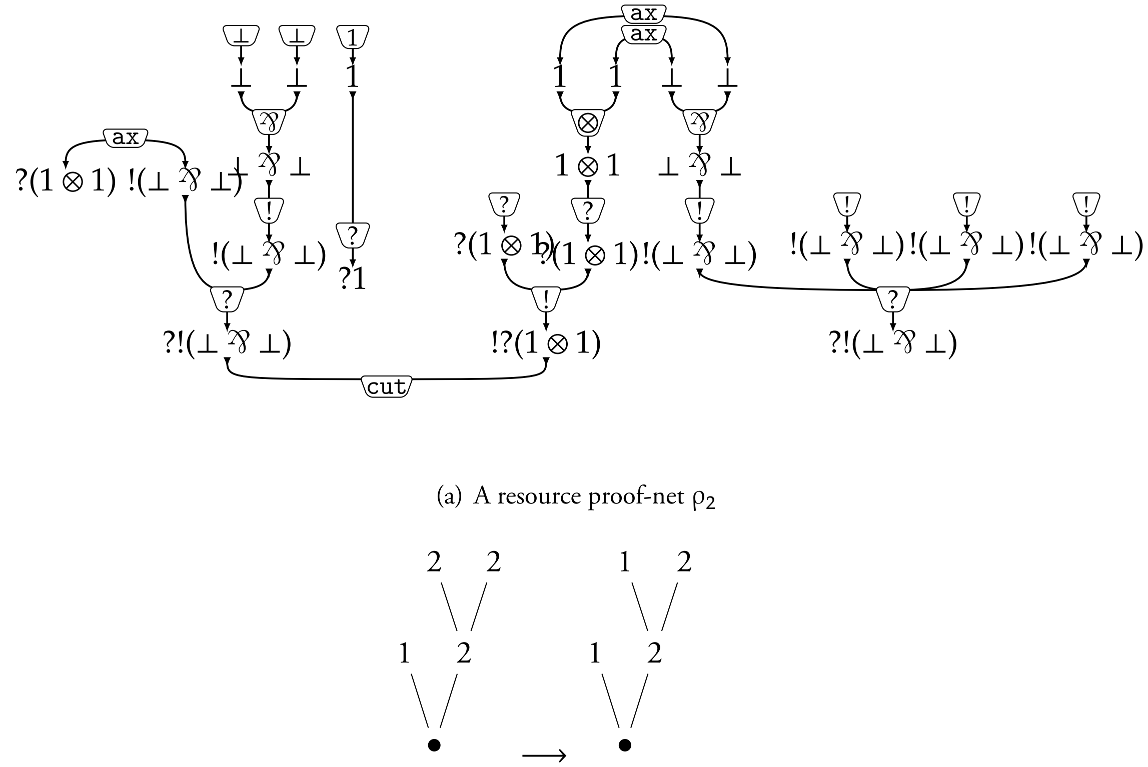
Figure 6.2: A proof-net R



(a) A resource proof-net ρ_1

(b) The thick subtree from which it arose

Figure 6.3: A resource proof-net $\rho_1 \in \mathscr{T}_{\mathrm{R}}$



(a) A resource proof-net ρ_2

(b) The thick subtree from which it arose

Figure 6.4: A resource proof-net $\rho_2 \in \mathscr{T}_R$

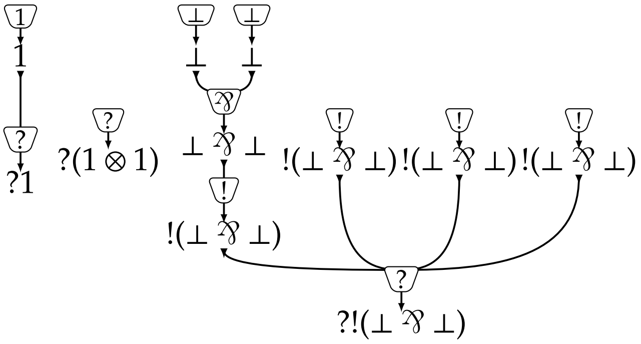


Figure 6.5: A resource proof-net, reduct both of ρ_1 and ρ_2

6.2 Box-connexity

6.2.1 To solve these shortcomings, we introduce a geometric restriction on proof-nets, *box-connexity*, for which the interaction between a proof-net and its approximants is reminiscent of the one between a λ-term and its approximants.

The notion of box-connexity is ill-behaved: it is not stable under reduction (see the example of §6.2.4). Nonetheless, it is essential in our proofs, and any failure of box-connexity leads to the construction of counter-examples to the properties we are interested in. Moreover, the class of box-connected MELL proof-net contain many classes of proof-nets of interest:

- acyclical and connex (in the sense of (Danos and Regnier 1989)) MELL proof nets without $\perp$ cells and weakenings;
- call-by-name translations of λ-terms;
- translations of MELL sequent calculus derivations (without the *mix* rule).

In a sense, box-connexity is an intermediate notion, designed to extend connexity (which is a well-behaved, well-understood notion) to encompass the very tame kind of non-connexity that we find in the λ-calculus. A better notion ought to be defined.

Definition 47 (*?-accessibility*)

A *?-path* on a (polyadic or MELL) proof-net R is a finite sequence $(f_0, \dots, f_n)$ of pairwise distincts flags in R such that, by induction on n:

(i) either $n = 0$ and (f_0) is a ?-path, for all f_0 flag in R;

(ii) or $(f_0, \dots, f_{n-1})$ is a ?-path and $j_\tau(f_{n-1}) = f_n$ (that is, $\{f_{n-1}, f_n\}$ is an edge);

(iii) or $(f_0, \dots, f_{n-1})$ is a ?-path and f_{n-1} is an output of the boundary c of f_n;

(iv) or $(f_0, \dots, f_{n-1})$ is a ?-path and
- f_{n-1} is an input of the boundary c of f_n;
- $\ell(c) \neq ?$.

(v) or $(f_0, \dots, f_{n-1})$ is a ?-path and
- f_{n-1} is an input of the boundary c of f_n;
- $\ell(c) = ?$;
- for all input f of c, there is a ?-path from f_0 to f.

A *complete support* of a ?-path $(f_0, \dots, f_n)$ is the union of $\{f_n\}$ and the complete support of $(f_0, \dots, f_{n-1})$ in the Items i to iv and the union of $\{f_n\}$ and a complete support of a ?-path from f_0 to all inputs of c in the Item v.

A flag f' is **?-accessible** from a flag f if there is a ?-path from f to f'.

Remark 9

Accessibility is preserved by isomorphisms and by the quotients $\simeq_?$ and $\simeq_{!?}$.

6.2.2 The main interest of the notion of complete support is to allow us to do inductions on them, whereas we can not perform inductions on ?-paths.

Lemma 4

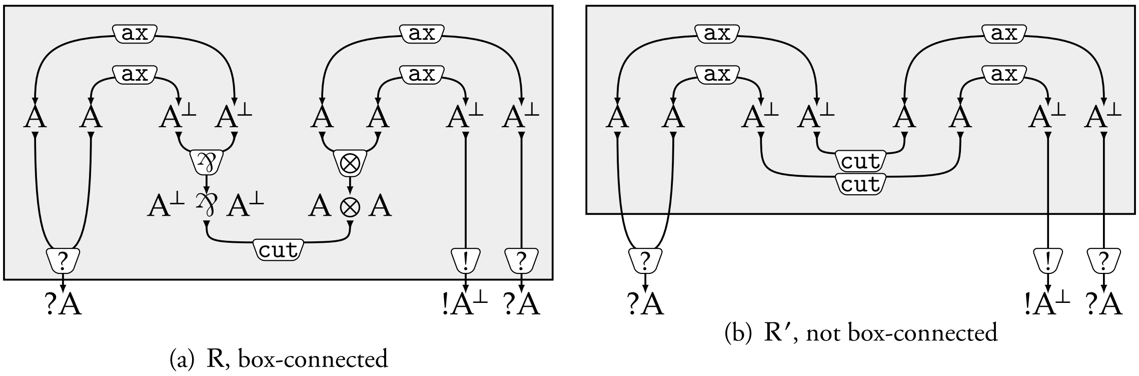
For every ?-path $(f_0, \dots, f_n)$, and every element f in a complete support S *of this path, there is a ?-path from f_0 to f whose complete support is included in* S.

6.2.3 Using the definition of ?-accessibility, we can define box-connexity.

Definition 48 (*box-connexity*)

> Let R be a MELL proof-net.
>
> Let v a non-root vertex of the box-tree $\mathscr{T}$ of R. A ?-path of R is *inside the box* of v if the image of a complete support of the path via box is above v.
>
> R is **box-connected** if, for every non-root vertex v of the box-tree $\mathscr{T}$ of R, and every flag f inside v, there is a path inside the box v from the principal door of v to f.

6.2.4



(a) R, box-connected

(b) R′, not box-connected

Figure 6.6: A box-connected proof-net and its non-box-connected reduct

Example : The proof-net R is box-connected. It reduces to the proof-net R′ which is not.

Box-connexity and fatness

6.2.5 Lemma 3 shows that the underlying graph of a MELL proof-net can be recovered by a polyadic proof-net in its Taylor expansion that does not erase any box. In the case of box-connexity, some elements of the Taylor expansion give even further information.

Definition 49 (*fatness*)

> A polyadic proof-net is n-**fat** if all its !-cells have at least n inputs.

Informally, let us consider a MELL proof-net R and ρ a polyadic proof-net in its Taylor expansion. The different copies of a box of R in ρ are disjoint. If ρ is 2-fat, the ?-accessibles of a premise of a ?-cell can not go outside the copy of the box related to the said premise. If R is box-connected, all the copy of the box is ?-accessible from its premise. So the ?-accessibles allow us to precisely recover the border of the boxes. The remainder of this Section is dedicated to proving the theorem:

Theorem 12

> *Let* R *and* R′ *be two box-connected* MELL *proof-nets, and*

$$\begin{tikzcd}
t & \mathrm{R}_t \arrow[l, "\varphi"'] \arrow[r, "\psi"] & \mathrm{R} \\
t' & \mathrm{R}'_{t'} \arrow[l, "\varphi'"'] \arrow[r, "\psi'"] & \mathrm{R}'
\end{tikzcd}$$

be two elements of their Taylor expansion, such that R_t *and* $\mathrm{R}'_{t'}$ *are 2-fat, and* $f : \mathrm{R}_t \simeq \mathrm{R}'_{t'}$ *as polyadic proof-nets.*

f *induces an isomorphism of* MELL *proof-nets* $\mathrm{R} \simeq \mathrm{R}'$.

In other words, a box-connected MELL proof-net is entirely characterized by any of the 2-fat elements of its Taylor expansion.

6.2.6 The ?-accessibles of a fat enough element of the Taylor expansion of a box-connected proof-net R are related to the ?-accessibles of R. More precisely, ?-accessibility can be transported from a 1-fat element of the Taylor expansion of R back to R.

Lemma 5

Let R *be a box-connected* MELL *proof-net. Let* R_t *be a 1-fat element in its Taylor expansion:*

$$\begin{tikzcd}
t & \mathrm{R}_t \arrow[l, "\varphi"'] \arrow[r, "\psi"] & \mathrm{R}
\end{tikzcd}$$

Let c *be a* !*-cell in* R_t *and* f *one of its premises. Let* f' *be* ?*-accessible from* $\psi(f)$ *inside the box of* $\psi(c)$*. There exists an element in* $\psi^{-1}(f')$ *that is* ?*-accessible from* f*.*

Proof: By induction on the cardinal of a complete support of the ?-path from $\psi(f)$ to f', thanks to Lemma 4, and case disjonction on the last step of the said path.

Let $(\psi(f), f_1, \dots, f')$ be a ?-path whose complete support is inside the box of c.

(i) if $f' = \psi(f)$, then f is the accessible element of the statement;

(ii) if $(\psi(f), \dots, f_{n-1})$ is a ?-path and $j_\tau(f_{n-1}) = f_n$, then, by induction hypothesis, let $g \in \psi^{-1}(f_{n-1})$ be accessible from f, and let f' be the other side of edge of g;

(iii) if $(\psi(f), \dots, f_{n-1})$ is a ?-path and f_{n-1} is an output of the boundary c_n of f_n, then, by induction hypothesis, let $g \in \psi^{-1}(f_{n-1})$ be accessible from f. g is an output of the boundary of a flag $f' \in \psi^{-1}(f_{n-1})$, which is accessible from $\psi(f)$;

(iv) or $(\psi(f), \dots, f_{n-1})$ is a ?-path and

- f_{n-1} is an input of the boundary d of f_n;
- $\ell(d) \neq ?$,

then, by induction hypothesis, let $g \in \psi^{-1}(f_{n-1})$ be accessible from f. g is an input of the boundary of a flag $f' \in \psi^{-1}(f_{n-1})$, which is accessible from $\psi(f)$;

(v) or $(\psi(f), \dots, f_{n-1})$ is a ?-path and

- f_{n-1} is an input of the boundary d of f_n;
- $\ell(d) = ?$;
- for all input h of c, there is a ?-path inside the box of $\psi(c)$ from $\psi(f)$ to h,

then, by induction hypothesis, let $g \in \psi^{-1}(f_{n-1})$ be accessible from f. g is an input of the boundary of a flag $f' \in \psi^{-1}(f_{n-1})$, which is accessible from $\psi(f)$, as all inputs of the boundary of f' are accessible from $\psi(f)$, by induction hypothesis.

■

6.2.7 For a 2-fat element, the ?-accessible are more related.

Lemma 6

> *Let* R *be a box-connected* MELL *proof-net. Let* R_t *be a 2-fat element in its Taylor expansion:*
>
> $$\begin{tikzcd} t & \mathrm{R}_t \arrow[l, "\varphi"'] \arrow[r, "\psi"] & \mathrm{R} \end{tikzcd}$$
>
> *Let* c *be a* ! *cell in* R_t *and* f *one of its premises.*
> *Let* $(f \cdots f_n)$ *be a* ?*-path in* R_t*. Its image via* ϕ *is above* $\phi(f)$*.*

Proof: By induction on a complete support path $(f \cdots f_n)$.

(i) if $f = f_n$, it is obvious;

(ii) if $(f_0, \ldots, f_{n-1})$ is a ?-path and $j_\tau(f_{n-1}) = f_n$, by induction hypothesis, $\phi(f_{n-1})$ is above $\phi(f)$ and so is $\phi(f_n)$, by definition of the boxes;

(iii) if $(f_0, \ldots, f_{n-1})$ is a ?-path and f_{n-1} is an output of the boundary of f_n, by induction hypothesis, $\phi(f_{n-1})$ is above $\phi(f)$, and so is $\phi(f_n)$, as boxes are upwards-closed;

(iv) if $(f_0, \ldots, f_{n-1})$ is a ?-path and
- f_{n-1} is an input of the boundary d of f_n;
- $\ell(d) \neq ?$,

then by induction hypothesis, $\phi(f_{n-1})$ is above $\phi(n)$, and so is $\phi(f_n)$, as boxes are downwards-closed on non ?-cells;

(v) if $(f_0, \ldots, f_{n-1})$ is a ?-path and
- f_{n-1} is an input of the boundary d of f_n;
- $\ell(d) = ?$;
- for all input g of d, there is a ?-path from f_0 to g,

then, by induction hypothesis, all inputs g of the boundary d of f_n are such that $\phi(g)$ if above $\phi(f)$. As R_t is 2-fat, if d was such that the output of d was below $\phi(f)$ (that is, d corresponds to a border of the box in c), one of its input would have an image different than $\phi(f)$. So, $\phi(f_n)$ is above $\phi(f)$. ■

Lemma 7

> *Let* R *be a box-connected* MELL *proof-net. Let* R_t *be a 2-fat element in its Taylor expansion:*
>
> $$\begin{tikzcd} t & \mathrm{R}_t \arrow[l, "\varphi"'] \arrow[r, "\psi"] & \mathrm{R} \end{tikzcd}$$
>
> *Let* c *be a* ! *cell in* R_t *and* f *one of its premises.*
> *Let* $(f \cdots f_n)$ *be a* ?*-path in* R_t*. Its image via* ψ *is a* ?*-path in the box of* $\psi(c)$*.*

Proof: By induction on a complete support of the path $(f \cdots f_n)$:

(i) if $f = f_n$, it is obvious;

(ii) if $(f_0, \ldots, f_{n-1})$ is a ?-path and $j_\tau(f_{n-1}) = f_n$, by induction hypothesis, $(\psi(f_0), \ldots, \psi(f_{n-1}))$ is a ?-path inside the box of $\psi(c)$. In particular, $\psi(f_{n-1})$ is in the box of $\psi(c)$, and so is $\psi(f_n)$, by definition of the boxes;

(iii) if $(f_0, \dots, f_{n-1})$ is a ?-path and f_{n-1} is an output of the boundary of f_n, by induction hypothesis, $\psi(f_{n-1})$ is inside the box of $\psi(c)$, and so is $\psi(f_n)$, as boxes are upwards-closed;

(iv) if $(f_0, \dots, f_{n-1})$ is a ?-path and
- f_{n-1} is an input of the boundary d of f_n;
- $\ell(d) \neq ?$.

then by induction hypothesis, $\psi(f_{n-1})$ is inside the box of $\psi(c)$, and so is $\psi(f_n)$, as boxes are downwards-closed on non ?-cells;

(v) if $(f_0, \dots, f_{n-1})$ is a ?-path and
- f_{n-1} is an input of the boundary d of f_n;
- $\ell(d) = ?$;
- for all input g of d, there is a ?-path from f_0 to g,

by induction hypothesis, and the same argument that in the preceding lemma, $\psi(d)$ is not the border of the box of $\psi(c)$, so, $\psi(f_n)$ is inside the box of $\psi(c)$, which concludes. ■

These Lemmas 5 and 7 combine and give:

Proposition 3

Let R *be a box-connected* MELL *proof-net. Let* R_t *be a 2-fat element in its Taylor expansion:*

$$\begin{tikzcd} t & \mathrm{R}_t \arrow[l, "\varphi"'] \arrow[r, "\psi"] & \mathrm{R} \end{tikzcd}$$

Let c *be a* ! *cell in* R_t *and* f *one of its premises. Let* A_l *be the set of cells* ?*-accessible from* l. $\psi(\mathrm{A}_l \cup \{c\})$ *is the box of* $\psi(c)$.

6.2.8 Moreover, the different copies of the same box can be distinguished in a way that is preserved by isomorphism of resource proof-nets.

Lemma 8 (*copies preservation*)

Let R *and* R′ *be two box-connected* MELL *proof-nets. We note* $\mathsf{box}_{\mathrm{R}} : \mathrm{R} \to \mathscr{T}_{\mathrm{R}}^{\circlearrowleft}$ *and* $\mathsf{box}_{\mathrm{R}'} : \mathrm{R}' \to \mathscr{T}_{\mathrm{R}'}^{\circlearrowleft}$ *their boxing functions. Let*

$$\begin{tikzcd} t & \mathrm{R}_t \arrow[l, "\varphi"'] \arrow[r, "\psi"] & \mathrm{R} \\ t' & \mathrm{R}'_{t'} \arrow[l, "\varphi'"'] \arrow[r, "\psi'"] & \mathrm{R}' \end{tikzcd}$$

be two elements of their Taylor expansion, such that R_t *and* $\mathrm{R}'_{t'}$ *are 2-fat, and* $f : \mathrm{R}_t \simeq \mathrm{R}'_{t'}$ *as polyadic proof-nets.*

Two cells c, c' *of* R_t *have the same image by* φ *and* $\mathsf{box}_{\mathrm{R}} \circ \psi$ *if and only if* $f(c)$ *and* $f(c')$ *have the same image by* φ' *and* $\mathsf{box}_{\mathrm{R}'} \circ \psi'$.

Proof: Let c, c' be two cells of R_t such that

$$\varphi(c) = \varphi(c')$$
$$\mathsf{box}_{\mathrm{R}} \circ \psi(c) = \mathsf{box}_{\mathrm{R}} \circ \psi(c').$$

Let d be a !-cell in R_t. By Proposition 3 and Lemma 6, c is accessible from d if and only if $\psi(c)$ is in the box of $\psi(d)$ and $\varphi(c)$ is above $\varphi(d)$. So, c is accessible from d if and only if c' is acessible from d.

