Polyadic approximations, fibrations and intersection types

Damiano Mazza, Luc Pellissier, Pierre Vial

Polyadic Approximations, Fibrations and Intersection Types

DAMIANO MAZZA, CNRS, UMR 7030, LIPN, Université Paris 13, Sorbonne Paris Cité, France
LUC PELLISSIER, Université Paris 13, Sorbonne Paris Cité, LIPN, CNRS, France
PIERRE VIAL, Université Paris 7, Sorbonne Paris Cité, IRIF, CNRS, France

Starting from an exact correspondence between linear approximations and non-idempotent intersection types, we develop a general framework for building systems of intersection types characterizing normalization properties. We show how this construction, which uses in a fundamental way Melliès and Zeilberger's "type systems as functors" viewpoint, allows us to recover equivalent versions of every well known intersection type system (including Coppo and Dezani's original system, as well as its non-idempotent variants independently introduced by Gardner and de Carvalho). We also show how new systems of intersection types may be built almost automatically in this way.

CCS Concepts: • **Theory of computation** → **Type structures**; **Linear logic**;

Additional Key Words and Phrases: Linear logic, intersection types, type systems, multicategories

ACM Reference Format:
Damiano Mazza, Luc Pellissier, and Pierre Vial. 2018. Polyadic Approximations, Fibrations and Intersection Types. *Proc. ACM Program. Lang.* 2, POPL, Article 6 (January 2018), 28 pages. https://doi.org/10.1145/3158094

1 INTRODUCTION

1.1 Polyadic Approximations

In his paper introducing linear logic, Girard [1987] proved an approximation theorem showing that, in the propositional case, proofs of full linear logic may be "approximated" (in an intuitive sense) arbitrarily well in its multiplicative-additive fragment, *i.e.*, the "purely linear" part of linear logic. Girard's approximation theorem is based on the (informal, at this stage) equation

$$!A = \lim_{n\to\infty} \overbrace{(A \,\&\, 1) \otimes \cdots \otimes (A \,\&\, 1)}^{n} \tag{1}$$

where $!A$ is the so-called exponential modality of linear logic, allowing duplication and erasing, and the n-fold tensor of $A \,\&\, 1$ means "A at most n times". These approximations are *affine* precisely because of the use of $A \,\&\, 1$ rather than simply A (which would give us "A exactly n times").

In a sense, the key contribution of this paper is understanding that Equation 1, once generalized to terms and reductions (rather than just formulas), may serve as the basis of a general methodology for constructing intersection types systems, including most of the standard ones.

Authors' addresses: Damiano Mazza, Laboratoire d'Informatique de Paris Nord, CNRS, UMR 7030, LIPN, Université Paris 13, Sorbonne Paris Cité, 99 avenue Jean-Baptiste Clément, Villetaneuse, 94130, France, Damiano.Mazza@lipn.univ-pairs13.fr; Luc Pellissier, Laboratoire d'Informatique de Paris Nord, Université Paris 13, Sorbonne Paris Cité, LIPN, CNRS, 99 avenue Jean-Baptiste Clément, Villetaneuse, 94130, France, Luc.Pellissier@lipn.univ-pairs13.fr; Pierre Vial, Institut de Recherche en Informatique Fondamentale, Université Paris 7, Sorbonne Paris Cité, IRIF, CNRS, 8 place Aurélie Nemours, Paris, 75013, France, pvial@pps.univ-paris-diderot.fr.

2475-1421/2018/1-ART6
https://doi.org/10.1145/3158094

The starting point is a correspondence between linear/affine approximations, as in Girard's theorem, and non-idempotent intersection types, which we now proceed to explain. First, one has to formalize the intuition behind Equation 1. This has already been done in several ways: as a Taylor expansion [Ehrhard and Regnier 2008], as a categorical limit [Melliès et al. 2009] and as a topological limit [Mazza 2012]. The syntactic approaches of Ehrhard and Regnier and of the first author are similar: they both express Girard's approximations directly in a calculus containing terms of the form $\langle t_1, \dots, t_n \rangle$, which we call *polyadic* and which morally correspond to the n-ary tensors of Equation 1. The former approach uses the *resource λ-calculus* of Boudol [1993]. We favor the first author's approach because it is syntactically simpler (there are no formal sums of terms) and more faithful to Equation 1 (the tensor is not syntactically commutative, unlike the resource λ-calculus).