Accessibility is preserved by isomorphism, so $f(c)$ is accessible from $f(d)$ if and only if $f(c')$ is accessible from $f(d)$, or, as all !-cells in $\mathrm{R}'_{t'}$ are of the form $f(d)$

$$\begin{aligned}\varphi'(f(c)) &= \varphi'(f(c'))\\ \mathsf{box}_{\mathrm{R}'} \circ \psi'(f(c)) &= \mathsf{box}_{\mathrm{R}'} \circ \psi'(f(c')),\end{aligned}$$

which concludes, as the other direction is symmetrical. ■

This lemma can be generalized:

Lemma 9 (*box preservation*)

> *Let* R *and* R' *be two box-connected* MELL *proof-nets. We note* $\mathsf{box}_{\mathrm{R}} : \mathrm{R} \to \mathscr{T}_{\mathrm{R}}^{\circlearrowleft}$ *and* $\mathsf{box}_{\mathrm{R}'} : \mathrm{R}' \to \mathscr{T}_{\mathrm{R}'}^{\circlearrowleft}$ *their boxing functions. Let*
>
> $$\begin{tikzcd} t & \mathrm{R}_t \arrow[l, "\varphi"'] \arrow[r, "\psi"] & \mathrm{R} \\ t' & \mathrm{R}'_{t'} \arrow[l, "\varphi'"'] \arrow[r, "\psi'"] & \mathrm{R}' \end{tikzcd}$$
>
> *be two elements of their Taylor expansion, such that* R_t *and* $\mathrm{R}'_{t'}$ *are 2-fat, and* $f : \mathrm{R}_t \simeq \mathrm{R}'_{t'}$ *as polyadic proof-nets.*
>
> *Two cells* c, c' *of* R_t *have the same image by* $\mathsf{box}_{\mathrm{R}} \circ \psi$ *if and only if* $f(c)$ *and* $f(c')$ *have the same image by* $\mathsf{box}_{\mathrm{R}'} \circ \psi'$.

Proof: Let c, c' be two cells of R_t such that

$$\mathsf{box}_{\mathrm{R}} \circ \psi(c) = \mathsf{box}_{\mathrm{R}} \circ \psi(c').$$

Suppose $\varphi(c) \neq \varphi(c')$ (or else, conclude with the last Lemma).

Let g be the flag in R_t whose boundary is sent, via ψ, to the principal door of the box $\mathsf{box}_{\mathrm{R}} \circ \psi(c)$ and from which c is accessible.

Let g' be the flag in R_t whose boundary is sent, via ψ, to the principal door of the box $\mathsf{box}_{\mathrm{R}} \circ \psi(c)$ and from which c' is accessible.

$f(g)$ and $f(g')$ have the same image via ψ': they are inputs of a !-cell, whose image via ψ' only have one output. By Lemma 8, $f(g)$ and $f(c)$ have the same image via $\mathsf{box}_{\mathrm{R}} \circ \psi$, and $f(g')$ and $f(c')$ too. So

$$\mathsf{box}_{\mathrm{R}'} \circ \psi'(f(c)) = \mathsf{box}_{\mathrm{R}'} \circ \psi(f(c')).$$

The other direction is symmetrical. ■

Finally,

Lemma 10

> *Let* R *and* R' *be two box-connected* MELL *proof-nets. We note* $\mathsf{box}_{\mathrm{R}} : \mathrm{R} \to \mathscr{T}_{\mathrm{R}}^{\circlearrowleft}$ *and* $\mathsf{box}_{\mathrm{R}'} : \mathrm{R}' \to$

$\mathscr{T}_{\mathrm{R}'}^{\circlearrowleft}$ *their boxing functions. Let*

$$\begin{tikzcd}
t & \mathrm{R}_t \arrow[l, "\varphi"'] \arrow[r, "\psi"] & \mathrm{R} \\
t' & \mathrm{R}'_{t'} \arrow[l, "\varphi'"'] \arrow[r, "\psi'"] & \mathrm{R}'
\end{tikzcd}$$

be two elements of their Taylor expansion, such that R_t *and* $\mathrm{R}'_{t'}$ *are 2-fat, and* $f : \mathrm{R}_t \simeq \mathrm{R}'_{t'}$ *as polyadic proof-nets.*

Let c *be any cell in* R_t. $\mathsf{box}_{\mathrm{R}} \circ \psi(c)$ *and* $\mathsf{box}_{\mathrm{R}'} \circ \psi'(f(c))$ *have the same number of pre-images in their respective thick subtrees* t *and* t'*.*

Proof: By induction on the depth of $\psi(c)$. If $\psi(c)$ is of depth 0, then c is not ?-accessible from any !-cell. $\mathsf{box}_{\mathrm{R}} \circ \psi(c)$ is the root of $\mathscr{T}_{\mathrm{R}}$, and as $f(c)$ is not accessible from any !-cell too, $\mathsf{box}_{\mathrm{R}'} \circ \psi'(c)$ is the root of $\mathscr{T}_{\mathrm{R}'}$. There is only one pre-image of a root in a thick subtree, which concludes.

Else, let d be be the !-cell from which c is ?-accessible and such that $\psi(c)$ is immediately in the box of $\psi(d)$. $\mathsf{box}_{\mathrm{R}} \circ \psi(d)$ have a certain number n of pre-images by in the thick subtree t. By induction hypothesis, $\mathsf{box}_{\mathrm{R}'} \circ \psi'(f(d))$ have the same number n of pre-images in the thick subtree t'. The number of pre-images of $\mathsf{box}_{\mathrm{R}} \circ \psi(c)$ is the sum, for each of the pre-images of $\mathsf{box}_{\mathrm{R}} \circ \psi(d)$, of their input arities, which are preserved by the isomorphism. ■

6.2.9 Let R be a box-connected MELL proof-net and

$$\begin{tikzcd}
t & \mathrm{R}_t \arrow[l, "\varphi"'] \arrow[r, "\psi"] & \mathrm{R}
\end{tikzcd}$$

be a 1-fat element of its Taylor expansion. As ψ is surjective, there exists sections of ψ (functions $\sigma : \mathrm{R} \to \mathrm{R}_t$ such that $\psi \circ \sigma = \mathrm{id}_{\mathrm{R}}$). Any such section σ can be specified by its composite $\varphi \circ \sigma$. Among all the sections, one is such that $\varphi \circ \sigma$ is the left-most branch of the tree t. We call this section *canonical.* We are now ready to give the main proof of this Section:

Proof of Theorem 12: ψ is surjective. Let σ be the canonical section of ψ. Let us call $\chi = \psi' \circ f \circ \sigma$. χ is well-defined, we just have to show that it is an isomorphism.

$$\begin{tikzcd}
\mathrm{R}_t \arrow[d, "f"] & \mathrm{R} \arrow[l, "\sigma"] \arrow[d, "\chi"] \\
\mathrm{R}'_{t'} \arrow[r, "\psi'"] & \mathrm{R}'
\end{tikzcd}$$

injectivity Let p, p' cells in R. If $\mathsf{box}(p) \neq \mathsf{box}(p')$, without loss of generality, suppose p is in the box of a !-cell c but not p'. $\sigma(p)$ is accessible from the left-most premise of $\sigma(c)$ but not $\sigma(p')$. As f is an isomorphism, $f(\sigma(p))$ is accessible from a premise of the !-cell $f(c)$, and $f(\sigma(p'))$ is not. So $\chi(p)$ is in a box in which $\chi(p')$ is not, and in particular, $\chi(p) \neq \chi(p')$.

Else, $\mathsf{box}(p) = \mathsf{box}(p')$. Remark that, as σ is a section, it is in particular injective, so $f(\sigma(p)) \neq f(\sigma(p'))$. As $\mathrm{R}'_{t'}$ is defined as a pullback, two different cells in $\mathrm{R}'_{t'}$ have different images either by φ' or by ψ'.

As σ is canonical, $\varphi(\sigma(p)) = \varphi(\sigma(p'))$ and so, by Lemma 8, $\varphi'(f(\sigma(p))) = \varphi'(f(\sigma(p')))$, which implies that

$$\psi'(f(\sigma(p)) \neq \psi'(f(\sigma(p')).$$

χ is injective.

surjectivity Let q be a cell in R'. $(\psi')^{-1}(q)$ is a set of cells of $\mathrm{R}'_{t'}$ of arity $n \geqslant 1$ (as $\mathrm{R}'_{t'}$ is 2-fat, so, in particular 1-fat), and all elements of $(\psi')^{-1}(q)$ have the same image via $\mathsf{box}_{\mathrm{R}'} \circ \psi'$. As f is an isomorphism, $f^{-1}((\psi')^{-1}(q))$ is also a set of arity n whose elements have the same image via $\mathsf{box}_{\mathrm{R}} \circ \psi$, by Lemma 9.

As the elements of $(\psi')^{-1}(q)$ have pairwise disjoint images via φ', by the contrapositive of Lemma 8, it is also the case that $f^{-1}((\psi')^{-1}(q))$ have n pairwise disjoint images via φ.

$$\begin{tikzcd}
f^{-1}((\psi')^{-1}(q)) \arrow[r, "\varphi"] \arrow[dr, "\mathsf{box}_{\mathrm{R}} \circ \psi"'] & t^{\circlearrowleft} \arrow[d] \\
 & \mathscr{T}_{\mathrm{R}}^{\circlearrowleft}
\end{tikzcd}$$

By Lemma 10, $\mathsf{box}_{\mathrm{R}} \circ \psi(f^{-1}((\psi')^{-1}(q)))$ have n pre-images in its thick subtree t, so, all the pre-images are in the image of $f^{-1}((\psi')^{-1}(q))$ via φ.

We have shown that there is an element of $f((\psi')^{-1}(q))$ in the image of σ, which means that q is in the image of χ.

morphism of MELL **proof-nets** χ is a morphism of labelled, colored, oriented graphs. Moreover the box-preservation conditions, applied to a cell and any of the inputs of a box, imply that χ is an isomorphism of MELL proof-nets. ■

Remark 10

In the above proof, the canonical section plays no particular role. Any section would define the same isomorphism.

6.3 Glueability of fat polyadic proof-nets

6.3.1 In the polyadic λ-calculus' case, empty approximants represent a loss of information: there is no way to guess which term $\langle\rangle$ is an approximation of. In the polyadic proof-net's case, the situation is far worse: a box approximated by the empty proof-net not only can not be recovered, but cause a loss of information on the structure of the term: indeed, all our method is predicated on finding the border of the boxes thanks to the ?-accessibles, which require cells from the box to be present.

We are going to present a rewriting system for sets of fat polyadic proof-nets, whose main property will be that a successful rewriting allows to reconstruct a box-connected MELL proof-net. In a second time, we will extend it to non-fat polyadic proof-nets.

Polyadic proof-nets with daimon

6.3.2 Let us consider the set of polyadic proof-nets. For technical reasons, we need to extend the cells of the polyadic proof-nets with a new family of cells, the *daimon*, with no inputs and an arbitrary number of outputs. The daimon is used to represent a part of a net from which no informations can be gathered, apart from the number and types of its outputs.

Definition 50

A *polyadic module with daimon* and a **polyadic proof-net with daimon** is defined as in Definition 27 by adding a new family of cells, the *daimon* $\maltese_p$ with p outputs.

Formally, ℓ can take the value $\maltese_p$ and we add a case to the condition on the cells:

- if $\ell(v) = \maltese_p$, the corolla τ_v has no inputs and p outputs $o_1, \dots, o_p$ and no conditions on the types.

For the remainder of this chapter, a polyadic proof-net is implicitly intended as polyadic proof-net with daimon. When needed, we will refer to a proof-net as in Definition 27 as a *daimonless proof-net.*

6.3.3 We say that two polyadic proof-nets *have the same interface* if there is an increasing bijection between their conclusions that preserves the types.

We will consider the conclusion of a set of polyadic proof-nets that have the same interface to be numbered, that is, we will name the conclusions by the only increasing function from a finite initial segment of $\mathbf{N}$ to the set of conclusions. When adding or removing a conclusion, we will do the renumbering on the fly: if we add a conclusion i to a proof-net, all the conclusions grater than i are incremented by 1.

Definition 51

We define the set $\mathbf{PolyPN}^*_{\maltese}$ as the set whose elements are finite lists of polyadic proof-nets (with daimon) with the same interface.

We will note by $\varepsilon_n \in \mathbf{PolyPN}^*_{\maltese}$ the list of n empty polyadic proof-nets, or if the length of the list does not matter, ε.

Glueability of fat polyadic proof-nets

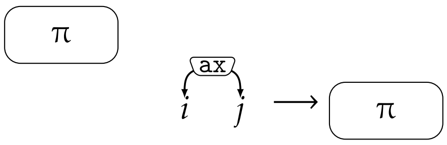
6.3.4 As polyadic proof-nets have no inductive structure, we can not define coherence inductively, contrarily to the λ-calculus' case. Instead, we will define a rewriting system on arbitrary number of polyadic proof-nets that do the same rewriting on all of them, stripping them from their cells, in a way compatible to their putative structure of approximants of the same MELL proof-net.

Definition 52 (*rewriting system on* $\mathrm{PolyPN}^*_{\maltese}$)

We define the following rewriting rules that rewrites a resource proof-net ρ into a (possibly empty) list of resource proof-nets:

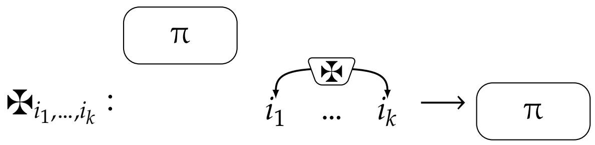
axiom If there are two conclusions $i < j$, such that ρ is the disjoint union of a polyadic proof-net π and an axiom in i and j,

$\mathtt{ax}_{i,j}$:



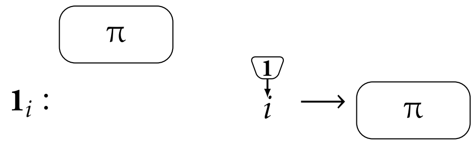
ρ reduces to π.

daimon If there are conclusions $i_1 < \cdots < i_k$, such that ρ is the disjoint union of a polyadic proof-net π and a daimon in $i_1, \dots, i_k$



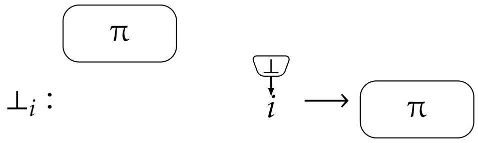
ρ reduces to π.

multiplicatives If there is a conclusion i, such that ρ is the disjoint union of a polyadic proof-net π and a **1** in i,



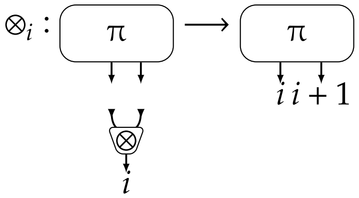
ρ reduces to π.

If there is a conclusion i such that ρ is the disjoint union of a polyadic proof-net π and a ⊥ in i,



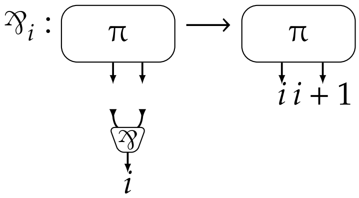
ρ reduces to π.

If there is a conclusion i such that ρ is a ⊗ on i connected to a proof-net π,



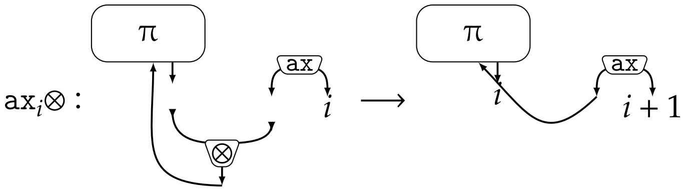
ρ reduces to π with the two conclusions changed.

If there is a conclusion i such that ρ is a ⅋ on i connected to a proof-net π,



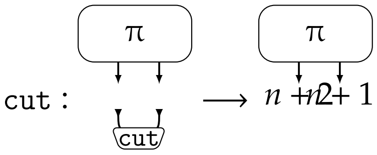
ρ reduces to π with the two conclusions changed.

multiplicatives/axiom If there is a conclusion i such that ρ is a $\mathtt{ax}$ whose second conclusion is the right input of a ⊗ cell, connected to a proof-net π,



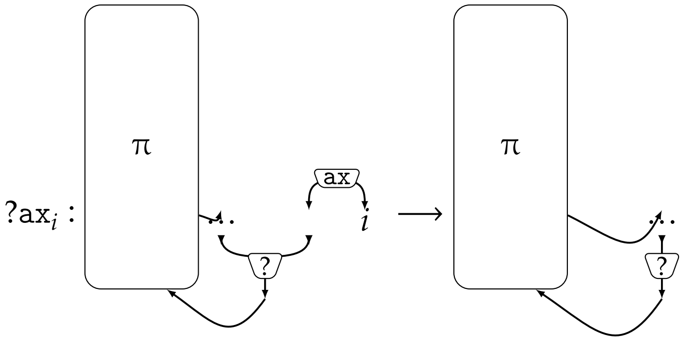
In the same way, there are rules $\mathtt{ax}_i⅋$, $⅋\mathtt{ax}_i$, $⊗\mathtt{ax}_i$ for similar situations involving axioms connected to ⊗-cells and ⅋-cells.

cut If there is a cut such that ρ is a $\mathtt{cut}$ whose two premises are connected to a proof-net π,



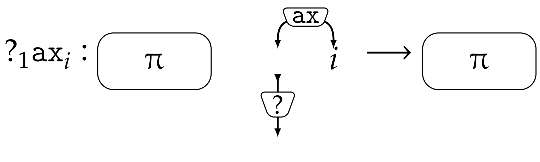
it reduces to π with two new conclusions with fresh names.

?/axiom If there is a conclusion i such that ρ is a $\mathtt{ax}$ whose second conclusion is a premise of a ? cell with other premises,



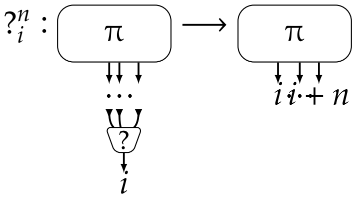
ρ reduces to π connected to the ?-cell with the conclusions changed.

If there is a conclusion i such that ρ is a $\mathtt{ax}$ whose second conclusion is a premise of a ? with no other premises,



ρ reduces to π.

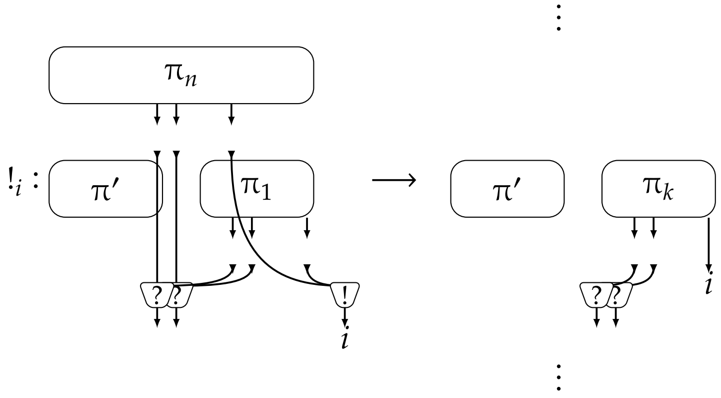
contraction If there is a conclusion i such that ρ is a ? with n inputs connected to a proof-net π,



then ρ reduces to π.

?-split If there exists a ?-conclusion i and a !-conclusion j, we split i in ρ in two !, one whose all premises are accessible from a premise of j, the other with no premise accessible from a premise of j. One of the !-cells might be of empty inputs.

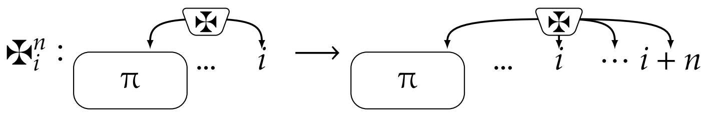
!-split If there exists a !-conclusion i such that all the conclusions of its connected component are ? such that every premise is accessible from exactly one premise of i, then the connected component of i is is the disjoint union of a family $(\pi_k)_{1\leqslant k\leqslant n}$ whose conclusions are all premises of the same ? conclusions, or of the ! conclusion i.



the proof-net ρ reduces to the k proof-nets made from the disjoint union of π' and one π_k.

Let I be any initial subset of **N** and $\rho = (\rho_\ell)_{\ell\in\mathrm{I}}$ be an element of **PolyPN**$^*_{\maltese}$. The rewriting rules aforewritten extend to a rewriting system on **PolyPN**$^*_{\maltese}$ in the following way: the same rule or, not for all $\ell \in \mathrm{I}$,

daimon-split If there is a conclusions i, such that ρ is a daimon on i connected to a polyadic proof-net π,



ρ reduces to π, with a daimon where the conclusion i has been rplaced with n conclusions. has to be used on all of the $(\rho_\ell)_{\ell\in\mathrm{I}}$ (in particular in the rule contraction, which depends on the arity) so that the interfaces of the obtained list remain the same.

6.3.5 The rules we just describe allow to unwind the structure of a list $(\rho_\ell)_{\ell\in I} \in \mathbf{PolyPN}^{*}_{\maltese}$ step-by-step. Glueability is defined as the possibility to be fully unwind.

Definition 53 (*Glueability*)

> We say that n fat polyadic proof-nets $r_1, \dots, r_n$ are **glueable**, if they have the same interface and there exists a rewriting $(r_1, \dots, r_n) \to \varepsilon$ in $\mathbf{PolyPN}^{*}_{\maltese}$.

6.3.6 The relation of glueability is commutative.

Lemma 11

> *Two polyadic proof-nets with no* ! *cells are glueable if and only if they are isomorphic.*

Proof : Two isomorphic proof-nets are glueable. Reciprocally, first remark that the path witnessing their glueability does not contain !-split or ?-split. So, the proof-nets along a path witnessing their glueability are of a decreasing number of cells. So, the proof can be done by induction on the number of cells, and by case disjunction on the first rule of the path witnessing the glueability. ■

Lemma 12

> *Let* (r_1, r_1') *and* (r_2, r_2') *be objects in* $\mathbf{PolyPN}^{*}_{\maltese}$.
> $r_1 \sqcup r_2$ *is glueable with* $r_1' \sqcup r_2'$ *if and only if* r_1 *is glueable with* r_1' *and* r_2 *glueable with* r_2'.

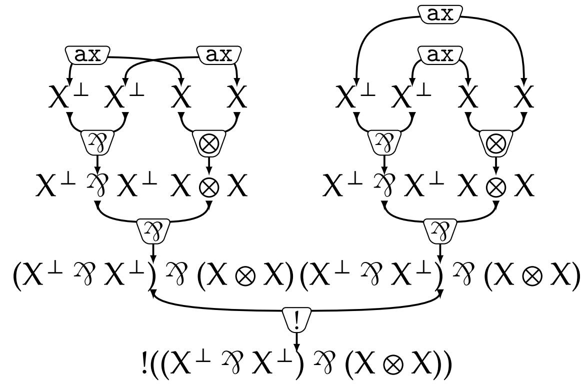


Figure 6.7: A non-uniform polyadic proof-net

Example : Let ϕ be the cocontraction of !**tt** and !**ff**. ϕ is depicted Figure 6.7. ϕ is not uniform, that is not glueable with itself.

Indeed, the left and right co-contracted proof-nets are not glueable, as they are multiplicative and not isomorphic. The only reduction applicable to (ϕ, ϕ) is !-split, which reduces to (**tt** $\sqcup$ **tt** $\sqcup$ **ff** $\sqcup$ **ff**, **tt** $\sqcup$ **ff** $\sqcup$ **tt** $\sqcup$ **ff**), which is made of two unglueable polyadic proof-nets.

6.3.7

Example : The three resource proof-nets of Figure 6.1 are not glueable: indeed, as the ? conclusions are disconnected from the rest of the proof-nets (in all three proof-nets), the only rule that could be applicable to them is **contraction**, but it is not, as the arity changes in the different proof-nets.

Unwinding of box-connected MELL *proof-nets*

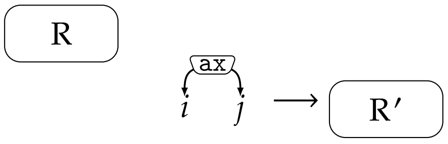
6.3.8 We define a rewriting system for box-connected MELL proof-net. In a sense, this rewriting system reduces a box-connected proof-net to the empty proof-net, exhibiting a sequential structure in the construction of the proof-net.

Definition 54 (*rewriting system for box-connected proof-nets*)

Let R be a box-connected MELL proof-net.

axiom If there are two conclusions $i < j$, such that R is the disjoint union of a proof-net R and an axiom in i and j,

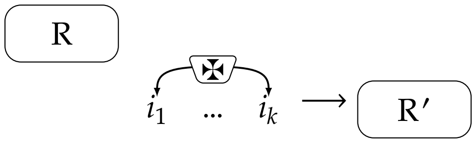
$\mathtt{ax}_{i,j}$:



R reduces to R′, a box-connected proof-net.

daimon If there are conclusions $i_1 < \cdots < i_k$, such that R is the disjoint union of a proof-net R′ and a daimon in $i_1, \ldots, i_k$

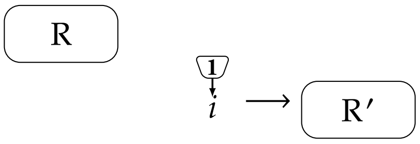
$\maltese_{i_1,\ldots,i_k}$:



R reduces to R′.

multiplicatives If there is a conclusion i, such that R is the disjoint union of a proof-net R′ and a **1** in i,

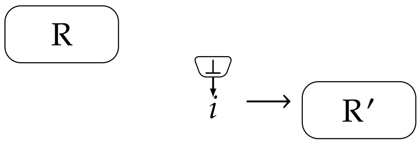
$\mathbf{1}_i$:



R reduces to R′.

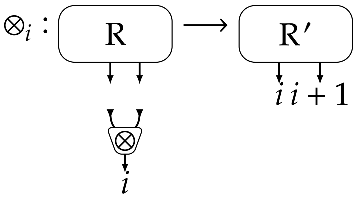
If there is a conclusion i, such that R is the disjoint union of a proof-net R′ and a $\perp$ in i,

$\perp_i$:



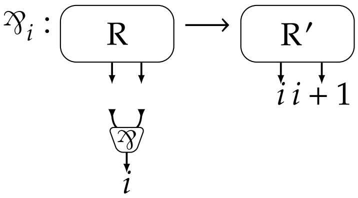
R reduces to R′.

If there is a conclusion i such that R is a ⊗ on i connected to a proof-net R′,



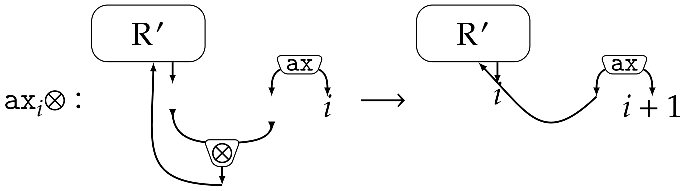
R reduces to R′ with the two conclusions changed.

If there is a conclusion i such that R is a ⅋ on i connected to a proof-net R′,



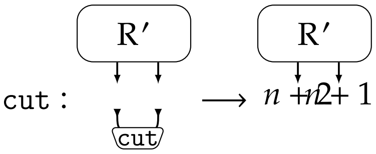
R reduces to R′ with the two conclusions changed.

multiplicatives/axiom If there is a conclusion i such that R is a ax whose second conclusion is the right input of a ⊗ cell, connected to a proof-net R′,



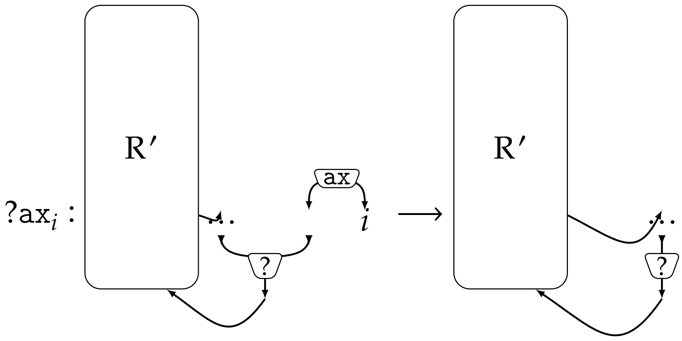
In the same way, there are rules ax$_i$⅋, ⅋ax$_i$, ⊗ax$_i$ for similar situations involving axioms connected to ⊗-cells and ⅋-cells.

cut If there is a cut such that R is a cut whose two premises are connected to a proof-net R′,



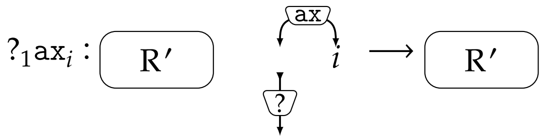
it reduces to R′ with two new conclusions with fresh names.

?/axiom If there is a conclusion i such that R is a ax whose second conclusion is a premise of a ? cell with other premises,



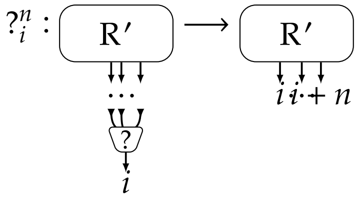
ρ reduces to R′ connected to the ?-cell with the conclusions changed.

If there is a conclusion i such that R is a ax whose second conclusion is a premise of a ? with no other premises,



R reduces to R′.

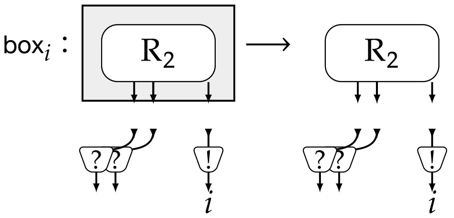
contraction If there is a conclusion i such that R is a ? with n inputs connected to a proof-net R′,



then R reduces to R′.

?-split If there exists a ?-conclusion i and a !-conclusion j, we split i in ρ in two !, one whose all premises are accessible from a premise of j, the other with no premise accessible from a premise of j. One of the !-cells might be of empty inputs.

box If there exists a !-conclusion i such that all the conclusions of its connected component are ?-cells,



6.3.9 The reduction systems we defined on polyadic proof-nets and on MELL proof-nets are actually connected: a reduction in polyadic proof-nets can be backwards simulated in MELL proof-nets.

In the remainder of this Section, we will say that $\rho \in \mathbf{PolyPN}^*_{\maltese}$ *projects* to a MELL proof-net R, which we write

$$\rho \mapsto \mathrm{R}$$

if, all polyadic proof-nets ρ_ℓ are endowed with the two projections to R and a thick subtree of its boxing tree of R making them elements of $\mathscr{T}_{\mathrm{R}}$.

Proposition 4 (*The Taylor projection is a backward simulation*)

Let R *be a box-connected proof-net and* $(\rho_i)_{1\leqslant i\leqslant n}$ *be elements of its polyadic Taylor expansion. This family defines an element of* $\mathbf{PolyPN}^*_{\maltese}$ *together with a projection* $\rho \mapsto \mathrm{R}$.

For all rewritings $(\rho'_i)_{1\leqslant i\leqslant m} \to (\rho_i)_{1\leqslant i\leqslant n}$,

$$\begin{tikzcd}
(\rho'_i)_{1\leqslant i\leqslant m} \arrow[r] & (\rho_i)_{1\leqslant i\leqslant n} \arrow[d, mapsto] \\
 & \mathrm{R}
\end{tikzcd}$$

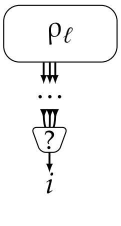
there exists a reduction $\mathrm{R}' \to \mathrm{R}$ *such that there is a projection* $\rho' \mapsto \mathrm{R}'$.

$$\begin{tikzcd}
(\rho'_i)_{1\leqslant i\leqslant m} \arrow[r] \arrow[d, mapsto, dashed] & (\rho_i)_{1\leqslant i\leqslant n} \arrow[d, mapsto] \\
\mathrm{R}' \arrow[r, dashed] & \mathrm{R}
\end{tikzcd}$$

Proof: If the reduction is an axiom, daimon, a multiplicative reduction, involves an axiom, contraction or ?-**split**, it is straightforward, as the rewriting on the polyadic proof-nets are matched with a rewriting that does not change the boxing function of the MELL proof-net.

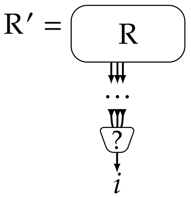
Consider, as an example, the rule **contraction**. Let $(\rho'_\ell)_{\ell\in\mathrm{I}'}$ be a family of polyadic proof-net such that ρ is the image of ρ' through $?^c_i$. It means that the ρ'_ℓ are of the shape of ρ_{ell} grafted with a ?-cell on the conclusions $i, \dots, i+k$:

$\rho'_\ell =$



with the right renumbering.

As conclusions $i, \dots, i + k$ of R are conclusions, they are not in any box. As such, they can be grafted to a contraction outside of any box. So, all the ρ'_ℓ are in the Taylor expansion of



For the rule !-**split**, the connected component of the !-conclusion i is to be boxed: the boxing tree of R' is obtained from the boxing tree of R by adding a new son to the root, which is the image via the boxing function of the connected component of i, and whose sons are selected among the other sons of the root such that the boxing function respects the conditions of a boxing function. ■

6.3.10 Moreover, the reduction system on MELL proof-nets can help us build reduction sequences for polyadic proof-nets: indeed, given a sequence of polyadic proof-nets that project to a MELL proof-net, we can build a reduction step in both.

Proposition 5

Let R *be a box-connected proof-net and* $(\rho_i)_{1\leqslant i\leqslant n}$ *of* $\mathbf{PolyPN}^{*}_{\maltese}$ *together with a projection* $\rho \mapsto \mathrm{R}$. *There exists a reduction* $\rho \to \rho'$ *and a reduction* $\mathrm{R} \to \mathrm{R}'$ *such that* $\rho' \mapsto \mathrm{R}'$ *and* R' *is strictly smaller than* R *in the lexicographic order* $(\mathfrak{m}_\mathrm{R}, |\mathrm{R}|)$, *where*

- $\mathfrak{m}_\mathrm{R}$ *the multiset of arities of* ? *in* R*;*
- $|\mathrm{R}|$ *the number of cells in* R.

$$\begin{tikzcd}
(\rho_i)_{1\leqslant i\leqslant n} \arrow[r, dashed] \arrow[d, maps to] & (\rho'_i)_{1\leqslant i\leqslant m} \arrow[d, dashed, maps to] \\
\mathrm{R} \arrow[r, dashed] & \mathrm{R}'
\end{tikzcd}$$

Proof: If R has a conclusion i which is the conclusion of a multiplicative, in all the ρ_ℓ, the conclusion in i is the same multiplicative (as it is not in a box), and we can apply the rule associated to it in both R and all the ρ_ℓ. The obtained proof-net as one less cell, so is strictly smaller.

In the same way, if R contains a cut outside of a box, we can apply the `cut` rule on all the proof-nets, which makes the MELL proof-net decrease.

So, we might suppose that all conclusions of R are either ! or ?. There is at most one box by connected component of R. If there is a !-conclusion i such that, in all the ρ_ℓ, all its connected component is accessible from the premises of i, we can apply the rule box_i to R and $!_i$ in all ρ_ℓ. Else, let j be one conclusion whose premises are not accessible in one of the ρ_ℓ. If j is the border of a box in R, we apply a ?-split in i, else, we apply a contraction in i. ■

Theorem 13

A set of fat polyadic proof-nets is gluable if and only if it is in the Taylor expansion of a box-connected

MELL *proof-net.*

Proof: Let $(\rho_i)_{1\leqslant i\leqslant n}$ be a clique of fat polyadic proof-nets. By the definition of coherence, there exists a reduction path $(\rho_i)_{1\leqslant i\leqslant n} \to \varepsilon$. By induction on the length of the path, and by repeatedly applying Proposition 4, $(\rho_i)_{1\leqslant i\leqslant n}$ is in the Taylor expansion of a MELL proof-net R.

Reciprocally, let $(\rho_i)_{1\leqslant i\leqslant n} \mapsto \mathrm{R}$. By Proposition 5, $(\rho_i)_{1\leqslant i\leqslant n}$ reduces to $(\rho_i')_{1\leqslant i\leqslant m}$ and R reduces to R' such that

$$\begin{tikzcd}
(\rho_i)_{1\leqslant i\leqslant m} \arrow[r, dashed] \arrow[d, mapsto] & (\rho_i')_{1\leqslant i\leqslant n} \arrow[d, dashed, mapsto] \\
\mathrm{R} \arrow[r, dashed] & \mathrm{R}'
\end{tikzcd}$$

As R' is strictly smaller in the well-ordering, this construction can be carried to a smaller element, which can only be ε. So the ρ_{ell} are glueable. ■

6.4 Coherence and glueability

6.4.1 Glueability is a relation between an arbitrary (even possibly infinite) number of polyadic proof-nets, whereas coherence is binary. Before linking the two, we first state results about glueability.

Lemma 13

Let ρ *be a 1-fat polyadic proof-net.*

All MELL *proof-nets projection of a glueable family of polyadic proof-nets containing* ρ *have isomorphic underlying graph (and so, only differ by their boxing function).*

Proof: By induction on the number of !-cells in ρ. ■

The following Lemma is a restatement of Theorem 12.

Lemma 14

Let ρ *be a 2-fat polyadic proof-net.*

All MELL *proof-nets projection of a glueable family of polyadic proof-nets containing* ρ *are isomorphic.*

6.4.2 We defined glueability as family of n-ary relations, but it really is generated by a binary coherence relation.

Definition 55 (*Coherence*)

Let ρ_1 and ρ_2 be two 1-fat polyadic proof-nets. We say that ρ_1 and ρ_2 are coherent, written

$$\rho_1 \frown \rho_2$$

if the family $\{\rho_1, \rho_2\}$ is glueable.

A glueable family is obviously a $\frown\!\!\!\!\smile$-clique: a rewriting path ending in ε for a family is a rewriting path ending in ε for any of its subsets. We conjecture that the converse is true.

6.5 Glueability of arbitrary polyadic proof-nets

6.5.1 By adding $\maltese$ nodes, we reach arbitrary polyadic proof-nets.

Definition 56

Let ϕ be a daimonless polyadic proof-net. We say that a polyadic proof-net with daimons ρ is a $\maltese$-fattening of ϕ if:

- all the daimon cells in ρ have exactly one output which is is the lone input of a !-cell, and all the other outputs of a daimon cell are inputs of ?-cells;
- ϕ is ρ stripped of all its daimon cells.

Definition 57

Let $(\rho_\ell)_{\ell in \mathrm{I}} \in \mathbf{PolyPN}^*_{\maltese}$ be a list of daimonless polyadic proof-nets. We say that they are **glueable** if there is a glueable family $(\rho'_\ell)_{\ell in \mathrm{I}} \in \mathbf{PolyPN}^*_{\maltese}$ of fat polyadic proof-net such that, for all $\ell in \mathrm{I}$, ρ'_ℓ is a $\maltese$-fattening of ρ_ℓ.

We define MELL proof-nets with daimons in the obvious way: by adding $\maltese$ cells that can only inside boxes of which they are the exclusive content, and their Taylor expansion.

Theorem 14

Let $(\rho_\ell)_{\ell in \mathrm{I}} \in \mathbf{PolyPN}^*_{\maltese}$ *be a list of daimonless polyadic proof-nets.*

$(\rho_\ell)_{\ell in \mathrm{I}}$ *are gluable if and only if they are in the Taylor expansion of a box-connected* MELL *proof-net.*

7

Relational model and box-connexity

7.1 Resource proof-nets

7.1.1 We introduced the polyadic Taylor expansion of proof-nets as a tool to study their resource Taylor expansion. We can now lift all the results of the Chapter to resource proof-nets and the resource Taylor expansion of box-connected proof-structures.

Reduction

7.1.2 Reduction of resource proof-nets is inherently non-deterministic. Indeed, consider the reduction of Figure 4.6(c), for polyadic proof-nets. Given a resource proof-net which is the quotient of the left projection of this reductive polyadic proof-net, there is a reduction for each permutation of n elements, which all have a different right projection. So, we consider that a resource proof-net reduces to a set of resource proof-nets.

Definition 58 (*equivalence of polyadic reductive proof-net*)

> Two polyadic reductive proof-net are equivalent if their left projection are equivalent.
>
> A resource reductive proof-net is an equivalence class of polyadic reductive proof-nets.

It is important to note that the set of left-hand sides of a resource proof-net is not an equivalence class. The image of a resource proof-net under a resource reductive proof-net is the reunion of all the left-hand sides.

Fat resource proof-nets and box-connexity

7.1.3 The definition of fatness is stable by $\simeq_{!?}$-equivalence, so we define n-fat resource proof-nets as resource proof-nets whose representative polyadic proof-nets are n-fat. It is immediate that Theorem 12 still holds in the quotient:

Theorem 15

> *Let* R *and* R' *be two box-connected* $\mathsf{MELL}_?$ *proof-nets, and* $\rho \in \mathscr{T}_{\mathrm{R}}^{!?}$, $\rho' \in \mathscr{T}_{\mathrm{R}'}^{!?}$ *that are 2-fat, and such that* $f : \rho \simeq_{!?} \rho'$.
>
> f *induces an isomorphism of* $\mathsf{MELL}_?$ *proof-nets* $\mathrm{R} \simeq \mathrm{R}'$.

Coherence

7.1.4 Coherence too, still makes sense in the quotient.

Definition 59 (*coherence of resource proof-nets*)

> Two resource proof-nets ρ and ρ' are coherent if there exists two polyadic proof-nets π and π', representing respectively ρ and ρ', such that $\pi \mathrel{\smile\!\!\!\!\!\frown} \pi'$.

Theorem 16

> *A family of resource proof-nets is a* $\smile\!\!\!\!\!\frown$*-clique if and only if it is in the Taylor expansion of a box-connected* $\mathsf{MELL}_?$ *proof-net.*

7.2 The relational model

Abstractly

7.2.1 Recall that, for MELL proof-nets, the type system of (simply-typed) polyadic proof-nets enjoys both subject reduction and subject expansion. Moreover, as we saw, if we are to consider MELL proof-nets through $\simeq_?$, we have to consider not polyadic proof-nets, but polyadic proof-nets quotiented through the $\simeq_{!?}$ equivalence, that is resource proof-nets.