The polyadic affine λ-calculus $\Lambda_{\mathrm{p}}^{\mathrm{aff}}$ [Mazza 2012] is easy to define: its terms and reduction are

$$t, u ::= x \mid \lambda\langle x_1, \dots, x_n \rangle.t \mid t\langle u_1, \dots, u_n \rangle, \qquad (\lambda\langle x_1, \dots, x_m \rangle.t)\langle u_1, \dots, u_n \rangle \quad \to \quad t\{u_i/x_i\},$$

where x ranges over a countably infinite set of affine variables, which must appear at most once, and the reduction rule requires $m \leq n$ (otherwise the term is "stuck"). In case $m < n$, the terms $u_{m+1}, \dots, u_n$ are simply discarded. The polyadic *linear* λ-calculus $\Lambda_{\mathrm{p}}^{\mathrm{lin}}$ is obtained from $\Lambda_{\mathrm{p}}^{\mathrm{aff}}$ by enforcing strict linearity in terms (abstracted variables *must* appear) and by requesting $m = n$ in the reduction rule (so no term may be discarded).

Recasting in order-theoretic language the idea of [Mazza 2012], $\Lambda_{\mathrm{p}}^{\mathrm{aff}}$ may be endowed with a partial order which, in the spirit of Equation 1, is exemplified by the inequality $t\langle u_1, \dots, u_n \rangle \sqsubseteq t\langle u_1, \dots, u_n, u_{n+1} \rangle$. The whole of point of the first author's topological construction (but also of the Taylor expansion of Ehrhard and Regnier [2008]) is that the pure λ-calculus faithfully embeds in the ideal completion of $(\Lambda_{\mathrm{p}}^{\mathrm{aff}}, \sqsubseteq)$: in other words, usual, non-affine λ-terms may be seen as ideals of polyadic affine terms (or, syntactically, as infinite polyadic affine terms). Such an embedding induces an approximation relation $\sqsubset$ on $\Lambda_{\mathrm{p}}^{\mathrm{aff}} \times \Lambda$: we write $t \sqsubset M$ just if $t \in [\![M]\!]$, where $[\![M]\!]$ is the ideal associated with the λ-term M (in Ehrhard and Regnier's approach, $[\![M]\!]$ would be the support of the Taylor expansion of M). In particular, we have the equation

$$!M = \sup_{n \in \mathbb{N},\ t_i \sqsubset M} \langle t_1, \dots, t_n \rangle, \tag{2}$$

where by $!M$ we mean "M as the argument of an application". This is clearly a reformulation of Equation 1 at the level of terms, once we remember that Girard's embedding of intuitionistic logic in linear logic translates $A \Rightarrow B$ as $!A \multimap B$.

The approximation relation may actually be defined directly by induction, using judgments of the form $\Gamma \vdash t \sqsubset M$ where Γ is a list of statements of the form $x_0 \sqsubset x$, with x_0 an affine variable and x a λ-calculus variable, such that $x_0 \sqsubset x, y_0 \sqsubset y \in \Gamma$ implies $x_0 \neq y_0$ (but we may have $x = y$, meaning that both x_0 and y_0 approximate the same variable). The approximation relation is defined as follows:

$$\frac{}{\Gamma, x_0 \sqsubset x \vdash x_0 \sqsubset x}\ \text{var} \qquad \frac{\Gamma, \dots x_i \sqsubset x \dots \vdash t \sqsubset M}{\Gamma \vdash \lambda\langle x_1, \dots, x_n \rangle.t \sqsubset \lambda x.M}\ \text{lam} \qquad \frac{\Gamma \vdash t \sqsubset M \quad \dots\ \Delta_i \vdash u_i \sqsubset N\ \dots}{\Gamma, \Delta_1, \dots, \Delta_n \vdash t\langle u_1, \dots, u_n \rangle \sqsubset MN}\ \text{app}$$

where, in the lam rule, x does not appear in Γ. The direct generalization of Girard's approximation theorem is the continuity of reduction: if $M \to^* N$, then for all $u \sqsubset N$, there exists $t \sqsubset M$ such that $t \to^* u$. In other words, given a computation in the λ-calculus, to compute an approximation of the result, it suffices to have an approximation of the initial term and perform a computation in $\Lambda_{\mathrm{p}}^{\mathrm{aff}}$. In fact, continuity also holds for $\Lambda_{\mathrm{p}}^{\mathrm{lin}}$: if u is linear, then one may take also t to be linear.

$$\frac{}{x : A \vdash x : A}\ \text{var} \qquad \frac{\Gamma \vdash M : B}{\Gamma \vdash \lambda x.M : \top \multimap B}\ \text{lam}_0\ x \notin \Gamma \qquad \frac{\Gamma, x : A_1 \wedge \cdots \wedge A_n \vdash M : B}{\Gamma \vdash \lambda x.M : A_1 \wedge \cdots \wedge A_n \multimap B}\ \text{lam}$$

$$\frac{\Gamma \vdash M : \Theta \multimap B \quad \Delta_1 \vdash N : A_1 \ \ldots\ \Delta_n \vdash N : A_n}{\Gamma \cdot \Delta_1 \cdots \Delta_n \vdash MN : B}\ \text{app} \qquad \frac{\Gamma, x : A_1 \wedge \cdots \wedge A_n \vdash M : C}{\Gamma, x : A_{\sigma(1)} \wedge \cdots \wedge A_{\sigma(n)} \vdash M : C}\ \text{perm}$$

Fig. 1. Gardner-de Carvalho's non-idempotent intersection type system, as presented by [Gardner 1994]. In the app rule, $\Theta = A_1 \wedge \cdots \wedge A_n$.

1.2 Polyadic Simple Types Are Intersection Types

Intersection types, originally introduced by Coppo and Dezani-Ciancaglini [1980] with semantic motivations, are a well-known type-theoretic approach to expressing and capturing dynamic properties of programs. Through the years, the theory of intersection types has ramified along a host of different directions and taken a number of different forms. One of these is the so-called *non-idempotent* variant. Intuitively, intersection is non-idempotent when $A \to A \wedge A$ does not hold. So, for instance, the typing judgment $f : A \to A \to B \vdash \lambda x.fxx : A \to B$ is not derivable with a non-idempotent intersection; instead $f : A \to A \to B \vdash \lambda x.fxx : A \wedge A \to B$ is derivable.

The first (and simplest) example of non-idempotent intersection type system was introduced by Gardner [1994] and, independently and with a slightly different formulation, by de Carvalho [2009]. Types are defined by

$$A ::= \alpha \mid \Theta \multimap A \quad \text{(types)} \qquad\qquad \Theta ::= A_1 \wedge \cdots \wedge A_n \quad \text{(intersections)}.$$

Nullary intersections are authorized and denoted by $\top$. Intersections should be seen as elements of the free monoid over types: $\Theta \wedge \Theta'$ is defined by concatenation and $\top$ is the neutral element. Typing judgments are of the form $\Gamma \vdash M : A$, where M is a pure λ-term, A a type and Γ a list (permutable at will) of type declarations of the form $x : \Theta$. Gardner's version of the typing rules is recalled in Fig. 1. In rule app, the context $\Gamma \cdot \Delta_1 \cdots \Delta_n$ is obtained by concatenating the type declarations in Γ, Δ_i, assuming that each variable appearing in any of Γ, Δ_i actually appears in all of them, by adding the fictitious declaration $x : \top$ when it does not.

The behavior of non-idempotent intersection of course hints to a connection with linear logic. Indeed, after Gardner, non-idempotent intersection types were considered again by Carlier et al. [2004] and their relationship with linear logic, and comparison with the idempotent case, expounded by Neergaard and Mairson [2004]. In his independent work, de Carvalho unveiled the strong link between non-idempotent intersection types and the relational semantics of linear logic, a line of work which was subsequently extended by Bernadet and Lengrand [2013]. We will see that there is an even tighter correspondence between intersection types and linear logic (in its polyadic form) which, in fact, goes well beyond the non-idempotent case.