The morphism of cyclic **Cat**-operads

$$\mathscr{T} : \mathsf{MELL} \to \mathfrak{Dist}_s$$

can be composed with the morphism induced by the $\simeq_{!?}$ equivalence, yielding a morphism

$$\mathscr{T}_{!?} : \mathsf{MELL}_? \to \mathfrak{Dist}_s$$

that associates to a MELL proof-net (quotiented by $\simeq_?$) the set of resource proof-nets in its resource Taylor expansion.

7.2.2 We can go further and build a denotational model from this functor. Indeed, we can forget precisely which resource proof-net approximates a MELL proof-net, but just remembering that there exists one. In this way, all the reductions collapse.

Let us consider the operad $\mathscr{R}el$ whose objects are sets and multiarrows $A_1, \ldots, A_n \to B$ two-valued distributors $A_1 \times \cdots \times A_n \times B \to \{\emptyset, \star\}$ (that we can alternatively view as relations). We will view $\mathscr{R}el$ as a **Cat**-operad by adding trivial 2-arrows.

There is an obvious morphism of cyclic **Cat**-operads from the sub-operad of $\mathfrak{Dist}_s$ consisting of sets, multi-distributors and bi-entire relations to $\mathscr{R}el$:

- on objects, it is the identity;
- on a multi-arrow, it is the post-composition with the terminal function.

It is a morphism because the relations are bi-entire. So, by post-composing $\mathscr{T}_{!?}$ with this functor, we get a morphism of cyclic **Cat**-operads that identifies all images of multi-arrows related by a 2-arrow:

$$[\![\cdot]\!] : \mathsf{MELL}_? \to \mathscr{R}el$$

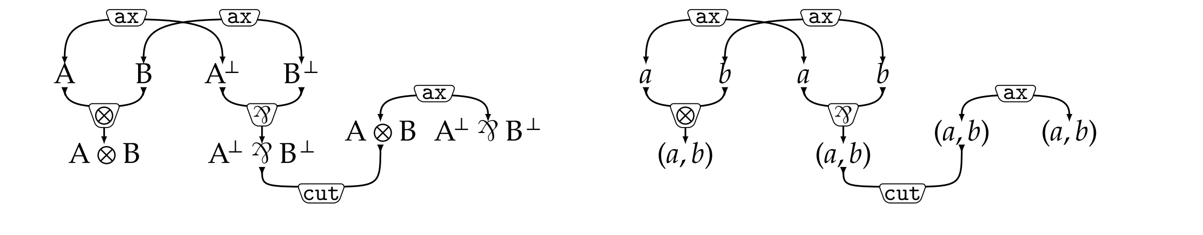
7.2.3 We can describe this functor more explicitly. First of all, we will describe its action on formulæ.

Proposition 6

For all formulæ A, B,

$$\begin{aligned}[\![A \otimes B]\!] \simeq [\![A \parr B]\!] &\simeq [\![A]\!] \times [\![B]\!]\\ [\![!A]\!] \simeq [\![?A]\!] &\simeq \mathscr{M}_{\text{fin}}([\![A]\!])\\ [\![A^\perp]\!] &\simeq [\![A]\!].\end{aligned}$$

Proof: All these bijections come from the fact that we require all our constructions to be compatible with extensionality. ■



(a) The proof-net R

(b) An experiment of R

Figure 7.1: A MLL proof-structure R and an experiment of R

7.2.4 Experiments have been introduced by Girard (1987, p. 3.16) to define the coherent semantics of linear logic proof-nets without having to go through sequent calculus. The tradition since has been to represent experiments as functions, defined on the wires of proof-nets, that associate to each wire the partial interpretation of the net it represents, as is shown in Figure 7.1. The content of the boxes are treated as distinct proof-nets, which are also susceptible to be experimented on, a multiple number of times. So, this gives the definition, by induction on the depth of a proof-net (the maximal nesting of boxes):

Definition 60 (*experiment of resource proof-nets*)

Let R be a $\mathsf{MELL}_?$ proof-net.

An **experiment** of R is a function e on the edges outside of any boxes of R satisfying the following conditions on cells outside of any box:

- for all cell c of type $\mathtt{ax}$, with o_1 and o_2 its two outputs, $\mathsf{e}(o_1) = \mathsf{e}(o_2)$;
- for all cell c of type $\mathtt{cut}$, with i_1 and i_2 its two inputs, $\mathsf{e}(i_1) = \mathsf{e}(i_2)$;
- for all cell c of type $\mathbf{1}$, with o its only input, $\mathsf{e}(o) = \{*\}$;
- for all cell c of type $\bot$, with o its only input, $\mathsf{e}(o) = \{*\}$;
- for all cell c of type $\otimes$ or $\invamp$, with $i_1 < i_2$ its two inputs and o its output, $\mathsf{e}(o) = (\mathsf{e}(i_1), \mathsf{e}(i_2))$;
- for all cell c of type ! or ?, with $i_1, \dots, i_n$ be its inputs and o its output, $\mathsf{e}(o) = [\mathsf{e}(i_1), \cdots, \mathsf{e}(i_n)]$.

We verify easily that the image of an edge is in the *web* of its type, where the web of a type is defined recursively:

- for an atom A, the web $[\![\mathrm{A}]\!]$ is any infinite countable set;
- $[\![\bot]\!] = [\![\mathbf{1}]\!] = \{*\}$;
- $[\![\mathrm{A} \otimes \mathrm{B}]\!] = [\![\mathrm{A} \invamp \mathrm{B}]\!] = [\![\mathrm{A}]\!] \times [\![\mathrm{B}]\!]$;
- $[\![!\mathrm{A}]\!] = [\![?\mathrm{A}]\!] = \mathscr{M}_{\mathrm{fin}}([\![\mathrm{A}]\!])$.

7.2.5 The conclusions of a resource proof-net are ordered: this allows to define unequivocally the result of an experiment on it.

Definition 61 (*result of an experiment*)

Let R be a resource proof-net with conclusions $o_1 < \cdots < o_n$ and e be an experiment of R.

The **result** of e on R is the tuple $(\mathsf{e}(o_1), \cdots, \mathsf{e}(o_n))$.

Let R be a pointed $\mathsf{MELL}_?$ proof-net with distinguished conclusion o and other conclusions $o_1 < \cdots < o_n$ and e be an experiment of R.

The **result** of e on R is the couple $\mathsf{e}(\mathrm{R}) = ((\mathsf{e}(o_1), \cdots, \mathsf{e}(o_n)), \mathsf{e}(o))$.

So, the $[\![\cdot]\!]$ functor associates, to a formula A, its web and to a $\mathsf{MELL}_?$ proof-net R the relation:

$$[\![\mathrm{R}]\!] = \left\{\mathsf{e}(\rho) \mid \rho \in \mathscr{T}_{\mathrm{R}}^{!?}, \mathsf{e} \text{ experiment of } \rho\right\}$$

7.3 Models of linear logic

7.3.1 The cyclic **Set**-operad $\mathscr{R}el$ has more structure than just being endowed with an arrow $\mathsf{MELL}_? \to \mathscr{R}el$. Indeed, it is a *denotational model* of multiplicative-exponential linear logic, which means not only that linear logic proofs can be interpreted into $\mathscr{Rel}$, but also that $\mathscr{Rel}$ behaves as a logical system obeying the same rules as $\mathsf{MELL}_?$. We will present that observation more formally.

There are different notions of models of linear logic (Seely categories, Lafont categories, linear-non-linear adjunctions… see (Melliès 2008) for a nice survey). We will only describe Lafont categories.

Definition 62 (*Lafont category*)

A **Lafont category** is a symmetric monoidal category such that, for every object A, there exists a free commutative comonoid !A, that is, an object !A endowed with maps

$$\begin{tikzcd}
!A \arrow[r, "\delta"] & !A \otimes !A \\
!A \arrow[r, "\eta"] & \mathbf{1} \\
!A \arrow[r, "\varepsilon"] & A
\end{tikzcd}$$

such that the swap of the category followed by δ is equal to δ and for every commutative comonoid X and morphism $f : X \to A$, there exists a unique comonoid morphism $f^\dagger : X \to\ !A$ such that

$$\begin{tikzcd}
 & !A \arrow[dd, "\varepsilon"] \\
X \arrow[ur, "f^\dagger"] \arrow[dr, "f"'] & \\
 & A
\end{tikzcd}$$

commutes.

Coherence spaces

Coherence spaces are the original semantics of linear logic, from which it sprung.

Definition 63 (Coh, *Girard (1987, section 3)*)

A *coherence space* is a pair $A = (|A|, \mathrel{\substack{\frown\\\smile}}_A)$ of a set $|A|$ (the *web* of A) and a reflexive binary relation $\mathrel{\substack{\frown\\\smile}}_A$ on the elements of $|A|$, the *coherence* of A. A *clique* of A is a set of pairwise coherent elements of $|A|$.

Every coherence space A has a *dual* coherence space $A^\perp$ with same web $|A|$ and coherence relation

$$a \mathrel{\substack{\frown\\\smile}}_{A^\perp} b \Leftrightarrow a = b \text{ or } \neg(a \mathrel{\substack{\frown\\\smile}}_A b)$$

The coherence of the dual is called the *incoherence* of the primal and written $a \asymp_A b$.

For every coherence spaces A and B, we define their *tensor product* as the coherence space

A ⊗ B with web |A| × |B| and coherence defined by

$$a \otimes b \frown_{A \otimes B} a' \otimes b' \Leftrightarrow a \frown_A a' \text{ and } b \frown_B b'$$

where $a \otimes b$ is just a semantically-flavoured notation for the couple (a, b).

For every coherence spaces A and B, we define their *cartesian product* as the coherence space A & B with web |A| + |B| and coherence defined by

$$\begin{aligned} &a \frown_{A\&B} a' \Leftrightarrow a \frown_A a' \\ &b \frown_{A\&B} b' \Leftrightarrow b \frown_B b' \\ &a \frown_{A\&B} b \end{aligned}$$

The category **Coh** is defined as the category with coherence spaces as objects and cliques of $A \multimap B = (A \otimes B^{\perp})^{\perp}$ as morphisms. For clarity's sake, we explicit the coherence of A ⊸ B.

$$a \multimap b \frown_{A \multimap B} a' \multimap b' \Leftrightarrow \begin{cases} a \frown_A a' \Rightarrow b \frown_B b' \\ a \asymp_A a' \Rightarrow b \asymp_B b' \end{cases}$$

where $a \multimap b$ is again a notation for (a, b). There is a forgetful functor from cliques of A ⊸ B to relations of A and B. As such, cliques compose as relations.

It is a Lafont category with finite products.

Finiteness spaces

Definition 64 (Fin, *Ehrhard (2005)*)

Let E be a countable set. Two subsets u and $u' \subseteq E$ are *orthogonal* (written $u \perp u'$) when their intersection $u \cap u'$ is finite.

The *orthogonal* of a family $\mathscr{F}$ of subsets of E is then defined as

$$\mathscr{F}^{\perp} = \{u \in E | \forall v \in \mathscr{F}, \#(u \cap v) < \aleph_0\}$$

A *finiteness space* is a couple $(|E|, \mathsf{F}(E))$ of a set |E|, the *web* of E and $\mathsf{F}(E) \subseteq \mathfrak{P}(E)$ the *finiteness* of E, a family verifying $\mathsf{F}(E)^{\perp\perp} = \mathsf{F}(E)$. The elements of $\mathsf{F}(E)$ are called *finitary*, the ones of $\mathsf{F}(E)^{\perp}$ are called *antifinitary*.

For every finiteness space E, we define its *dual* by $E^{\perp} = (|E|, \mathsf{F}(E)^{\perp})$.

For every finiteness spaces E and F, we define their *tensorial product* as the space E ⊗ F with web |E| × |F| and finiteness defined by:

$$\mathsf{F}(E \otimes F) = \{w \subseteq |E| \times |F|, \pi_E(w) \in \mathsf{F}(E), \pi_F(w) \in \mathsf{F}(F)\}$$

where π_E and π_F are the usual projections. The unit of the tensor product is defined by $|\mathbf{1}| = \{*\}$ and $\mathsf{F}(\mathbf{1}) = \{\emptyset, \{*\}\}$.

For every finiteness spaces E and F, we define their *cartesian product* as the space E & F with web $|E| \sqcup |F|$ and finiteness $\mathsf{F}(E) \sqcup \mathsf{F}(F)$.

We say that a relation $R \subseteq |E| \times |F|$ between two finiteness spaces E and F is *finitary* if

$$\begin{aligned} &\forall u \in \mathsf{F}(E), R(u) = \{f \in |F|, \exists e \in u, eRf\} \in \mathsf{F}(F) \\ &\forall v \in \mathsf{F}(F)^{\perp}, {}^{t}R(v) = \{e \in |E|, \exists f \in u, eRf\} \in \mathsf{F}(E)^{\perp} \end{aligned}$$

The *linear implication* $E \multimap F$ is defined as the space with web $|E| \times |F|$ and finiteness structure $\mathsf{F}(E \multimap F)$ the set of finitary relations between E and F.

The category **Fin** of finiteness spaces has finiteness spaces as objects and finitary relations as morphisms. It is a Lafont category with finite products.

Conway Games

Definition 65 (Conway^{-})

A *Conway game* A is an oriented rooted well-founded graph (V_A, E_A, λ_A) consisting of a set V_A of vertices (the *positions* of the game), a set $E_A \subseteq V_A \times V_A$ of edges (the *moves* of the game), and a function $\lambda_A : E_A \to \{-1; +1\}$ indicating wether a move is played by Opponent (-1) or Proponent ($+1$). We write $\star_A$ for the root of the underlying graph. A Conway game is called *negative* when all its moves starting from the root are played by Opponent.

A *play* $s = m_1 \cdot m_2 \cdot \cdots \cdot m_k$ of a Conway game A is a path $s : \star_A \twoheadrightarrow x_k$ starting from the root $\star_A$ whose vertices are labeled by $m_1,\ldots,m_k$. A play is *alternating* when $\forall i \in \{1, ..., k-1\}, \lambda_A(m_{i+1}) = -\lambda_A(m_i)$.

Every Conway game A induces a *dual* game $A^{\perp}$ by reversing the polarities of the moves.

For every Conway games A and B, we define their *tensor product* as the game $A \otimes B$ with set of vertices $V_{A\otimes B} = V_A \times V_B$, moves defined as

$$x \otimes y \to \begin{cases} z \otimes y \text{ if } x \to z \text{ in the game A} \\ x \otimes z \text{ if } y \to z \text{ in the game B} \end{cases}$$

and the polarity of a move in $A \otimes B$ defined as the polarity of the underlying move in the component A or the component B. The Conway game **1** with a unique position $\star$ and no moves is the neutral element of the tensor product.

For every negative Conway games A and B, we define their *cartesian product* $A \& B$ as the game whose

- set of positions is the amalgamated sum of the positions of A and B under the initial position;
- Opponent moves from the initial position $\star_{A\&B}$ are of two kinds

$$\star_{A\&B} \to \begin{cases} x \text{ if } \star_A \to x \text{ in the game A} \\ y \text{ if } \star_B \to y \text{ in the game B} \end{cases}$$

- moves from a position x in the component A (respectively B) are exactly the moves from x in the Conway game A (respectively B), with the same polarity.

A *strategy* of a Conway game A is defined as a non empty set of *alternating plays* of even length such that every non-empty play starts with an Opponent move, σ is closed by even length prefix, and σ is deterministic, that is, for all plays s and all moves m, n, n',

$$s \cdot m \cdot n \in \sigma \wedge s \cdot m \cdot n' \Rightarrow n = n'$$

The category **Conway**$^{-}$ has negative Conway games as objects and strategies σ of $A^{\perp} \otimes B$ as morphisms $\sigma : A \to B$. It is symmetric monoidal closed and has finite and infinite products.

Lafont categories and approximations

7.3.2 Each monoidal category $\mathscr{C}$ defines a **Set**-operad whose multimorphisms $a_1, \dots, a_n \to b$ are the morphisms $a_1 \otimes \cdot \otimes a_n \to b$ in the original category. If it endowed by a duality, that is a full and faithful functor

$$(\cdot)^* : \mathscr{C}^{\mathrm{op}} \to \mathscr{C}$$

such that there is a natural isomorphism, for all $\mathrm{A}, \mathrm{B}, \mathrm{C} \in \mathscr{C}$, $\mathscr{C}(\mathrm{A} \otimes \mathrm{B}, \mathrm{C}^*) \simeq \mathscr{C}(\mathrm{A}, (\mathrm{B} \otimes \mathrm{C})^*)$, it even defines a cyclic **Set**-operad. Moreover:

Proposition 7

Let $\mathscr{C}$ be a Lafont category with a duality. By abuse of notation, we will also write $\mathscr{C}$ its associated cyclic operad. There exists a morphism of cyclic operad

$$\mathsf{MELL}_? \to \mathscr{C}.$$

7.3.3 We built the relational model and interpretations of proof-nets into it through approximations: we defined the interpretation of an approximation, and then transfered the definition to $\mathsf{MELL}_?$. Alas, it is not possible for the other semantics we presented: more precisely, by returning to Girard's approximation's theorem that we touched in the introduction, we can say that defining brutally the interpretation of a $\mathsf{MELL}_?$ proof-net as the union of the interpretation of its approximants amounts to consider the exponential of A to be equal to the projective limit (the categorical generalization of the union) of the $\mathrm{A}^{\otimes n}$. It is indeed the case in the relational model, where we have

$$!\mathrm{A} = \bigcup_{n \in \mathbf{N}} \mathrm{A}^{\otimes n} / \mathfrak{S}_n.$$

(where the quotient denotes the action of the symmetric group by permutation), but not in the other models we presented.

7.4 Box-connected injectivity

Injectivity and full completeness, or Syntax vs. Semantics

7.4.1 If we consider a calculus and a denotational semantics for it

$$f : \mathscr{C} \to \mathscr{D}$$

we can ask whether every multi-arrow in $\mathscr{D}$ is the image of a term of $\mathscr{C}$, that is, whether the syntax allows to represent everything that the semantics contain?

This problem is called *full completeness*.

7.4.2 Let us consider a calculus $\mathscr{C}$. If the calculus is confluent, there is an equivalence induced on the terms by the reduction: two terms (of the same type) are equivalent if they reduce to the same normal form.

If we consider moreover a denotational semantics for $\mathscr{C}$,

$$f : \mathscr{C} \to \mathscr{D}$$

another equivalence arises: two terms are equivalent if they have the same image by f. By definition of a denotational model, this equivalence is coarser than the syntactic one. If they coincide, the model is said *injective*.

7.4.3 A model both fully complete and injective would be a representation of the normal terms of the calculus.

7.4.4 The problem of injectivity of the relational model has been positively solved by Carvalho (2016). To distinguish to MELL proof-nets, it uses elements of the relational model of a complexity that depends on the proof-nets.

The box-connected case

7.4.5 The box-connected case is very different. Indeed, as 2-fat points of the relational model characterize reduction classes of resource proof-nets, that resource proof-nets characterize equivalence classes of polyadic proof-nets, and 2-fat polyadic proof-nets characterize the MELL box-connected proof-net whose Taylor expansion they are in, we get the theorem:

Theorem 17

Let R *and* R' *be two box-connected proof-net.* $\mathrm{R} \simeq_? \mathrm{R}'$ *if and only if they have the same 2-point.*

In particular, the relational model is injective for box-connected proof-nets.

8

Semantics viewed as approximations

❦

8.0.1 We have left two problems open, when trying to construct semantics from approximations:

- the set of approximants of a λ-term does not have a structure of a model of the λ-calculus;
- usual semantics of linear logic can not be defined through approximations.

We will now consider one solution to each of these problems, the first due to Mazza (2012), the second one to Melliès, Tabareau, and Tasson (2009), and show that they are actually very related.

8.1 The infinitary affine polyadic λ-calculus

A limit of approximations to get back the exponential

8.1.1 It is possible to view the exponential as a limit of polyadic terms, due to Mazza (2012), topological, and directly based on the syntax. Consider the **Cat**-operad **AffPoly** of §4.1.4, extended by a special term $\perp$, that can not be substituted. This calculus is affine, in the sense that no duplication is performed, and in fact it strongly normalizes even in absence of types (the size of terms strictly decreases with reduction).

The set of terms is equipped with the structure of *uniform space*, a generalization of a metric space that has a well-behaved notion of Cauchy sequence. The Cauchy-completion of the set of terms, denoted by $\Lambda^{\text{aff}}_{\infty}$, contains infinitary terms, i.e. terms allowing infinite sequences $\langle u_1, u_2, u_3, \dots\rangle$. The original calculus embeds (and is dense) in $\Lambda^{\text{aff}}_{\infty}$ by considering a finite sequence as an almost-everywhere $\perp$ sequence. Reduction, which is continuous, is defined as above, except that infinitely many substitutions may occur. This yields non-termination, in spite of the calculus still being affine: if $\Delta_n := \lambda x.x_0\langle x_1, \dots, x_n\rangle$, then $\Delta := \lim_{n\to\infty} \Delta_n = \lambda x.x_0\langle x_1, x_2, x_3, \dots\rangle$ and $\Omega := \Delta\langle\Delta, \Delta, \Delta, \dots\rangle \to \Omega$.

8.1.2 Ideally, these infinitary terms should correspond to usual λ-terms. But there is a continuum of them, definitely too many. The solution is to consider a partial equivalence relation $\simeq$ such that, in particular, $x_i \simeq x_j$ for all i, j and $t\langle u_1, u_2, u_3, \dots\rangle \simeq t'\langle u'_1, u'_2, u'_3, \dots\rangle$ whenever $t \simeq t'$ and, for all $i, i' \in \mathbf{N}$, $u_i \simeq u'_{i'}$. After introducing a suitable notion of reduction $\Rightarrow$ on the equivalence classes of $\simeq$, one finally obtains a term calculus for intuitionistic affine logic, isomorphic to a calculus similar to that of (Benton et al. 1993). In particular, Girard's translation of intuitionistic logic in linear logic may be encapsulated in the construction, as done in (Mazza 2012), obtaining the subcalculus generated by

$$t, u ::= x_i \mid \lambda x.t \mid t\langle u_1, u_2, u_3, \dots\rangle,$$

which we denote by $\Lambda^{\text{aff}}_{\infty}$. Then, one has the isomorphism for the reduction relations

$$(\Lambda^{\text{aff}}_{\infty}/{\simeq}, \Rightarrow) \quad \cong \quad (\Lambda, \to_\beta),$$

where $(\Lambda, \to_\beta)$ is the usual pure λ-calculus with β-reduction. Similar infinitary calculi (also with a notion of partial equivalence relation) were considered by Kfoury (2000) and Melliès (2004), although without a topological perspective. The indices identifying the occurrences of exponential variables are also reminiscent of Abramsky, Jagadeesan, and Malacaria (2000) games semantics.

So, the infinitary polyadic λ-calculus manages to build usual λ-terms as a quotient of a topological limit of polyadic λ-terms. In a sense, it consists of enriching the space of approximations of λ-terms with infinitary approximations.

8.2 Functorial semantics and Kan extensions

Functorial semantics

8.2.1 The idea of functorial semantics is to describe an algebraic theory as a certain category constituted of the different powers of the domain of the theory as the objects, the operations of the theory as morphisms, and encode the relations between the operations in the composition operation. We will not consider algebraic theories as Lawvere did, but the more general symmetric monoidal theories, or PROPs – standing for *product and permutation categories* (MacLane 1965) .

Definition 66 (*symmetric monoidal theory*)

An *n-sorted symmetric monoidal theory* is defined as a symmetric monoidal category $\mathbb{T}$ whose objects are n-tuples of natural numbers and with a tensorial product defined as the point-wise arithmetical sum.

A *model* of $\mathbb{T}$ in a symmetric monoidal category (SMC) $\mathscr{C}$ is a symmetric strong monoidal functor $\mathbb{T} \to \mathscr{C}$.

> A *morphism* of models of $\mathbb{T}$ in $\mathscr{C}$ is a monoidal natural transformation between models of $\mathbb{T}$ in $\mathscr{C}$. We will denote as $\mathrm{Mod}(\mathbb{T}, \mathscr{C})$ the category with models of $\mathbb{T}$ in $\mathscr{C}$ as objects and morphisms between models as morphisms.

8.2.2 The simplest symmetric monoidal theory, denoted by $\mathbb{B}$, has as objects the natural numbers seen as finite ordinals and as morphisms the bijections between them (the permutations). Alternatively, $\mathbb{B}$ can be seen as the free symmetric monoidal category on one object (the object 1, with monoidal unit 0). As such, a model of $\mathbb{B}$ is nothing but an object A in a symmetric monoidal category $\mathscr{C}$, and the categories $\mathscr{C}$ and $\mathrm{Mod}(\mathbb{B}, \mathscr{C})$ are equivalent.

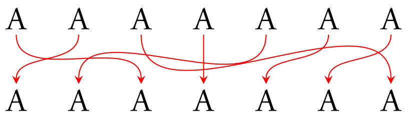
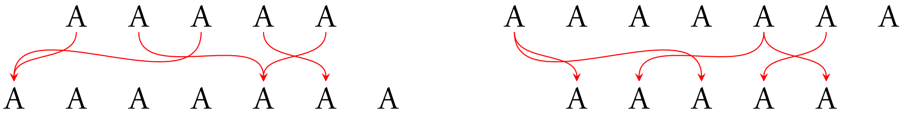


Figure 8.1: A morphism $7 \to 7$ of $\mathbb{B}$, presented on an object $A : \mathscr{C}$

8.2.3 The key non-trivial example in our context is that of commutative (co)monoids. We remind that a commutative monoid in a SMC $\mathscr{C}$ is a triple $(A, \mu : A \otimes A \to A, \eta : I \to A)$, with A an object of $\mathscr{C}$, such that the arrows μ and η interact with the associator, unitors and symmetry of $\mathscr{C}$ to give the usual laws of associativity, neutrality and commutativity (see MacLane 1978). A morphism of monoids $f : (A, \mu, \eta) \to (A', \mu', \eta')$ is an arrow $f : A \to A'$ such that $f \circ \mu = \mu' \circ (f \otimes f)$ and $f \circ \eta = \eta'$. We denote the category of monoids of $\mathscr{C}$ and their morphisms as $\mathrm{Mon}(\mathscr{C})$. The dual notion of comonoid, and the relative category $\mathrm{Comon}(\mathscr{C})$, is obtained by reversing the arrows in the above definition. Now, consider the symmetric monoidal theory $\mathbb{F}$ whose objects are the natural numbers seen as finite ordinals and its morphisms are the functions between them ($\mathbb{F}$ is the skeleton of the category of finite sets). We easily check that $\mathrm{Mod}(\mathbb{F}, \mathscr{C}) \simeq \mathrm{Mon}(\mathscr{C})$ and $\mathrm{Mod}(\mathbb{F}^{\mathrm{op}}, \mathscr{C}) \simeq \mathrm{Comon}(\mathscr{C})$. Indeed, a strict symmetric monoidal functor from $\mathbb{F}$ to $\mathscr{C}$ picks an object of $\mathscr{C}$ and the image of any arrow $m \to n$ of $\mathbb{F}$ is unambiguously obtained from the images of the unique morphisms $0 \to 1$ and $2 \to 1$ in $\mathbb{F}$, which are readily verified to satisfy the monoid laws.



(a) A morphism $5 \to 7$ of $\mathbb{F}$

(b) A morphism $7 \to 5$ of $\mathbb{F}^{\mathrm{op}}$

Figure 8.2: $\mathbb{F}$ and $\mathbb{F}^{\mathrm{op}}$, presented on an object $A : \mathscr{C}$

Summing up, finding the free commutative comonoid A^{∞} on an object A of a SMC $\mathscr{C}$ is the same thing as turning a strict symmetric monoidal functor $\mathbb{B} \to \mathscr{C}$ into a strict symmetric monoidal functor $\mathbb{F}^{\mathrm{op}} \to \mathscr{C}$ which is universal in a suitable sense. This is where Kan extensions come into the picture.

Kan extensions

8.2.4 Kan extensions allow to extend a functor along another. Let $K : \mathscr{C} \to \mathscr{D}$ and $F : \mathscr{C} \to \mathscr{E}$ be two functors. Let us suppose we have the following diagram in **Cat**:

$$\begin{tikzcd}
\mathscr{C} \arrow[rr, "F"] \arrow[dr, "K"] & & \mathscr{E} \\
& \mathscr{D} &
\end{tikzcd}$$

If we think of K as an inclusion functor, it seems natural to try to define a functor $\mathscr{D} \to \mathscr{E}$ that would in a sense be universal among those that extend F. There are two ways of formulating this statement precisely, yielding *left* and *right Kan extensions.*

Definition 67 (*Kan extension*)

Let $\mathscr{C}, \mathscr{D}, \mathscr{E}$ be three categories and $F : \mathscr{C} \to \mathscr{E}$, $K : \mathscr{C} \to \mathscr{D}$ two functors.

A **left Kan extension of** F **along** K is a functor $\mathrm{Lan}_K F : \mathscr{D} \to \mathscr{E}$ together with a natural transformation $\eta : F \Rightarrow \mathrm{Lan}_K F \circ K$ such that for any other pair $(G : \mathscr{D} \to \mathscr{E}, \gamma : F \Rightarrow G \circ K)$, γ factors uniquely through η:

$$\begin{tikzcd}
\mathscr{C} \arrow[rr, "K"] \arrow[dr, bend right, "F"', ""{name=F}] & & \mathscr{D} \arrow[dl, bend left, "G", ""{name=G, swap}] \arrow[dl, dashed, ""{name=L, swap}] \\
& \mathscr{E} &
\arrow[Rightarrow, from=F, to=L, bend left, "\eta"]
\arrow[Rightarrow, from=1-2, to=G]
\end{tikzcd}$$

The **right Kan extension of** F **along** K is a functor $\mathrm{Ran}_K F : \mathscr{D} \to \mathscr{E}$ together with a natural transformation $\varepsilon : \mathrm{Ran}_K F \circ K \Rightarrow F$ such that for any other pair $(G : \mathscr{D} \to \mathscr{E}, \gamma : G \circ K \Rightarrow F)$, γ factors uniquely through ε:

$$\begin{tikzcd}
\mathscr{C} \arrow[rr, "K"] \arrow[dr, bend right, "F"', ""{name=F}] & & \mathscr{D} \arrow[dl, bend left, "G", ""{name=G, swap}] \arrow[dl, dashed, ""{name=R, swap}] \\
& \mathscr{E} &
\arrow[Rightarrow, from=R, to=F, bend right, "\varepsilon"']
\arrow[Rightarrow, from=G, to=R]
\end{tikzcd}$$

These two notions are obviously dual. We will only be interested in right Kan extensions.

It is easy to check that $\mathbf{Cat}(G, \mathrm{Ran}_K F) \simeq \mathbf{Cat}(G \circ K, F)$ (with **Cat** seen as a 2-category of small categories, functors and natural transformations). In other words, Ran_K is right adjoint to U_K, the functor precomposing with K (whence the terminology "right"—the left adjoint to U_K is the left Kan extension). This observation is important because it tells us that Kan extensions may be relativized to any 2-category. In particular, we may speak of *symmetric monoidal Kan extensions* by taking the underlying 2-category to be **SymMonCat** (symmetric monoidal categories, strict symmetric monoidal functors and monoidal natural transformations).

8.2.5 Kan extensions can be explained very elegantly via distributors (see §5.1.2). We first fix some terminology.

We will view allow ourselves to view a distributor $F : \mathscr{C} \nrightarrow \mathscr{D}$ as a presheaf $\widehat{F} : \mathscr{D} \to \widehat{\mathscr{C}}$.

We say that a distributor $F : \mathscr{C} \nrightarrow \mathscr{D}$ is *represented* by a functor $\phi : \mathscr{C} \to \mathscr{D}$ if F is the curryfication of the composite

$$\begin{tikzcd} \mathscr{C} \arrow[r, "F"] & \mathscr{D} \arrow[r, "Y"] & \widehat{\mathscr{D}} \end{tikzcd}$$

where $\widehat{\mathscr{D}}$ is the category of pre-sheafs over $\mathscr{D}$ and Y the Yoneda embedding. A distributor is said to be *representable* if it is represented by a functor.

The following lemma is folklore:

Lemma 15

Let $f : \mathscr{C} \to \mathscr{D}$ and $j : \mathscr{C} \to \mathscr{E}$ be two functors. The representative of the distributor $f_* \circ j^*$ (if it exists) is the left Kan extension of f along j.

Proof: Let $g : \mathscr{E} \to \mathscr{D}$. We will note $\mathrm{Lan}_j f$ the representative of $f_* \circ j^*$.

$$\begin{aligned} \mathbf{Cat}(\mathrm{Lan}_j f, g) &\simeq \mathbf{Dist}(f_* \circ j^*, g_*) \\ &\simeq \mathbf{Dist}(f_*, g_* \circ j_*) \\ &\simeq \mathbf{Dist}(f_*, (g \circ j)_*) \\ &\simeq \mathbf{Cat}(f, g \circ j) \end{aligned}$$

■

Proposition 8

Let $f : \mathscr{C} \to \mathscr{D}$ and $j : \mathscr{C} \to \mathscr{E}$ be two functors. If it exists, the coend

$$\int^{c \in \mathscr{C}} \mathscr{E}(j(c), —) \otimes f(c)$$

is the left Kan extension of f along j. Dually,

$$\int_{c:\mathscr{C}} \mathscr{E}(—, j(c)) \circ f(c)$$

is the right Kan extension of f along j.

Proof: Let $g : \mathscr{E} \to \mathscr{D}$ be any functor.

$$\begin{aligned}
&\mathbf{Dist}\left(f_* \circ j^*, g_*\right)\\
\simeq&\mathbf{Dist}\left((d,e) \mapsto \int^c \mathscr{E}(jc,e) \times \mathscr{D}(d,fc), (d,e) \mapsto \mathscr{D}(d,ge)\right)\\
\simeq&\int_e \widehat{\mathscr{D}}\left(d \mapsto \int^c \mathscr{E}(jc,e) \times \mathscr{D}(d,fc), d \mapsto \mathscr{D}(d,ge)\right), \text{by MacLane's parameter's theorem}\\
\simeq&\int_e\int_c \widehat{\mathscr{D}}\left(d \mapsto \mathscr{E}(jc,e) \times \mathscr{D}(d,fc), d \mapsto \mathscr{D}(d,ge)\right), \text{by continuity of the Hom functor}\\
\simeq&\int_e\int_c \widehat{\mathscr{D}}\left(\mathscr{E}(jc,e) \otimes d \mapsto \mathscr{D}(d,fc), d \mapsto \mathscr{D}(d,ge)\right)\\
\simeq&\int_e\int_c \mathbf{Set}\left(\mathscr{E}(jc,e), \widehat{\mathscr{D}}(d \mapsto \mathscr{D}(d,fc), d \mapsto \mathscr{D}(d,ge))\right)\\
\simeq&\int_e\int_c \mathbf{Set}\left(\mathscr{E}(jc,e), \mathscr{D}(fc,ge))\right), \text{by the Yoneda lemma}\\
\simeq&\int_e\int_c \mathscr{D}(\mathscr{E}(jc,e) \otimes fc, ge)\\
\simeq&\int_c \mathbf{Cat}(e \mapsto \mathscr{E}(jc,e) \otimes fc, g), \text{by Fubini's theorem and MacLane's parameter theorem}\\
\simeq&\mathbf{Cat}\left(\int_c e \mapsto \mathscr{E}(jc,e) \otimes fc, g\right), \text{by continuity of the Hom functor}
\end{aligned}$$

So, the functor

$$\int_c e \mapsto \mathscr{E}(jc,e) \otimes fc$$

is the representative of $f_* \circ j^*$. It is the left Kan extension of f along j. ■

8.2.6 There is a well-known formula for computing Kan extensions in **Cat**.

Definition 68 (*cotensor product of an object by a set*)

Let $\mathscr{C}$ be a (locally small) category. Let A be an object in $\mathscr{C}$ and E a set. The **cotensor product** $\mathrm{E} \circ \mathrm{A}$ of A by E is defined by:

$$\forall \mathrm{B} \in \mathscr{C}, \mathscr{C}(\mathrm{B}, \mathrm{E} \circ \mathrm{A}) \simeq \mathbf{Set}(\mathrm{E}, \mathscr{C}(\mathrm{B},\mathrm{A}))$$

Any locally small category with products is cotensored over **Set** (all of its objects have cotensor products with any set) and the cotensor product is given by:

$$\mathrm{E} \circ \mathrm{A} = \prod_{\mathrm{E}} \mathrm{A}$$

We will write $\langle f_e \rangle_{e \in \mathrm{E}} : \mathrm{B} \to \mathrm{E} \circ \mathrm{A}$ for the infinite pairing of arrows $f_e : \mathrm{B} \to \mathrm{A}$ and $\pi_e : \mathrm{E} \circ \mathrm{A} \to \mathrm{A}$ the projections.

Theorem 18 (*MacLane 1978, X.4, Theorem 1*)

With the notations of Definition 67, whenever the objects exist:

$$\mathrm{Ran}_{\mathrm{K}}\mathrm{F}(d) = \int_{c:\mathscr{C}} \mathscr{D}(d, \mathrm{K}c) \circ \mathrm{F}c.$$

The symmetric monoidal case

8.2.7 Recall from §5.1.1 the notion of discrete fibration and of presheaf, and the following lemma (essentially a reformulation of Theorem 3

Lemma 16

> *A functor* $\mathrm{F} : \mathscr{C} \to \mathscr{B}$ *is a discrete fibration if and only if* $\mathscr{C}$ *is isomorphic to the category of elements* $\mathscr{E}\ell(\varphi)$ *of a presheaf over* $\mathscr{B}$*, and, up to the isomorphism,* F *is the first projection.*

Every functor factors essentially uniquely as a composite of a final functor and a discrete fibration. As such, any functor $\mathscr{B} \to \mathscr{C}$ can be seen as a presheaf over $\mathscr{C}$. This allows to reify diagrams in $\mathscr{C}$ as presheafs over $\mathscr{C}$; which offers a language to formalize the fact that diagrams of a certain shape have colimits.

Consider a class $\mathscr{F}$ of diagram shapes (that is, of small categories), containing $\mathbf{1}$. The category $\mathscr{C}^{\mathscr{F}}$ of the diagrams of shape $\mathscr{F}$ in $\mathscr{C}$ corresponds to a subcategory $\overline{\mathscr{C}}$ of $\widehat{\mathscr{C}}$ by the aforementioned correspondence. As $\mathscr{F}$ contains $\mathbf{1}$, the Yoneda embedding $\mathscr{C} \to \widehat{\mathscr{C}}$ restricts to

$$\begin{aligned} y : \mathscr{C} &\to \overline{\mathscr{C}} \\ c &\mapsto (\mathbf{1} \mapsto c) \end{aligned}$$

We can reason about the properties of this embedding. As a restriction of the Yoneda embedding, it is fully faithful. Moreover, for every category $\mathscr{A}$ and functor $f = \mathscr{A} \to \overline{\mathscr{C}}$, a computation based on the Yoneda lemma shows that

$$\mathbf{Dist}(y^* \circ f_*, y^* \circ g_*) = \mathbf{Cat}(f, g)$$

If it exists, the functor colim : $\overline{\mathscr{C}} \to \mathscr{C}$ that associates to every diagram in $\overline{\mathscr{C}}$ its colimit is the left adjoint of y. So, the fact that the diagrams of a certain shape have a colimit in $\mathscr{C}$ can be formally expressed as the fact that the Yoneda embedding associated to that shape has a left adjoint; the fact that the colimits commute with the tensor product can be expressed as the fact that the left adjoint is symmetric monoidal.

Lemma 17

> *Let* $f : \mathscr{B} \nrightarrow \mathscr{C}$ *be a distributor and* $\overline{\mathscr{C}}$ *be a full subcategory of* $\widehat{\mathscr{C}}$ *(the presheaf category over* $\mathscr{C}$*) containing* $\mathscr{C}$*. Suppose that the Yoneda embedding of* $\mathscr{C}$ *into* $\overline{\mathscr{C}}$ *has a left adjoint*
>
> $$\mathrm{colim} \dashv y : \overline{\mathscr{C}} \to \mathscr{C}$$
>
> *and that* f *factors through* y^* *as*
>
> $$\begin{tikzcd} \mathscr{B} \arrow[r, "\overline{f}_*"] & \overline{\mathscr{C}} \arrow[r, "y^*"] & \mathscr{C} \end{tikzcd}$$
>
> *where* $\overline{f} : \mathscr{B} \to \overline{\mathscr{C}}$ *is a functor. The functor* $\mathrm{colim} \circ \overline{f}$ *is a representative of* f*.*

Proof :

$$\begin{aligned}\mathbf{Dist}(f, g^*) &= \mathbf{Dist}(y^* \circ \overline{f}_*, g_*)\\ &\simeq \mathbf{Dist}(y^* \circ \overline{f}_*, y^* \circ y_* \circ g_*), \text{ as the Yoneda embedding is fully faitfull}\\ &\simeq \mathbf{Cat}(\overline{f}, y \circ g)\\ &\simeq \mathbf{Cat}(\mathrm{colim} \circ \overline{f}, g)\end{aligned}$$

■

Definition 69

Let $T : \mathbf{Cat} \to \mathbf{Cat}$ be the monad associating to any category its free symmetric monoidal category. If A is an algebra over T, we will write $[] : TA \to A$ the structure map.

A *symmetric monoidal distributor* is a distributor F making

$$\begin{tikzcd} TA \arrow[r, "TF"] \arrow[d, "{[-]^*}"] & TB \arrow[d, "{[-]^*}"] \\ A \arrow[r, "F"] & B \end{tikzcd}$$

commute for all $F : A \nrightarrow B$.