The terms of $\Lambda_{\mathrm{p}}^{\mathrm{lin}}$ are actually proof terms for a fragment of multiplicative intuitionistic linear logic, whose formulas are given by

$$A, B ::= \alpha \mid A_1 \otimes \cdots \otimes A_n \multimap B,$$

where the case $n = 0$ is written $1 \multimap B$. Via Curry-Howard, this logical fragment induces a simple-type discipline on $\Lambda_{\mathrm{p}}^{\mathrm{lin}}$ (where $\vec{A} = A_1 \otimes \ldots \otimes A_n$):

$$\frac{}{x_0 : A \vdash x_0 : A}\ \text{var} \qquad \frac{\Gamma, \ldots x_i : A_i \ldots \vdash t : B}{\Gamma \vdash \lambda\langle x_1, \ldots, x_n\rangle.t : \vec{A} \multimap B}\ \text{lam} \qquad \frac{\Gamma \vdash t : \vec{A} \multimap B \quad \ldots\ \Delta_i \vdash u_i : A_i \ \ldots}{\Gamma, \Delta_1, \ldots, \Delta_n \vdash t\langle u_1, \ldots, u_n\rangle : B}\ \text{app}$$

The typing rules are so similar to the approximation rules that it is tempting to superpose them:

$$\frac{}{x_0 \sqsubset x : A \vdash x_0 \sqsubset x : A}\ \text{var} \qquad \frac{\Gamma, x_1 \sqsubset x : A_1, \ldots, x_n \sqsubset x : A_n \vdash t \sqsubset M : B}{\Gamma \vdash \lambda\langle x_1, \ldots, x_n\rangle.t \sqsubset \lambda x.M : A_1 \otimes \cdots \otimes A_n \multimap B}\ \text{lam}$$

$$\frac{\Gamma \vdash t \sqsubset M : A_1 \otimes \cdots \otimes A_n \multimap B \qquad \Delta_1 \vdash u_1 \sqsubset N : A_1 \quad \ldots \quad \Delta_n \vdash u_n \sqsubset N : A_n}{\Gamma, \Delta_1, \ldots, \Delta_n \vdash t\langle u_1, \ldots, u_n\rangle \sqsubset MN : B}\ \text{app}$$

where, in rule lam, x does not appear in Γ.

If, in the above superposition, we discard the green decorations and the approximation symbols, we obtain a type system for the λ-calculus, in which typing contexts may contain more than one declaration for a variable. If we adopt the following changes of notation

$$\begin{array}{rcll} A_1 \otimes \ldots \otimes A_n & \rightsquigarrow & A_1 \wedge \cdots \wedge A_n & \\ \Gamma, x : A_1, \ldots, x : A_n \vdash M : B & \rightsquigarrow & \Gamma, x : A_1 \wedge \cdots \wedge A_n \vdash M : B & x \notin \Gamma \end{array}$$

we see that this is exactly the system of Fig. 1:

- the rules var and app are identical;
- the lam_0 rule is just the case $n = 0$ of our lam rule;
- the perm rule is given by the (implicit) exchange rule on contexts of polyadic derivations.

We have thus unveiled an exact correspondence between non-idempotent intersection types and simply-typed linear approximations.

A crucial observation now is that approximations may actually be generalized to the point of being completely non-linear, *i.e.*, along with erasing, duplication too may be allowed. There are thus four flavors of polyadic calculi, depending on whether duplication and erasing (corresponding, logically, to contraction and weakening) are switched on or off, and all of them approximate the λ-calculus in a meaningful way (*i.e.*, reduction is continuous). In other words, the intrinsic ability of polyadic terms $\langle t_1, \ldots, t_n\rangle$ to approximate $!M$ (*cf.* Equation 2) is independent from their linearity or affinity. Repeating the above syntactic game with cartesian polyadic approximations (*i.e.*, allowing both weakening and contraction), we obtain exactly the system of Coppo et al. [1981], in its incarnation characterizing solvability/head normalization (because of the unrestricted presence of the type Ω, represented here by the empty tensor 1).

We thus have what seems to be a general correspondence between polyadic simple types and intersection types: *an intersection type derivation for M is the same thing as a simply-typed polyadic approximation of M*. In other words, an intersection type derivation δ is isomorphic to a simple-type derivation for a polyadic term δ^-, and we have

$$\delta :: \Gamma \vdash_{\text{IT}} M : A \qquad \text{iff} \qquad \delta^- \sqsubset M. \tag{3}$$

Such a sharp correspondence deserves to be treated more abstractly. For this purpose, we will be confronted with the question of finding a general but workable definition of "programming language" and "type system". We will adopt a categorical perspective, taking 2-operads to answer the first question and a suitable kind of fibrations to answer the second.

1.3 Type Systems as 2-Operadic Fibrations

Melliès and Zeilberger [2015] recently suggested an amazingly simple definition of type system: it is a functor $p : \mathcal{D} \longrightarrow C$ mapping a category $\mathcal{D}$ of derivations to a monoid C of programs (in the simple case in which programs are untyped, otherwise C is also a category, and we speak of a *type refinement* system). One may think of the elements of C as untyped programs depending on a parameter x (a free variable), with composition $t \circ u$ defined as $t\{u/x\}$, *i.e.*, substitution of u for the parameter x in t. In this picture, the objects of $\mathcal{D}$ are types, and a morphism $f : A \to B$ of $\mathcal{D}$

is a type derivation of the judgment $x : A \vdash p(f) : B$, *i.e.*, the functor p maps derivations to their subject (the program they type). If a program is not in the image of p, it is not typable.

This abstract view of type systems will be the basis of our work, with two minor twists:

(1) following Hyland [2017], we consider symmetric multicategories and operads instead of categories and monoids, because usual calculi and type systems fit very naturally in this framework (a term may have more or less than one free variable);
(2) we work in a **Cat**-enriched setting, because intersection types are all about capturing properties of reduction, which may be conveniently represented by a second categorical dimension (types are objects, terms are morphisms, reductions are 2-morphisms).

In the following, for brevity we call *2-operad* a **Cat**-enriched symmetric multicategory or, in equivalent terminology, a colored **Cat**-operad. The formal definition will be given in Sect. 2.1; for the moment, the unacquainted reader may think of these simply as a means of describing, in categorical language, the syntax and operational semantics (*i.e.*, reduction, or evaluation) of programming languages seen as term calculi.

Although the one provided by Melliès and Zeilberger is a very useful, workable definition of type system, it is a bit too broad for our purposes, because we need to be able to speak of *subject reduction* and *subject expansion* (*i.e.*, the fact that typing is preserved by reduction or expansion). It turns out that these notions, which are of paramount importance in intersection type systems, find a categorical counterpart in *oplifting* and *lifting* properties, respectively. These are at the heart of the idea of *fibration*, a key concept appearing in many different contexts in category theory. In our case, we will introduce in Sect. 2.2 (operadic) *Niefield fibrations*, a declination of the notion of fibration providing us with a robust framework for speaking of subject reduction and expansion.

Therefore, from our point of view, building an intersection type system for the λ-calculus amounts to constructing a Niefield fibration

$$\mathbf{p} : \mathcal{E} \longrightarrow \Lambda,$$

where Λ is the 2-operad presenting the pure λ-calculus. Now, the key property of Niefield fibrations is that the above is essentially the same thing as a (lax) morphism

$$\mathbf{F} : \Lambda \longrightarrow \mathfrak{Rel},$$

where $\mathfrak{Rel}$ is a (weak) 2-operad based on relational distributors (functors $\mathcal{A} \times \mathcal{B}^{\mathrm{op}} \to \mathbf{Rel}$, where $\mathcal{A}, \mathcal{B}$ are arbitrary categories and **Rel** is the category of sets and relations). This equivalence, which we will formally state and prove in Sect. 2.2, is just a suitable adaptation to our framework of the so-called *Grothendieck construction*, a standard categorical construction relating fibrations (of a given kind) with presheaves (of some form). It is thanks to this alternative view of type systems as "2-operadic relational presheaves" that polyadic approximations come back to center stage.