Lemma 18

Let $j : \mathscr{A} \to \mathscr{B}$ be a monoidal symmetric functor between two monoidal symmetric categories. j^* is symmetric monoidal if and only if

$$\int^{A:T\mathscr{A}} \mathscr{A}(a, [A]) \times T\mathscr{B}(TjA, B) \to \mathscr{B}(ja, B)$$

is an isomorphism.

Theorem 19 (*(Melliès and Tabareau 2008)*)

Let $\mathscr{A}, \mathscr{B}, \mathscr{C}$ be symmetric monoidal categories.

Suppose that the Yoneda embedding $y : \mathscr{C} \to \overline{\mathscr{C}}$ has a left adjoint colim $\dashv y$ *and that both* colim *and y are symmetric monoidal.*

Let $j : \mathscr{A} \to \mathscr{B}$ be a symmetric monoidal functor such that j^* is a symmetric monoidal distributor, and $f : \mathscr{A} \to \mathscr{C}$ be a symmetric monoidal functor. If the distributor $f_* \circ j^*$ factors through y^* as:

$$\begin{tikzcd} \mathscr{B} \arrow[r, "g_*"] & \overline{\mathscr{C}} \arrow[r, "y^*"] & \mathscr{C} \end{tikzcd}$$

then the forgetful functor

$$U_j : \mathbf{SymMonCat}(\mathscr{B}, \mathscr{C}) \to \mathbf{SymMonCat}(\mathscr{A}, \mathscr{C})$$

has a left adjoint

$$\mathrm{Lan}_j : \mathbf{SymMonCat}(\mathscr{A}, \mathscr{C}) \to \mathbf{SymMonCat}(\mathscr{B}, \mathscr{C})$$

computed with a Kan extension.

Proof: The functor colim $\circ\, g$ is the Kan extension of f along j: by Lemma 17, it is the representative of $f_* \circ j^*$, and by Lemma 15, the representative of $f_* \circ j^*$ is the Kan extension of f along j.

It remains to check that it is symmetric monoidal. As colim is (by hypothesis), it suffices to show that g is. By hypothesis, $f_* \circ j^*$ is symmetric monoidal. As composition with y^* induces a fully faithful functor, g is symmetric monoidal. The same argument gives the functoriality of the construction. ■

The Melliès-Tabareau-Tasson construction

8.2.8 Melliès, Tabareau, and Tasson (2009) showed that may build free commutative comonoids as follows:

- compute the free co-pointed object $A^\bullet$ on A (which is A & 1 if the category has binary products);
- compute the symmetric versions of the tensorial powers of $A^\bullet$, i.e. the following equalizers, where $\mathfrak{S}_n$ is the set of canonical symmetries of $(A^\bullet)^{\otimes n}$:

$$\begin{tikzcd} A^{\leqslant n} \arrow[r] & (A^\bullet)^{\otimes n} \arrow[loop right, "\mathfrak{S}_n"] \end{tikzcd}$$

- compute the following projective limit, where $A^{\leqslant n} \longleftarrow A^{\leqslant n+1}$ is the canonical arrow "throwing away" one component:

$$\begin{tikzcd} 1 & A^{\leqslant 1} \arrow[l] & A^{\leqslant 2} \arrow[l] & \cdots \arrow[l] & A^{\leqslant n} \arrow[l] & \cdots \arrow[l] \\ & & & A^\infty \arrow[ulll] \arrow[ull] \arrow[ul] \arrow[ur] \arrow[urr] & & \end{tikzcd}$$

At this point, for A^∞ to be the commutative comonoid on A it is enough that all relevant limits (the equalizers and the projective limit) commute with the tensor. Although not valid in general, this condition holds in several Lafont categories of very different flavor, such as Conway games and coherence spaces.

The idea of Melliès, Tabareau, and Tasson 2009 was to decompose the Kan extension in two, so that the commutation condition is weaker and satisfied by more Lafont categories. The intermediate step uses a symmetric monoidal theory denoted by $\mathbb{I}$, whose objects are natural numbers (seen as finite ordinals) and morphisms are the injections. Note that $\mathrm{Mod}(\mathbb{I}^{\mathrm{op}}, \mathscr{C})$ is equivalent to the slice category $\mathscr{C} \downarrow \mathbf{1}$. By definition, this is the category of *copointed objects* of $\mathscr{C}$: pairs $(A, w : A \to I)$ (with I the tensor unit, not necessarily terminal), with morphisms $f : (A, w) \to (A', w')$ arrows $f : A \to A'$ such that $w = w' \circ f$[1].

8.2.9 Its models in a SMC $\mathscr{C}$ are pointed objects, i.e. pairs (A, p) consisting of an object A of $\mathscr{C}$ and a morphism $p \in \mathscr{C}(\mathbf{1}, A)$, where I is the monoidal unit (which is not required to be terminal). A morphism of pointed objects $f : (A, p) \to (A', p')$ is an arrow $f : A \to A'$ such that $f \circ p = p'$, so the category of pointed objects of $\mathscr{C}$ and their morphisms is just the slice category $\mathbf{1} \downarrow \mathscr{C}$. We invite

[1] The w stands for weakening.

the reader to check that $\mathrm{Mod}(\mathbb{I},\mathscr{C}) \simeq \mathbf{1} \downarrow \mathscr{C}$. Dually, the models of $\mathbb{I}^{\mathrm{op}}$ are co-pointed objects, that is, objects A endowed with a map $\mathrm{A} \to \mathbf{1}$, and we have $\mathrm{Mod}(\mathbb{I}^{\mathrm{op}},\mathscr{C}) \simeq \mathscr{C} \downarrow \mathbf{1}$.

There are of course strict symmetric monoidal injections $j : \mathbb{B} \to \mathbb{I}^{\mathrm{op}}$ and $j' : \mathbb{I}^{\mathrm{op}} \to \mathbb{F}^{\mathrm{op}}$, such that $j' \circ j = i$. Unsurprisingly, $\mathrm{Ran}_j\mathrm{A}(1)$ is the free copointed object on A, which we denoted by $\mathrm{A}^\bullet$ above. Since Kan extensions compose (assuming they exist), we have $\mathrm{A}^\infty = \mathrm{Ran}_{j'}\mathrm{A}^\bullet(1)$:

$$\begin{tikzcd}
\mathbb{B} \arrow[r, "j"] \arrow[dr, bend right=40, "\mathrm{A}"'] & \mathbb{I}^{\mathrm{op}} \arrow[r, "j'"] \arrow[d, "\mathrm{A}^\bullet"'] \arrow[dl, phantom, "\Leftarrow"] & \mathbb{F}^{\mathrm{op}} \arrow[dl, bend left=40, "\mathrm{A}^\infty"] \arrow[l, phantom, shift right=6, "\Leftarrow"] \\
& \mathscr{C} &
\end{tikzcd}$$

8.2.10 For the second Kan extension to be computed in **SymMonCat** using the **Cat** formula, a milder commutation condition than requiring countable biproducts suffices. It is the commutation condition we mentioned above when we recalled the three-step computation of A^∞ (free copointed object, equalizers, projective limit), which indeed results from specializing the general Kan extension formula.

8.3 The polyadic affine λ-calculus in Kan extensions' language

8.3.1 The bridge between the categorical and the topological approach, that will allow us to bring the infinitary calculus to semantic shores will be built upon a further decomposition of the Kan extension: in the second step, we interpose a 2-sorted theory, denoted by $\mathbb{P}$ (this is why we introduced multi-sorted theories, all theories used so far are 1-sorted):

$$\begin{tikzcd}
\mathbb{B} \arrow[r, "j"] \arrow[drr, bend right=30, "\mathrm{A}"'] & \mathbb{I}^{\mathrm{op}} \arrow[r, "k"] \arrow[dr, bend right=20, "\mathrm{A}^\bullet"'] & \mathbb{P} \arrow[r, "l"] \arrow[d, "(\mathrm{A}^\omega,\mathrm{A}^\bullet)"'] \arrow[dl, phantom, "\Leftarrow"] & \mathbb{F}^{\mathrm{op}} \arrow[dl, bend left=30, "\mathrm{A}^\infty"] \arrow[l, phantom, shift right=6, "\Leftarrow"] \\
& & \mathscr{C} &
\end{tikzcd}$$

We will call the models of $\mathbb{P}$ *partitionoids*. Intuitively, the free partitionoid on A allows to speak of infinite streams on $\mathrm{A}^\bullet$, from which one may extract arbitrary elements and substreams via maps of type $\mathrm{A}^\omega \to (\mathrm{A}^\bullet)^{\otimes m} \otimes (\mathrm{A}^\omega)^{\otimes n}$. Such maps are the key to model the infinitary affine λ-calculus.

Example: This intuition is especially evident in **Rel** (the category of sets and relations), where A^ω is the set of all functions $\mathbf{N} \to \mathrm{A}^\bullet$ which are almost everywhere $*$ (in **Rel**, $\mathrm{A}^\bullet = \mathrm{A} \uplus \{*\}$). The free object

$$\mathrm{A}^\omega := \mathrm{Ran}_k\mathrm{A}^\bullet(1,0)$$

is characterized by maps

$$\mathrm{A}^\omega \to (\mathrm{A}^\bullet)^{\otimes m} \otimes (\mathrm{A}^\omega)^{\otimes n}$$

for each choice of $i_1, \ldots, i_m \in \mathbf{N}$ and injections $\beta_1, \ldots, \beta_n : \mathbf{N} \to \mathbf{N}$. These maps, intuitively, take a stream $\langle u_0, u_1, u_2, \ldots\rangle$ and return

$$u_{i_1} \otimes \cdots \otimes u_{i_m} \otimes \langle u_{\beta_1(0)}, u_{\beta_1(1)}, \ldots\rangle \otimes \cdots \otimes \langle u_{\beta_n(0)}, u_{\beta_n(1)}, \ldots\rangle.$$

They are able to extract and manipulate elements of the stream arbitrarily.

8.3.2 In the end, our alternative formula for computing the free commutative comonoid A^∞ on an object A of a symmetric monoidal category may be summarized as follows:

- from the free copointed object $A^\bullet$ on A, compute the limit of the following projective diagram, which we call A^ω:

$$\begin{tikzcd}
\mathbf{1} & A^\bullet \arrow[l, "\varepsilon_1"'] & (A^\bullet)^{\otimes 2} \arrow[l, "\varepsilon_2"'] & \cdots \arrow[l] & (A^\bullet)^{\otimes n} \arrow[l, "\varepsilon_n"'] & (A^\bullet)^{\otimes n+1} \arrow[l, "\varepsilon_{n+1}"'] & \cdots \arrow[l]
\end{tikzcd}$$

  where each arrow "throws away" the rightmost component;

- under certain conditions, A^ω is a "copointed comagma", i.e. it is endowed with arrows $\delta : A^\omega \to A^\omega \otimes A^\omega$ and $\varepsilon : A^\omega \to I$ satisfying no equation. Then, A^∞ is the following equalizer:

$$\begin{tikzcd}
 & A^\omega & \\
A^\infty \arrow[r] & A^\omega \arrow[u, bend left, "(\varepsilon \otimes \mathrm{id}) \circ \delta"] \arrow[u, bend right, "\mathrm{id}"'] \arrow[r, shift left, "(\mathrm{id} \otimes \delta) \circ \delta"] \arrow[r, shift right, "(\delta \otimes \mathrm{id}) \circ \delta"'] \arrow[d, bend right, "\mathrm{swap} \circ \delta"'] \arrow[d, bend left, "\delta"] & (A^\omega)^{\otimes 3} \\
 & (A^\omega)^{\otimes 2} &
\end{tikzcd}$$

The infinitary affine λ-calculus

8.3.3 In order for the notations to be still manageable, we will not present the infinitary affine λ-calculus as an operad, with a language for the reductions. Nonetheless, the translation to an operadic language is fairly immediate.

We consider three pairwise disjoint, countable sets of *linear*, *affine* and *exponential* variables, ranged over by $k, l, m \ldots$, $a, b, c \ldots$ and $x, y, z \ldots$, respectively. The terms of the infinitary affine λ-calculus belong to the following grammar:

$$\begin{array}{rll}
t, u ::= & l \mid \lambda l.t \mid tu \mid k \otimes l\,[\langle u \rangle := t] \mid t \otimes u & \text{linear} \\
 & \mid a \mid a^\bullet\,[\langle u \rangle := t] \mid \bullet t & \text{affine} \\
 & \mid x_i \mid x^\omega\,[\langle u \rangle := t] \mid \langle u_0, u_1, u_2, \ldots \rangle & \text{exponential}
\end{array}$$

The linear part of the calculus comes from Benton et al. 1993. It is the internal language of symmetric monoidal closed categories. As usual, let constructs are binders. The notation $\langle u_0, u_1, u_2, \ldots \rangle$ stands for an infinite sequence of terms. We use $\mathbf{u}$ to range over such sequences and write $\mathbf{u}(i)$ for u_i. Note that each u_i is inductively smaller than $\mathbf{u}$, so terms are infinite but well-founded. The usual linearity/affinity constraints apply to linear/affine variables, with the additional constraint that if x_i, x_j are distinct occurrences of an exponential variable in a term, then $i \neq j$. Furthermore, the free variables of a term of the form $\mathbf{u}$ (resp. $\bullet t$) must all be exponential (resp. exponential or affine).

The reduction rules are as follows:

$$(\lambda l.t)u \to t[u/l] \qquad k \otimes l\,[\langle u \otimes v\rangle := t] \to t[u/k][v/l]$$
$$a^{\bullet}\,[\langle \bullet u\rangle := t] \to t[u/a] \qquad x^{\omega}\,[\langle \mathbf{u}\rangle := t] \to t[\mathbf{u}(i)/x_i]$$

In the exponential rule, i ranges over $\mathbf{N}$, so there may be infinitely many substitutions to be performed. There are also the usual commutative conversions involving let binders, which we omit for brevity. The reduction is confluent, as the rules never duplicate any subterm.

The results of Mazza 2012 are formulated in an infinitary calculus with exponential variables only, whose terms and reduction are defined as follows:

$$t, u ::= x_i \mid \lambda x.t \mid t\langle u_0, u_1, u_2, \ldots\rangle, \qquad (\lambda x.t)\mathbf{u} \to t[\mathbf{u}(i)/x_i]$$

(the abstraction binds all occurrences of x). Such a calculus may be embedded in the one introduced above, as follows:

$$x_i^{\circ} := a^{\bullet}\,[\langle x_i\rangle := a]$$
$$(\lambda x.t)^{\circ} := \lambda l.x^{\omega}\,[\langle l\rangle := t^{\circ}]$$
$$(t\langle u_0, u_1, u_2, \ldots\rangle)^{\circ} := t^{\circ}\langle \bullet u_0^{\circ}, \bullet u_1^{\circ}, \bullet u_2^{\circ}, \ldots\rangle$$

and we have $t \to t'$ implies $t^{\circ} \to^{*} t'^{\circ}$, so we do not lose generality. However, the categorical viewpoint adopted in the present paper naturally leads us to consider a simply-typed version of the calculus, given in Figure 8.3. It is for this calculus that our construction provides Although it may appear additive, the treatment of contexts is multiplicative also in the exponential case, as enforced by the condition in the caption of Figure 8.3. The typing system enjoys the subject reduction property, as can be proved by an induction on the depth of the reduced redex.

Denotational semantics

Definition 70 (*reduced fpp, monoidal theory* $\mathbb{P}$)

A **finite partial partition** (**fpp**) is a finite (possibly empty) sequence $(\mathrm{S}_1, \ldots, \mathrm{S}_k)$ of non-empty, pairwise disjoint subsets of $\mathbf{N}$.

Fpp's have an operadic structure: let $\beta := (\mathrm{S}_1, \ldots, \mathrm{S}_k)$, with S_i infinite, and let $\beta' := (\mathrm{S}'_1, \ldots, \mathrm{S}'_{k'})$; we define

$$\beta' \circ_i \beta := (\mathrm{S}_1, \ldots, \mathrm{S}_{i-1}, \mathrm{T}_1, \ldots, \mathrm{T}_{k'}, \mathrm{S}_{i+1}, \ldots, \mathrm{S}_k),$$

where each T_j is obtained as follows: let $n_0 < n_1 < n_2 < \cdots$ be the elements of S_i in increasing order; then, $\mathrm{T}_j := \{n_m \mid m \in \mathrm{S}'_j\}$.

We will only consider **reduced** fpp's, in which each S_i is either a singleton or infinite. We will use the notation $(\mathrm{S}_1, \ldots, \mathrm{S}_m; \mathrm{T}_1, \ldots, \mathrm{T}_n)$ to indicate that the S_i are singletons and the T_j are infinite, and we will say that such an fpp has size $m + n$. Note that the composition of reduced fpp's is reduced. The set of all reduced fpp's will be denoted by $\mathscr{P}$.

Reduced fpp's induce a 2-sorted monoidal theory $\mathbb{P}$, as follows: each $\beta \in \mathscr{P}$ of size $m + n$ induces an arrow $\beta : (0,1) \to (m,n)$ of $\mathbb{P}$. There is also an arrow $w : (1,0) \to (0,0)$ to account for partiality. Composition is defined as above.

Example: Let $\beta := (\mathrm{E}, \mathrm{O})$, where E and O are the even and odd integers, and let $\beta' := (\{0\}, \mathbf{N} \setminus \{0\})$ (these are actually total partitions). Then $\beta' \circ_1 \beta = (\{0\}, \mathrm{E} \setminus \{0\}, \mathrm{O})$, whereas $\beta \circ_2 \beta' = (\{0\}, \mathrm{O}, \mathrm{E} \setminus \{0\})$.

Definition 71 (*partitionoid*)

A **partitionoid** in a symmetric monoidal category $\mathscr{C}$ is a strict symmetric monoidal functor

$$\mathrm{G} : \mathbb{P} \to \mathscr{C}.$$

Spelled out, it is a tuple $(\mathrm{G}_0, \mathrm{G}_1, w, (r_\beta)_{\beta \in \mathscr{P}})$ with (G_0, w) a copointed object and $r_\beta : \mathrm{G}_1 \to \mathrm{G}_0^{\otimes m} \otimes \mathrm{G}_1^{\otimes n}$ whenever β is of size $m + n$, such that the composition of compatible w and r_β satisfies the equations induced by $\mathbb{P}$.

A morphism of partitionoid s $\mathrm{G} \to \mathrm{G}'$ is a pair of arrows $f_0 : \mathrm{G}_0 \to \mathrm{G}_0'$, $f_1 : \mathrm{G}_1 \to \mathrm{G}_1'$ such that f_0 is a morphism of copointed objects and $r'_\beta \circ f_1 = (f_0^{\otimes m} \otimes f_1^{\otimes n}) \circ r_\beta$ for all $\beta \in \mathscr{P}$ of size $m + n$.

We say that F is the **free partitionoid on** A if it is endowed with an arrow $e : \mathrm{F}_0 \to \mathrm{A}$ such that, for every partitionoid G with an arrow $f : \mathrm{G}_0 \to \mathrm{A}$, there exists a unique morphism of partitionoid s $(u_0, u_1) : \mathrm{G} \to \mathrm{F}$ such that $f = e \circ u$.

Example: For any set X, $(\mathrm{X}, \mathrm{X}^{\mathbf{N}}, !_{\mathrm{X}}, (r_\beta)_{\beta \in \mathscr{P}})$ is a partitionoid in **Set**, where $!_{\mathrm{X}}$ is the terminal arrow $\mathrm{X} \to 1$ and, if

$$\beta = (\{i_1\}, \dots, \{i_m\}; \{j^1_1 < j^1_2 < \cdots\}, \dots, \{j^n_1 < j^n_2 < \cdots\})$$

and $f : \mathbf{N} \to \mathrm{X}$,

$$r_\beta(f) := (f(i_1), \dots, f(i_m), k \mapsto f(j^1_k), \dots, k \mapsto f(j^n_k)) \in \mathrm{X}^m \times (\mathrm{X}^{\mathbf{N}})^n.$$

Lemma 19

If $(\mathrm{F}_0, \mathrm{F}_1)$ *is the free partitionoid on* A, *then* $\mathrm{F}_0 = \mathrm{A}^\bullet$, *the free co-pointed object on* A.

Proof: This follows from observing that $(\mathrm{A}^\bullet, \mathrm{F}_1)$ is also a partitionoid on A. ■

8.3.4 We are ready to define the categories that model the infinitary affine calculus.

Definition 72 (*infinitary affine category*)

Let A be an object in a symmetric monoidal category. We denote by $\dagger_{\mathrm{A}}$ the following diagram:

$$\begin{tikzcd}
\mathbf{1} & \mathrm{A}^\bullet \arrow[l, "\varepsilon_1"'] & (\mathrm{A}^\bullet)^{\otimes 2} \arrow[l, "\varepsilon_2"'] & \cdots \arrow[l] & (\mathrm{A}^\bullet)^{\otimes n} \arrow[l, "\varepsilon_n"'] & (\mathrm{A}^\bullet)^{\otimes n+1} \arrow[l, "\varepsilon_{n+1}"'] & \cdots \arrow[l]
\end{tikzcd}$$

where $\varepsilon_1 = \varepsilon$ is the copoint of $\mathrm{A}^\bullet$ and $\varepsilon_{n+1} := (\mathrm{id})^{\otimes n} \otimes \varepsilon$, the arrow erasing the rightmost component.