1.4 The Approximation Presheaf

Let us go back a moment to Equation 2. Although morally correct, such an equation cannot be taken literally, because $!M$ is not really a λ-term. This mismatch may be fixed by considering an untyped term calculus for linear logic, *i.e.*, with explicit duplication and erasing. This calculus, which we call $\Lambda_!$ and will introduce formally in Sect. 3.1, contains, in particular, terms of the form $!T$ marking the term T as duplicable and discardable. One may then define a polyadic calculus Λ_{p} approximating $\Lambda_!$ in the same sense as discussed in Sect. 1.1 for $\Lambda_{\mathrm{p}}^{\mathrm{aff}}$ and Λ, and Equation 2 becomes literally true. This, we will see, is not just a matter of taste; it will bring our work to a higher level of generality.

Of course, we may consider the 2-operad **Poly** of Church-style simply-typed polyadic terms, and we have a type system

$$(\cdot)^- : \mathbf{Poly} \longrightarrow \Lambda_{\mathrm{p}}$$

mapping a polyadic simple-type derivation (*i.e.*, a Church-style term) of $\Gamma \vdash t : A$ to t.

Fix now an arbitrary sub-operad $\mathcal{D}$ of **Poly**. We will define the *approximation presheaf* (relative to $\mathcal{D}$) as a lax morphism

$$\mathrm{Apx}[\mathcal{D}] : \Lambda_! \longrightarrow \mathfrak{Rel}.$$

The formal definition will be given in Sect. 4, let us point out the essence:

- simplifying a bit, we may consider $\Lambda_!$ to have only one type $*$; $\mathrm{Apx}[\mathcal{D}](*)$ is then defined to be the set $\mathcal{T}[\mathcal{D}]$ (seen as a discrete category) of objects of $\mathcal{D}$, *i.e.*, the polyadic simple types used by $\mathcal{D}$;
- given a term T of $\Lambda_!$ with n free variables, $\mathrm{Apx}[\mathcal{D}](T)$ has to be a functor $\mathcal{T}[\mathcal{D}]^n \times \mathcal{T}[\mathcal{D}]^{\mathrm{op}} \to \mathbf{Rel}$; since $\mathcal{T}[\mathcal{D}]$ is discrete, this is just a map assigning a set to each $(n+1)$-tuple Γ, A of types of $\mathcal{D}$, which we define as follows:

$$\mathrm{Apx}[\mathcal{D}](T)(\Gamma, A) := \{\delta \in \mathcal{D}(\Gamma; A) \mid \delta^- \sqsubset T\}.$$

Intuitively, in view of applying the Grothendieck construction, $\mathrm{Apx}[\mathcal{D}](T)(\Gamma, A)$ is going to be the set of intersection type derivations of $\Gamma \vdash T : A$. This explains the definition: we are just implementing Equivalence 3 of Sect. 1.2: an intersection type derivation for T must be the same thing as a simply typed polyadic approximation of T. We are parametrizing the construction on an arbitrary suboperad $\mathcal{D}$ of **Poly** because not all simply-typed approximations may be considered suitable in general (for instance, we may not want contraction/idempotency, or we may want to exclude the empty intersection $\langle\rangle$).

1.5 Intersection Type Systems from Polyadic Approximations

We are now ready to formulate the general construction relating polyadic approximations and intersection types. The framework has two parameters:

- a suboperad $\mathcal{D} \hookrightarrow \mathbf{Poly}$ of "valid" derivations;
- a morphism of 2-operads $\mathbf{G} : \mathcal{L} \to \Lambda_!$.

The 2-operad $\mathcal{L}$ must be seen as presenting a(n untyped) programming language which has a semantic-preserving encoding in $\Lambda_!$ (*i.e.*, in intuitionistic linear logic). For instance, $\mathcal{L}$ could be Λ (the pure λ-calculus) and $\mathbf{G}$ Girard's encoding.

In fact, we will see that **Poly** and $\Lambda_!$ are themselves parameters, *i.e.*, they are not the most general 2-operads one may chose for the construction. However, they have the advantage of being easily presentable and are enough to recover intersection types for all major variants of the λ-calculus, so we stick to them in our presentation.