If X is another object, we denote by $\mathrm{X} \otimes \dagger_{\mathrm{A}}$ the above diagram in which each $(\mathrm{A}^\bullet)^{\otimes n}$ and ε_n are replaced by $\mathrm{X} \otimes (\mathrm{A}^\bullet)^{\otimes n}$ and $\mathrm{id}_{\mathrm{X}} \otimes \varepsilon_n$, respectively.

We set $\mathrm{A}^\omega := \lim \dagger_{\mathrm{A}}$ (if it exists).

An *infinitary affine category* is a symmetric monoidal closed category such that, for all A, the free partitionoid on A exists and is $(\mathrm{A}^\bullet, \mathrm{A}^\omega)$.

Several well-known categories are examples of affine infinitary categories: sets and relations, coherence spaces and linear maps, Conway games. Finiteness spaces are a non-example. We give the relational example here, which is a bit degenerate but easy to describe and grasp.

Example : The category **Rel** has sets as objects and relations as morphisms. It is symmetric monoidal closed: the Cartesian product (which, unlike in **Set**, is not a categorical product in **Rel**!) acts both as $\otimes$ (with unit the singleton $\{*\}$) and $\multimap$. Let A be a set and let us assume that $* \notin \mathrm{A}$. The free copointed object on A is (up to iso) $\mathrm{A} \cup \{*\}$, with copoint the relation $\{(*, *)\}$. The F_1 part of the free partitionoid on A in **Rel** is (up to iso) the set of all functions $\mathbf{N} \to \mathrm{A}^\bullet$ which are almost everywhere $*$. Given a reduced fpp $\beta := (\{i_1\}, \dots, \{i_m\}; \{j_0^1 < j_1^1 < \dots\}, \dots, \{j_0^n < j_1^n < \dots\})$, the corresponding morphism of type $\mathrm{A}^\omega \to (\mathrm{A}^\bullet)^{\otimes m} \otimes (\mathrm{A}^\omega)^{\otimes n}$ is

$$r_\beta := \{(\mathbf{a}, (a_{i_1}, \dots, a_{i_m}, \langle a_{j_0^1}, a_{j_1^1}, \dots\rangle, \dots, \langle a_{j_0^n}, a_{j_1^n}, \dots\rangle)) \mid \mathbf{a} \in \mathrm{A}^\omega\},$$

where we wrote $\langle a_0, a_1, a_2, \dots\rangle$ for the function $\mathbf{a} : \mathbf{N} \to \mathrm{A}^\bullet, i \mapsto a_i$. This example is especially instructive because one sees very concretely how the infinite sequences in the calculus are modelled in the semantics.

Theorem 20

An infinitary affine category is a denotational model of the infinitary affine λ-calculus.

Proof : The interpretation of types is parametric in an assignment of an object to the base type X, and it is straightforward (notations are identical). In fact, we will confuse types and the objects interpreting them.

Let now $\Gamma; \Delta; \Sigma \vdash t : \mathrm{A}$ be a typing judgment. The type of the corresponding morphism will be of the form $\mathrm{C}_1 \otimes \cdots \otimes \mathrm{C}_n \longrightarrow \mathrm{A}$, where the C_i come from the context and are defined as follows. If it comes from $l : \mathrm{C} \in \Sigma$ (resp. $a : \mathrm{C} \in \Delta$), then $\mathrm{C}_i := \mathrm{C}$ (resp. $\mathrm{C}_i := \mathrm{C}^\bullet$). If it comes from $x : \mathrm{C} \in \Gamma$, then $\mathrm{C}_i := \mathrm{C}^\omega$ if x appears infinitely often in t, otherwise, if it appears k times, $\mathrm{C}_i := (\mathrm{C}^\bullet)^{\otimes k}$.

The morphism interpreting a type derivation of $\Gamma; \Delta; \Sigma \vdash t : \mathrm{A}$ is defined as customary by induction on the last typing rule. The lin-ax rule and all the rules concerning $\otimes$ and $\multimap$ are modeled in the standard way, using the symmetric monoidal closed structure. The only delicate point is modeling the seemingly additive behavior of the exponential context Γ in the binary rules (the same consideration will hold for the elimination rules of $\bullet$ and ω as well). Let us treat for instance the $\otimes$I rule, and let us assume for simplicity that $\Gamma = x : \mathrm{C}, y : \mathrm{D}, z : \mathrm{E}$, with x (resp. z) appearing infinitely often (resp. m and n times) in t and u, whereas y appears infinitely often in t but only k times in u. Let us also disregard the affine and linear contexts, which are unproblematic. The interpretation of the two derivations gives us two morphisms

$$[t] : \mathrm{C}^\omega \otimes \mathrm{D}^\omega \otimes (\mathrm{E}^\bullet)^{\otimes m} \longrightarrow \mathrm{A}, \qquad [u] : \mathrm{C}^\omega \otimes (\mathrm{D}^\bullet)^{\otimes k} \otimes (\mathrm{E}^\bullet)^{\otimes n} \longrightarrow \mathrm{B}.$$

Now, we seek a morphism of type $\mathrm{C}^\omega \otimes \mathrm{D}^\omega \otimes (\mathrm{E}^\bullet)^{\otimes(m+n)} \longrightarrow \mathrm{A} \otimes \mathrm{B}$, because x and y appear infinitely often in $t \otimes u$, whereas z appears $m + n$ times. This is obtained by precomposing $[t] \otimes [u]$ with the morphisms $r_\beta : \mathrm{C}^\omega \to \mathrm{C}^\omega \otimes \mathrm{C}^\omega$ and $r_{\beta'} : \mathrm{D}^\omega \to (\mathrm{D}^\bullet)^{\otimes k} \otimes \mathrm{D}^\omega$ associated with the fpp's $\beta = (; \mathrm{T}_t, \mathrm{T}_u)$ such that T_t (resp. T_u) contains all i such that x_i is free in t (resp. in u), and $\beta' = (\mathrm{S}'_u; \mathrm{T}'_t)$ is defined in a similar way with the variable y.

The weakening on exponential and affine variables in all axiom rules is modeled by the canonical morphisms $\mathrm{A}^\bullet \to \mathbf{1}$ and $\mathrm{A}^\omega \to \mathbf{1}$. For the rules aff-ax and exp-ax, we use the canonical morphism $\mathrm{A}^\bullet \to \mathrm{A}$ and the identity on $\mathrm{A}^\bullet$, respectively.

The $\bullet$I rule is modeled by observing that objects of the form $\Gamma^\omega \otimes \Delta^\bullet$ are copointed (from tensoring their copoints), so from an arrow $\Gamma^\omega \otimes \Delta^\bullet \longrightarrow \mathrm{A}$ we obtain a unique arrow $\Gamma^\omega \otimes \Delta^\bullet \longrightarrow \mathrm{A}^\bullet$ by universality of $\mathrm{A}^\bullet$. The $\bullet$E rule is just composition.

For what concerns the ωI rule, let us assume for simplicity that $\Gamma = x : \mathrm{C}$. This defines a sequence of objects $(\mathrm{C}_i)_{i\in\mathbf{N}}$ such that C_i is either C^ω or $(\mathrm{C}^\bullet)^{\otimes k_i}$ according to whether x appears in $\mathbf{u}(i)$ infinitely often or k_i many times. Let now $\mathrm{S}_i := \{j \in \mathbf{N} \mid x_j \text{ is free in } \mathbf{u}(i)\}$, define the fpp $\beta_i = (\mathrm{S}_0, \dots, \mathrm{S}_i)$ and let

$$\varepsilon'_i := (\mathrm{id})^{\otimes i} \otimes w_i : \mathrm{C}_0 \otimes \cdots \otimes \mathrm{C}_{i-1} \otimes \mathrm{C}_i \longrightarrow \mathrm{C}_0 \otimes \cdots \otimes \mathrm{C}_{i-1},$$

where $w_i : \mathrm{C}_i \to \mathbf{1}$ is equal to $r_\varnothing$ if $\mathrm{C}_i = \mathrm{C}^\omega$ (with $\varnothing$ the empty fpp) or it is equal to $\varepsilon^{\otimes k_i}$ if $\mathrm{C}_i = (\mathrm{C}^\bullet)^{\otimes k_i}$. Let $\widehat{\beta}_i$ be the reduced fpp obtained from β_i by "splitting" its finite sets into singletons. If we set $\theta_i := r_{\widehat{\beta}_i}$, we have that for all $i \in \mathbf{N}$, $\varepsilon'_i \circ \theta_{i+1} = \theta_i$. Let now f_i be the interpretations of the derivations of $x : \mathrm{C};; \vdash \mathbf{u}(i) : \mathrm{A}^\bullet$ and consider the diagram

$$\begin{tikzcd}
 & & \mathrm{C}^\omega \arrow[dll, "\theta_0"'] \arrow[dl, "\theta_1"] \arrow[d, "\theta_2"] \arrow[dr, "\theta_3"'] \arrow[drr, "\theta_n"] & & \\
\mathbf{1} \arrow[d, "\mathrm{id}"] & \mathrm{C}_0 \arrow[l, "\varepsilon'_0"] \arrow[d, "f_0"] & \mathrm{C}_0 \otimes \mathrm{C}_1 \arrow[l, "\varepsilon'_1"] \arrow[d, "f_0 \otimes f_1"] & \mathrm{C}_0 \otimes \mathrm{C}_1 \otimes \mathrm{C}_2 \arrow[l, "\varepsilon'_2"] \arrow[d, "f_0 \otimes f_1 \otimes f_2"] & \cdots \arrow[l, "\varepsilon'_3"] \\
\mathbf{1} & \mathrm{A}^\bullet \arrow[l, "\varepsilon_1"'] & (\mathrm{A}^\bullet)^{\otimes 2} \arrow[l, "\varepsilon_2"'] & (\mathrm{A}^\bullet)^{\otimes 3} \arrow[l, "\varepsilon_3"'] & \cdots \arrow[l, "\varepsilon_4"']
\end{tikzcd}$$

We showed above that all the upper triangles commute. It is easy to check that the bottom squares commute too, making $(\mathrm{C}^\omega, ((f_0 \otimes \cdots \otimes f_{i-1}) \circ \theta_i)_{i\in\mathbf{N}})$ a cone for $\dagger_\mathrm{A}$. Since $\mathrm{A}^\omega = \lim \dagger_\mathrm{A}$, this gives us a unique arrow $f : \mathrm{C}^\omega \to \mathrm{A}^\omega$, which we take as the interpretation of the derivation. The ωE rule is just composition, modulo the interposition of the canonical arrow $\mathrm{A}^\omega \to (\mathrm{A}^\bullet)^{\otimes k}$ in case x appears k times in t.

It remains to check that the above interpretation is stable under reduction, which may be done via elementary calculations. ■

From Infinitary Affine Terms to Linear Logic

8.3.5 In (Mazza 2012), it was shown that usual λ-terms may be recovered as *uniform* infinitary affine terms. The categorical version of this result is that, in certain conditions, a model of the infinitary affine λ-calculus is also a model of linear logic.

Theorem 21

Let $\mathscr{C}$ be an infinitary affine category. If, for every objects X and A in $\mathscr{C}$, the canonical morphism

$$\mathrm{X} \otimes \int_{(n,m)\in\mathbb{P}} (\mathrm{A}^\omega)^{\otimes n} \otimes (\mathrm{A}^\bullet)^{\otimes m} \longrightarrow \int_{(n,m)\in\mathbb{P}} \mathrm{X} \otimes (\mathrm{A}^\omega)^{\otimes n} \otimes (\mathrm{A}^\bullet)^{\otimes m}$$

is an isomorphism, then $\mathscr{C}$ is a Lafont category. Moreover, the free commutative comonoid A^∞ on A may be computed as the equalizer of the diagram:

$$\begin{tikzcd}
 & \mathrm{A}^{\omega} & \\
\mathrm{A}^{\infty} \arrow[r] & \mathrm{A}^{\omega} \arrow[u, bend left, "(\varepsilon \otimes \mathrm{id}) \circ \delta"] \arrow[u, bend right, "\mathrm{id}"'] \arrow[r, bend left, "(\mathrm{id} \otimes \delta) \circ \delta"] \arrow[r, bend right, "(\delta \otimes \mathrm{id}) \circ \delta"'] \arrow[d, bend right, "\mathrm{swap} \circ \delta"'] \arrow[d, bend left, "\delta"] & (\mathrm{A}^{\omega})^{\otimes 3} \\
 & (\mathrm{A}^{\omega})^{\otimes 2} &
\end{tikzcd}$$

where $\delta : \mathrm{A}^{\omega} \to \mathrm{A}^{\omega} \otimes \mathrm{A}^{\omega}$ *and* $\varepsilon : \mathrm{A}^{\omega} \to \mathbf{1}$ *are the morphisms induced by the fpp* $(;\mathrm{E},\mathrm{O})$ *(even and odd numbers) and the empty fpp, respectively, and* $\mathrm{swap} : \mathrm{A}^{\omega} \otimes \mathrm{A}^{\omega} \to \mathrm{A}^{\omega} \otimes \mathrm{A}^{\omega}$ *is the symmetry of* $\mathscr{C}$*.*

Proof: Let $l : \mathbb{P} \to \mathbb{F}^{\mathrm{op}}$ be the strict monoidal functor mapping $(m,n) \mapsto m+n$ and collapsing every arrow $(0,1) \to (m,n)$ to the unique morphism $1 \to m+n$ in $\mathbb{F}^{\mathrm{op}}$. By composing Kan extensions, we know that $\mathrm{A}^{\infty} = \mathrm{Ran}_l(\mathrm{A}^{\bullet}, \mathrm{A}^{\omega})(1)$. Remark that $\mathbb{F}^{\mathrm{op}}(1,p)$ is a singleton for all $p \in \mathbf{N}$, so the hypothesis is exactly what allows to apply Theorem 19, giving us

$$\mathrm{A}^{\infty} = \int_{(m,n)\in\mathbb{P}} (\mathrm{A}^{\omega})^{\otimes n} \otimes (\mathrm{A}^{\bullet})^{\otimes m}.$$

Now, $\int_{(m,n)\in\mathbb{P}} (\mathrm{A}^{\omega})^{\otimes n} \otimes (\mathrm{A}^{\bullet})^{\otimes m}$ is the universal object making

$$\begin{tikzcd}
 & \int_{(m,n)\in\mathbb{P}} (\mathrm{A}^{\omega})^{\otimes n} \otimes (\mathrm{A}^{\bullet})^{\otimes m} \arrow[dl, bend right, "\kappa_{n,m}"] \arrow[dr, bend left, "\kappa_{n',m'}"] & \\
(\mathrm{A}^{\bullet})^{\otimes n} \otimes (\mathrm{A}^{\omega})^{\otimes m} \arrow[rr] & & (\mathrm{A}^{\bullet})^{\otimes n'} \otimes (\mathrm{A}^{\omega})^{\otimes m'}
\end{tikzcd}$$

commute. We are going to show that $\int_{(m,n)\in\mathbb{P}} (\mathrm{A}^{\omega})^{\otimes n} \otimes (\mathrm{A}^{\bullet})^{\otimes m}$ is a cone for the diagram of the Theorem. We will only show that

$$\begin{tikzcd}
 & \int_{(m,n)\in\mathbb{P}} (\mathrm{A}^{\omega})^{\otimes n} \otimes (\mathrm{A}^{\bullet})^{\otimes m} \arrow[dl, bend right, "\delta \circ \kappa_{0,1}"] \arrow[dr, bend left, "\mathrm{swap} \circ \delta \circ \kappa_{0,1}"] & \\
(\mathrm{A}^{\omega})^{\otimes 2} \arrow[rr] & & (\mathrm{A}^{\omega})^{\otimes 2}
\end{tikzcd}$$

commutes. The family $(\iota_n \otimes \iota_m \circ \delta \circ \kappa_{0,1})_{n,m}$ is a cone for $\dagger_{\mathrm{A}}^{\otimes 2}$. Moreover, the $\theta_{n,m} \circ \delta$ are defined in terms of the operations of $\mathbb{P}$, they actually are the canonical maps, and

$$\forall n,m, \iota_n \otimes \iota_m \circ \delta \circ \kappa_{0,1} = \kappa_{0,n+m}$$

The exact same reasoning gives:

$$\forall n, m, \iota_n \otimes \iota_m \circ \mathrm{swap} \circ \delta \circ \kappa_{0,1} = \kappa_{0,n+m}$$

But $(\kappa_{0,n+m})_{n,m}$ factors uniquely through $(\mathrm{A}^{\omega})^{\otimes 2}$ (the limit of $\dagger_{\mathrm{A}}^{\otimes 2}$) and as such,

$$\forall n, m, \delta \circ \kappa_{0,1} = \mathrm{swap} \circ \delta \circ \kappa_{0,1}$$

which is what we wanted. So $\int_{(m,n)\in\mathbb{P}} (\mathrm{A}^{\omega})^{\otimes n} \otimes (\mathrm{A}^{\bullet})^{\otimes m}$ is a cone for the diagram of the Theorem.

Let us now prove that every cone for this diagram is a cone of the diagrams defining $\int_{(m,n)\in\mathbb{P}} (\mathrm{A}^{\omega})^{\otimes n} \otimes (\mathrm{A}^{\bullet})^{\otimes m}$.

It is easy to verify that any object B making the diagram defining A^{∞} commute is endowed with exactly one map $\mathrm{B} \to (\mathrm{A}^{\omega})^{\otimes n}$ for all $n \in \mathbf{N}$, built from δ and ε which, is moreover, stable under all swaps. In particular, by composing these maps $(\mathrm{B} \to (\mathrm{A}^{\omega})^{\otimes n})_{n\in\mathbf{N}}$ with the arrow $\mathrm{A}^{\omega} \to \mathrm{A}^{\bullet}$, it is clear that there is a unique family of arrows

$$\forall n, m \in \mathbf{N}, \mathrm{B} \to (\mathrm{A}^{\bullet})^{\otimes n} \otimes (\mathrm{A}^{\omega})^{\otimes m}$$

stable under extractions and weakenings. So any cone for the diagram defining A^{ω} is a cone for the diagram defining $\int_{(m,n)\in\mathbb{P}} (\mathrm{A}^{\omega})^{\otimes n} \otimes (\mathrm{A}^{\bullet})^{\otimes m}$ and as such, factorizes through it. So $\int_{(m,n)\in\mathbb{P}} (\mathrm{A}^{\omega})^{\otimes n} \otimes (\mathrm{A}^{\bullet})^{\otimes m}$ is the limit of the diagram of the Theorem, and thus isomorphic to A^{∞}. ■

Intuitively, this construction amounts to collapsing the family of non-associative and non-commutative "contractions" built with δ, ε and swap.

It should be remarked that the particular δ used is not canonical, other morphisms would yield the same result. Indeed, from (Mazza 2012) we know that recovering usual λ-terms from infinitary affine terms is possible using uniformity which, as recalled in the introduction, amounts to identifying

$$\lambda x.\langle x_0, x_1, x_2, \ldots\rangle \approx \lambda x.\langle x_{\beta(0)}, x_{\beta(1)}, x_{\beta(2)}, \ldots\rangle,$$

for every injection $\beta : \mathbf{N} \to \mathbf{N}$. Theorem 21 amounts to defining a congruence on terms verifying

$$\lambda x.\,\langle x_0, x_1, x_2, \ldots\rangle \simeq \lambda x.\,\langle x_0, x_2, x_4, \ldots\rangle$$
$$\lambda x.\,\langle x_0, x_2, x_4, \ldots\rangle \otimes \langle x_1, x_3, x_5, \ldots\rangle \simeq \lambda x.\,\langle x_1, x_3, x_5, \ldots\rangle \otimes \langle x_0, x_2, x_4, \ldots\rangle$$

which is sufficient to recover ≈.

Comparison between the approaches

8.3.6 We saw how the functorial semantic framework provides a bridge between the categorical and topological approaches to expressing the exponential modality of linear logic as a form of limit. This gives a way to construct, under certain hypotheses, denotational models of the infinitary affine λ-calculus. Moreover, it gives us a formula for computing the free exponential which is alternative to that of Melliès et al. Since both formulas apply only under certain conditions, it is natural to ask whether one of them is more general than the other. Although we do not have a general result, we are able to show that, under a mild condition verified in all models of linear logic we are aware of, our construction is applicable in every situation where Melliès et al.'s is.