By composing $\mathbf{G}$ with the approximation presheaf, we get a presheaf

$$\mathcal{L} \xrightarrow{\mathbf{G}} \Lambda_! \xrightarrow{\mathrm{Apx}[\mathcal{D}]} \mathfrak{Rel}$$

which, via the Grothendieck construction, induces a type system for $\mathcal{L}$, denoted by

$$\mathbf{p}[\mathcal{D}, \mathbf{G}] : \mathcal{E}[\mathcal{D}, \mathbf{G}] \longrightarrow \mathcal{L}.$$

We say that a term M of $\mathcal{L}$ is $(\mathcal{D}, \mathbf{G})$-*typable* if it is in the image of $\mathbf{p}[\mathcal{D}, \mathbf{G}]$.

Most well known intersection type systems for the λ-calculus (or minor reformulations of them) arise from the above construction, *i.e.*, they are isomorphic to $\mathbf{p}[\mathcal{D}, \mathbf{G}]$ for some choice of $\mathcal{D}$ and $\mathbf{G}$. For instance, $\mathbf{p}[\mathbf{LinPoly}, \mathbf{G}_0]$ (where **LinPoly** are linear derivations and $\mathbf{G}_0$ is Girard's call-by-name encoding) gives rise to Gardner-de Carvalho's system as discussed in Sect. 1.2. Many more examples

will be given in Sect. 4.4. What is more important, however, is that this abstract setting also allows us to prove a quite general theorem through which the usual properties of intersection type systems may be recovered. This will also be done in Sect. 4.

The rest of the paper is devoted to filling in the technical gaps left by this introduction: in Sect. 2 we give some background on 2-operads, exemplify how they may be used to present programming languages and formalize our operadic version of the Grothendieck construction; in Sect. 3, we introduce polyadic calculi and approximations; in Sect. 4, we formally define the approximation presheaf, prove Theorem 4.7 and give (detailed) examples of applications of the construction and of Theorem 4.7; finally, in Sect. 5, we discuss related work and give some concluding remarks.

2 OPERADS AND FIBRATIONS

2.1 Programming Languages as 2-Operads

We give here a brief survey of the categorical definitions used in the paper; see [Curien and Obradovic 2017; Leinster 2004] for more complete definitions. We denote by $\mathbb{B}(n)$ the set of permutations on n elements.

Definition 2.1 (2-operad, bioperad). A (small) *2-operad* $\mathcal{C}$ is given by the following:

- a set $\mathcal{C}_0$ of *objects* (also called *colors*);
- for every objects $C_1,\dots,C_n,A$, a category $\mathcal{C}(C_1,\dots,C_n;A)$ 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 $\Gamma_1,\dots,\Gamma_n$, an *operadic composition* functor

$$\circ_{\Gamma_1,\dots,\Gamma_n;\Delta;A} : \mathcal{C}(\Delta;A)\times\mathcal{C}(\Gamma_1;B_1)\times\cdots\times\mathcal{C}(\Gamma_n;B_n)\longrightarrow\mathcal{C}(\Gamma_1,\dots,\Gamma_n;A);$$

- for each $n\in\mathbb{N}$, $\sigma\in\mathbb{B}(n)$, object A and objects $\Gamma := C_1,\dots,C_n$, a functor

$$\mathsf{exch}_\sigma^{\Gamma;A} : \mathcal{C}(C_1,\dots,C_n;A)\longrightarrow\mathcal{C}(C_{\sigma^{-1}(1)},\dots,C_{\sigma^{-1}(n)};A)$$

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 and compatibility laws with respect to exch. In case such laws hold only up to (coherent) isomorphism, we speak of a *bioperad*.

A 2-operad is called *unital* if, for every object A, there is a multimorphism $\mathrm{id}_A\in\mathcal{C}(A;A)$ behaving as the identity for operadic composition (up to isomorphism in the case of a bioperad).

In the above definition, composition is given "globally"; one may alternatively define a family of "local" composition functors $\circ^i_{\Gamma;\Delta;A} : \mathcal{C}(B_1,\dots,B_n;A)\times\mathcal{C}(\Gamma;B_i)\to(B_1,\dots,B_{i-1},\Gamma,B_{i+1},\dots,B_n;A)$, with an additional commutation law. In the unital case, the two approaches are equivalent.