Indeed, Melliès et al.'s construction amounts to checking that the Kan extension along m (below, left) is a monoidal Kan extension, whereas the one exposed in this article amounts to checking that the two Kan extensions along k, then l are monoidal (below, right):

$$\begin{tikzcd}
 & \mathscr{C} & \\
\mathbb{I}^{\mathrm{op}} \arrow[ur] \arrow[rr, "m"'] & & \mathbb{F}^{\mathrm{op}} \arrow[ul] \arrow[ll, Rightarrow, bend left=20, shorten=25pt, yshift=25pt]
\end{tikzcd}$$

$$\begin{tikzcd}
 & \mathbb{P} \arrow[d] \arrow[ddr, "l"] \arrow[ddl, Rightarrow, bend right, shorten=15pt] & \\
 & \mathscr{C} & \\
\mathbb{I}^{\mathrm{op}} \arrow[uur, "k"] \arrow[ur] & & \mathbb{F}^{\mathrm{op}} \arrow[ul] \arrow[uul, Rightarrow, bend right, shorten=15pt]
\end{tikzcd}$$

As Kan extensions compose, it suffices to know that the Kan extension along m is monoidal, that $m = k \circ l$, and that there exists two monoidal natural transformations inside the two upper triangles that can be composed to the last one to be sure that the Kan extensions along k and along l are monoidal too. We thus get:

Proposition 9

Let $\mathscr{C}$ be a symmetric monoidal category with all free partitionoids. Assume that Melliès et al.'s formula works and that A^{ω} exists. If there exists, for all integers n, m monoidal maps

$$\begin{aligned}
(\mathrm{A}^{\infty})^{\otimes n+m} \to & (\mathrm{A}^{\omega})^{\otimes n} \otimes (\mathrm{A}^{\bullet})^{\otimes m} \\
& (\mathrm{A}^{\omega})^{\otimes n} \otimes (\mathrm{A}^{\bullet})^{\otimes m} \to (\mathrm{A}^{\bullet})^{\otimes n+m}
\end{aligned}$$

that composed together are the $n + m$ tensor of the map $\mathrm{A}^{\infty} \to \mathrm{A}^{\leqslant 1} \to \mathrm{A}^{\bullet}$ then $\mathscr{C}$ is an infinitary affine category and a Lafont category.

Actually, in all models we are aware of, either both formulas work, or neither does. For instance, our construction fails for finiteness spaces (Ehrhard 2005), as does the construction given in (Melliès, Tabareau, and Tasson 2009).

$$\frac{}{\Gamma;\Delta;l : \mathrm{A} \vdash l : \mathrm{A}}\,(\text{lin-ax}) \qquad\qquad \frac{}{\Gamma;\Delta,a : \mathrm{A};\vdash a : \mathrm{A}}\,(\text{aff-ax})$$

$$\frac{i \in \mathbf{N}}{\Gamma, x : \mathrm{A};\Delta;\vdash x_i : \mathrm{A}^\bullet}\,(\text{exp-ax})$$

$$\frac{\Gamma;\Delta;\Sigma,l : \mathrm{A} \vdash t : \mathrm{B}}{\Gamma;\Delta;\Sigma \vdash \lambda l.t : \mathrm{A} \multimap \mathrm{B}}\,(\multimap \mathrm{I})$$

$$\frac{\Gamma;\Delta;\Sigma \vdash t : \mathrm{A} \multimap \mathrm{B} \qquad \Gamma;\Delta';\Sigma' \vdash u : \mathrm{A}}{\Gamma;\Delta,\Delta';\Sigma,\Sigma' \vdash tu : \mathrm{B}}\,(\multimap \mathrm{E})$$

$$\frac{\Gamma;\Delta;\Sigma \vdash t : \mathrm{A} \qquad \Gamma;\Delta';\Sigma' \vdash u : \mathrm{B}}{\Gamma;\Delta,\Delta';\Sigma,\Sigma' \vdash t \otimes u : \mathrm{B}}\,(\otimes\mathrm{I})$$

$$\frac{\Gamma;\Delta;\Sigma \vdash u : \mathrm{A} \otimes \mathrm{B} \qquad \Gamma;\Delta';\Sigma',k : \mathrm{A},l : \mathrm{B} \vdash t : \mathrm{C}}{\Gamma;\Delta,\Delta';\Sigma,\Sigma' \vdash k \otimes l\,[\langle u \rangle := t] : \mathrm{C}}\,(\otimes\mathrm{E})$$

$$\frac{\Gamma;\Sigma;\vdash t : \mathrm{A}}{\Gamma;\Sigma;\vdash \bullet t : \mathrm{A}^\bullet}\,(\bullet\mathrm{I}) \qquad\qquad \frac{\Gamma;\Delta;\Sigma \vdash u : \mathrm{A}^\bullet \qquad \Gamma;\Delta',a : \mathrm{A};\Sigma' \vdash t : \mathrm{C}}{\Gamma;\Delta,\Delta';\Sigma,\Sigma' \vdash a^\bullet\,[\langle u \rangle := t] : \mathrm{C}}\,(\bullet\mathrm{E})$$

$$\frac{\ldots \qquad \Gamma;;\vdash \mathbf{u}(i) : \mathrm{A}^\bullet \qquad \ldots}{\Gamma;;\vdash \mathbf{u} : \mathrm{A}^\omega}\,(\omega\mathrm{I})$$

$$\frac{\Gamma;\Delta;\Sigma \vdash u : \mathrm{A}^\omega \qquad \Gamma,x : \mathrm{A};\Delta';\Sigma' \vdash t : \mathrm{C}}{\Gamma;\Delta,\Delta';\Sigma,\Sigma' \vdash x^\omega\,[\langle u \rangle := t] : \mathrm{C}}\,(\omega\mathrm{E})$$

Figure 8.3: The simply-typed infinitary affine λ-calculus. In every non-unary rule we require that t, u (or, for the ωI rule, $\mathbf{u}(i), \mathbf{u}(j)$ for all $i \neq j \in \mathbf{N}$) contain pairwise disjoint sets of occurrences of the exponential variables in Γ.

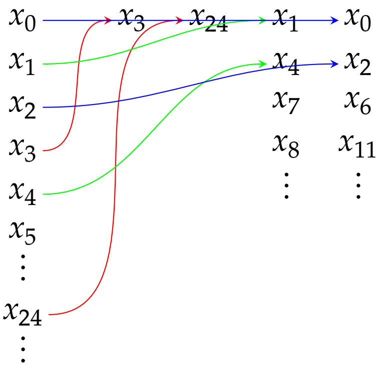
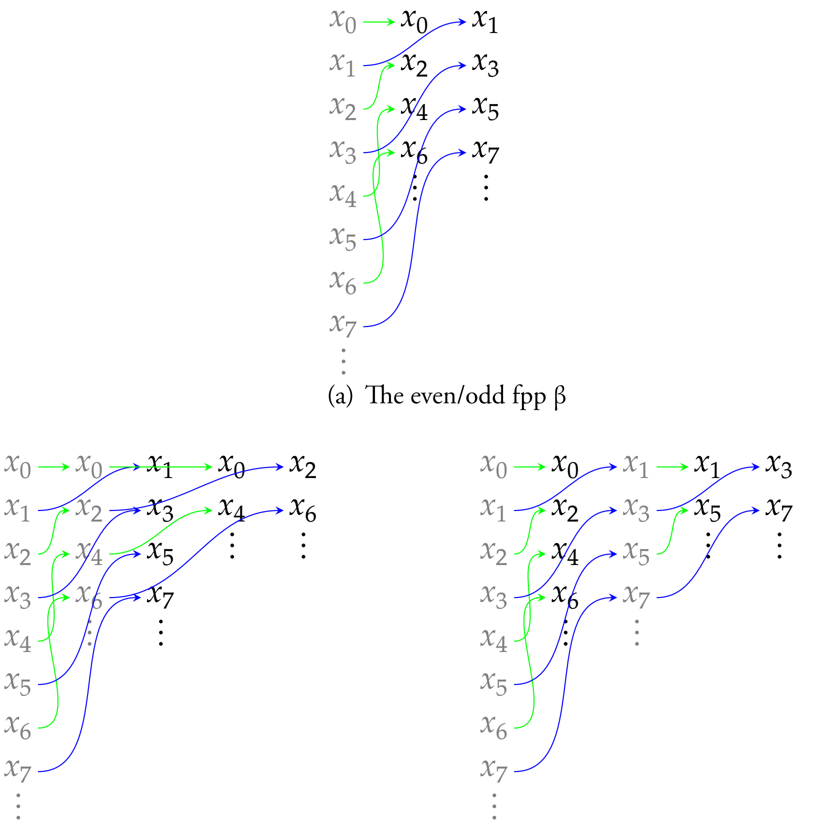


Figure 8.4: A depiction of the fpp $\{\{3\},\{24\},\{1,4,7,8,\dots\},\{0,2,6,11,\dots\}\}$



(a) The even/odd fpp β

(b) $\beta \circ_1 \beta$

(c) $\beta \circ_2 \beta$

Figure 8.5: Operations on the even/odd fpp

Bibliography

Samson Abramsky, Radha Jagadeesan, and Pasquale Malacaria (2000).
"Full Abstraction for PCF".
In: *Inf. Comput.* **163**.2 (2000), pp. 409–470
(Cited p. 136).

Beniamino Accattoli (2012).
"An Abstract Factorization Theorem for Explicit Substitutions".
In: *23rd International Conference on Rewriting Techniques and Applications (RTA'12) , RTA 2012, May 28 - June 2, 2012, Nagoya, Japan.*
2012,
Pp. 6–21
(Cited p. 36).

P. N. Benton, Gavin M. Bierman, Valeria de Paiva, and Martin Hyland (1993).
"A Term Calculus for Intuitionistic Linear Logic".
In: *Typed Lambda Calculi and Applications, International Conference on Typed Lambda Calculi and Applications, TLCA '93, Utrecht, The Netherlands, March 16-18, 1993, Proceedings.*
1993,
Pp. 75–90
(Cited pp. 136, 145).

Dennis V. Borisov and Yuri I. Manin (2008).
"Generalized Operads and Their Inner Cohomomorphisms".
In:
Geometry and Dynamics of Groups and Spaces: In Memory of Alexander Reznikov.
Ed. by Mikhail Kapranov, Yuri Ivanovich Manin, Pieter Moree, Sergiy Kolyada, and Leonid Potyagailo.
Basel: Birkhäuser Basel, 2008,
Pp. 247–308.
doi: `10.1007/978-3-7643-8608-5_4`.
url: `http://dx.doi.org/10.1007/978-3-7643-8608-5_4`
(Cited p. 43).

Pierre Boudes (2009).
"Thick Subtrees, Games and Experiments".

In: *Typed Lambda Calculi and Applications, 9th International Conference, TLCA 2009, Brasilia, Brazil, July 1-3, 2009. Proceedings.*
Ed. by Pierre-Louis Curien.
Lecture Notes in Computer Sciences 5608.
Springer Verlag, 2009,
Pp. 65–79
(Cited p. 73).

Gérard Boudol (1993).
"The Lambda-Calculus with Multiplicities (Abstract)".
In: *Proceedings of CONCUR.*
1993,
Pp. 1–6
(Cited pp. 16, 21).

Daniel de Carvalho (2009).
"Execution Time of lambda-Terms via Denotational Semantics and Intersection Types".
In: *CoRR* **abs/0905.4251** (2009)
(Cited p. 95).

Daniel de Carvalho (2016).
"The Relational Model Is Injective for Multiplicative Exponential Linear Logic".
In: *25th EACSL Annual Conference on Computer Science Logic, CSL 2016, August 29 - September 1, 2016, Marseille, France.*
2016,
41:1–41:19
(Cited p. 133).

Nathalie Chauvac and Marie-Pierre Bès (2011).
"Les réseaux des doctorants : Le cas d'étudiants en Sciences Humaines de l'Université de Toulouse Le Mirail".
In: *7èmes Journées Nationales des Observatoires de l'Enseignement Supérieur.*
Toulouse, France, June 2011.
URL: `https://hal.archives-ouvertes.fr/hal-01519744`
(Cited p. 3).

Giorgio Colli (1998).
Zenone di Elea. Lezioni 1964-1965.
Adelphi, 1998
(Cited p. 8).

Alain Connes (1985).
"Non-commutative differential geometry".
In: *Publications Mathématiques de l'Institut des Hautes Études Scientifiques* **62**.1 (1985), pp. 41–144.
DOI: `10.1007/BF02698807`
(Cited p. 40).

Mario Coppo, Mariangiola Dezani-Ciancaglini, and Betti Venneri (1981).
"Functional Characters of Solvable Terms".
In: *Math. Log. Q.* **27**.2-6 (1981), pp. 45–58

(Cited p. 93).

Vincent Danos and Laurent Regnier (1989).
"The Structure of Multiplicatives".
In: *Archive for Mathematical Logic* **28** (Apr. 1989), pp. 181–203
(Cited p. 104).

Thomas Ehrhard (2005).
"Finiteness spaces".
In: *Mathematical Structures in Computer Science* **15**.4 (2005), pp. 615–646
(Cited pp. 130, 152).

Thomas Ehrhard and Laurent Regnier (2008).
"Uniformity and the Taylor expansion of ordinary lambda-terms".
In: *Theoretical Computer Science* **403**.2-3 (2008), pp. 347–372.
DOI: 10.1016/j.tcs.2008.06.001
(Cited pp. 21, 100, 101).

G. Gentzen (1936).
"Die Widerspruchsfreiheit der reinen Zahlentheorie".
In: *Mathematische Annalen* **112** (1936), pp. 493–565.
URL: http://eudml.org/doc/159839
(Cited p. 8).

E Getzler and M M Kapranov (1995).
"Cyclic operads and cyclic homology".
In: *Geometry, topology & physics* (1995)
(Cited p. 40).

Jean-Yves Girard (1987).
"Linear logic".
In: *Theoretical Computer Science* **50**.1 (1987), pp. 1–101.
DOI: 10.1016/0304-3975(87)90045-4
(Cited pp. 11, 12, 43, 59, 60, 128, 129).

Jean-Yves Girard (2006).
Le Point aveugle.
Hermann, 2006
(Cited p. 12).

W. A. Howard (1980).
"The formulae-as-types notion of construction".
In:
To H. B. Curry: essays on combinatory logic, lambda calculus and formalism.
Ed. by J. P. Seldin and J. R. Hindley.
London-New York: Academic Press, 1980,
Pp. 480–490
(Cited p. 8).

John Martin Elliott Hyland (2017).
"Classical lambda calculus in modern dress".
In: *Mathematical Structures in Computer Science* **27**.5 (2017), pp. 762–781.

DOI: `10.1017/S0960129515000377`
(Cited p. 26).

A. R. T. Kemasang (2009).
"Tea — midwife and nurse to capitalism".
In: *Race & Class* **51**.1 (June 2009), pp. 69–83.
DOI: `10.1177/0306396809106164`
(Cited p. 3).

Delia Kesner and Pierre Vial (2017).
"Types as Resources for Classical Natural Deduction".
In: *2nd International Conference on Formal Structures for Computation and Deduction, FSCD 2017, September 3-9, 2017, Oxford, UK.*
2017,
24:1–24:17
(Cited p. 97).

Assaf J. Kfoury (2000).
"A linearization of the Lambda-calculus and consequences".
In: *J. Log. Comput.* **10**.3 (2000), pp. 411–436
(Cited pp. 21, 100, 136).

Joachim Lambek (1958).
"The Mathematics of Sentence Structure".
In: *The American Mathematical Monthly* **65**.3 (Mar. 1958), pp. 154–170.
DOI: `10.2307/2310058?ref=search-gateway:568c869572671699121919c72c134a62`
(Cited p. 11).

Joachim Lambek (1969).
"Deductive systems and categories II. Standard constructions and closed categories".
In:
Category Theory, Homology Theory and their Applications I: Proceedings of the Conference held at the Seattle Research Center of the Battelle Memorial Institute, June 24–July 19, 1968 Volume One.
Ed. by Peter J. Hilton.
Berlin, Heidelberg: Springer Berlin Heidelberg, 1969,
Pp. 76–122.
DOI: `10.1007/BFb0079385`.
URL: `https://doi.org/10.1007/BFb0079385`
(Cited pp. 23, 27).

Olivier Laurent (2004).
On the denotational semantics of the untyped lambda-mu calculus.
Unpublished note.
2004
(Cited p. 97).

Fosco Loregian (2015).
"This is the (co)end, my only (co)friend".
In: *arXiv.org* (Jan. 2015).
arXiv: `1501.02503v3 [math.CT]`
(Cited p. 83).

Saunders MacLane (1965).
"Categorical algebra".
In: *Bulletin of the American Mathematical Society* **71**.1 (1965), pp. 40–106
(Cited p. 136).

Saunders MacLane (1978).
Categories for the Working Mathematician.
2nd.
Springer-Verlag, 1978
(Cited pp. 137, 140).

J Maraist, M Odersky, D N Turner, and P Wadler (1999).
"Call-by-name, call-by-value, call-by-need and the linear lambda calculus".
In: *Theoretical Computer Science* **228**.1-2 (1999), pp. 175–210.
DOI: 10.1016/S0304-3975(98)00358-2
(Cited p. 60).

J Peter May (1972).
The Geometry of Iterated Loop Spaces.
Vol. 271.
Lecture Notes in Mathematics.
Berlin, Heidelberg: Springer, Nov. 1972.
DOI: 10.1007/BFb0067491
(Cited p. 27).

Damiano Mazza (2012).
"An Infinitary Affine Lambda-Calculus Isomorphic to the Full Lambda-Calculus".
In: *Proceedings of LICS*.
2012,
Pp. 471–480
(Cited pp. 19, 21, 135, 136, 146, 149, 151).

Paul-André Melliès (2004).
Asynchronous Games 1: A Group-Theoretic Formulation of Uniformity.
Technical Report PPS//04//06//n°31.
Preuves, Programmes et Systèmes, 2004
(Cited pp. 21, 136).

Paul-André Melliès (2008).
"Categorical Semantics of Linear Logic".
In: *Interactive Models of Computation and Program Behavior*.
Société Mathématique de France, 2008,
Pp. 1–213
(Cited p. 129).

Paul-André Melliès and Nicolas Tabareau (2008).
"Free models of T-algebraic theories computed as Kan extensions".
Available on the second author's web page.
2008
(Cited pp. 18, 142).

Paul-André Melliès, Nicolas Tabareau, and Christine Tasson (2009).
"An Explicit Formula for the Free Exponential Modality of Linear Logic".
In: *Proceedings of ICALP, Part II.*
2009,
Pp. 247–260
(Cited pp. 18, 135, 143, 152).

Paul-André Melliès and Noam Zeilberger (2015).
"Functors are Type Refinement Systems".
In: *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015.*
New York, New York, USA: ACM Press, 2015,
Pp. 3–16.
DOI: 10.1145/2676726.2676970
(Cited pp. 16, 21, 62).

Susan Niefield (2004).
"Change of Base for Relational Variable Sets".
In: *Theory Appl. Categ.* **12**.7 (2004), pp. 248–261
(Cited p. 63).

Michel Parigot (1992).
"$\lambda\mu$-Calculus: An algorithmic interpretation of classical natural deduction".
In:
Logic Programming and Automated Reasoning: International Conference LPAR '92 St. Petersburg, Russia, July 15–20, 1992 Proceedings.
Ed. by Andrei Voronkov.
Berlin, Heidelberg: Springer Berlin Heidelberg, 1992,
Pp. 190–201.
DOI: 10.1007/BFb0013061
(Cited p. 52).

G D Plotkin (1975).
"Call-by-name, call-by-value and the λ-calculus".
In: *Theoretical Computer Science* **1**.2 (1975), pp. 125–159.
DOI: 10.1016/0304-3975(75)90017-1
(Cited pp. 34, 58).

Dana S. Scott (1970).
"Outline of a mathematical theory of computation".
In: *Technical Monograph PRG-2, Oxford University Computing Laboratory* (1970)
(Cited p. 10).

R A G Seely (1987).
"Modelling Computations : a 2-categorical Framework".
In: *Logic in Computer Science,*
1987,
P. 34
(Cited p. 26).

Christine Tasson (2009).

“Sémantiques et Syntaxes Vectorielles de la Logique Linéaire”.
PhD thesis. Dec. 2009
(Cited p. 100).

Kazushige Terui (2012).
“Semantic Evaluation, Intersection Types and Complexity of Simply Typed Lambda Calculus”.
In: *23rd International Conference on Rewriting Techniques and Applications (RTA’12) , RTA 2012, May 28 - June 2, 2012, Nagoya, Japan.*
2012,
Pp. 323–338
(Cited p. 90).

Takeshi Tsukada, Kazuyuki Asada, and C.-H. Luke Ong (2017).
“Generalised species of rigid resource terms”.
In: *32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2017, Reykjavik, Iceland, June 20-23, 2017.*
2017,
Pp. 1–12
(Cited pp. 16, 21).

R. J. Wood (1982).
“Abstract pro arrows I”.
In: *Cahiers de Topologie et Géométrie Différentielle Catégoriques* **23**.3 (1982), pp. 279–290.
URL: `http://eudml.org/doc/91304`
(Cited p. 85).

Index

List of Figures

Contents