A 2-operad $\mathcal{C}$ is *monochromatic* if it has only one object, usually denoted by $*$. In that case, one writes $\mathcal{C}(n)$ for $\mathcal{C}(\underbrace{*,\dots,*}_{n};*)$.

A *morphism* of 2-operads $\mathbf{f}:\mathcal{C}\to\mathcal{D}$ is given by:

- a function $\mathbf{f}_0:\mathcal{C}_0\to\mathcal{D}_0$;
- for each object A and sequence of objects Γ of $\mathcal{C}$, functors $\mathbf{f}_{\Gamma;A}:\mathcal{C}(\Gamma;A)\to\mathcal{D}(\mathbf{f}_0\Gamma;\mathbf{f}_0A)$;

such that (whenever it makes sense):

- $\mathbf{f}(\theta\circ(\theta_1,\dots,\theta_n)) = \mathbf{f}\theta\circ(\mathbf{f}\theta_1,\dots,\mathbf{f}\theta_n)$;
- $\mathbf{f}\circ\mathsf{exch}_\sigma^{\Gamma;A} = \mathsf{exch}_\sigma^{\mathbf{f}_0\Gamma;\mathbf{f}_0A}\circ\mathbf{f}$.

(Subscripts are usually omitted). If $\mathcal{C}$ and $\mathcal{D}$ are unital, then $\mathbf{f}$ is *unital* if it further satisfies

- $\mathbf{f}(\mathrm{id}_A) = \mathrm{id}_{\mathbf{f}A}$.

A (unital) *lax morphism* of (unital) bioperads is as above, but in which the equalities are replaced by (coherent) natural transformations in the right-to-left direction, which are not necessarily invertible.

From the point of view of a programming languages theorist, 2-operads are best understood as a means of presenting (typed) term calculi:

- objects are types;
- a multimorphism $t : C_1, \dots, C_n \to A$ is a term in context
$$x_1 : C_1, \dots, x_n : C_n \vdash t : A;$$
operadic composition is substitution, *i.e.*, $t \circ^i u = t\{u/x_i\}$, and the action of the symmetric group corresponds to the possibility of injectively renaming free variables or, equivalently, to the presence of an exchange rule on typing contexts;
- a 2-arrow $t \Rightarrow t'$ is a type-preserving computation starting from t and leading to t' (*e.g.* term-rewriting, cut-elimination, etc.). Computations are required to be compatible with substitution, in the sense that, for any two computations $t \Rightarrow t'$ and $u \Rightarrow u'$, there is a computation $t\{u/x\} \Rightarrow t'\{u'/x\}$ and such a computation is equal to both sides of the diagram
$$\begin{array}{ccc} t\{u/x\} & \Longrightarrow & t'\{u/x\} \\ \Downarrow & & \Downarrow \\ t\{u'/x\} & \Longrightarrow & t'\{u'/x\} \end{array} \tag{4}$$
where the intermediate computations are obtained by considering the identity computations on t and u.

In this perspective, a morphism of 2-operads $\mathbf{f} : \mathcal{C} \to \mathcal{D}$ is just a modular, semantic-preserving encoding/translation/compilation of term languages:

- a type A of $\mathcal{C}$ is encoded by $\mathbf{f}(A)$ in $\mathcal{D}$;
- a term $x_1 : C_1, \dots, x_n : C_n \vdash t : A$ of $\mathcal{C}$ is encoded by $x_1 : \mathbf{f}(C_1), \dots, x_n : \mathbf{f}(C_n) \vdash \mathbf{f}(t) : \mathbf{f}(A)$;
- we have a "substitution lemma" $\mathbf{f}(t\{u/x\}) = \mathbf{f}(t)\{\mathbf{f}(u)/x\}$ (modularity);
- if $t \Rightarrow t'$, then $\mathbf{f}(t) \Rightarrow \mathbf{f}(t')$ (preservation of operational semantics) and modularity holds for computations too.

Following the above intuition, it is natural to consider unital operads: identities are just variables. However, unital morphisms are less anodyne: they force the encoding of a variable to be necessarily a variable. We will also see very natural examples of calculi in which not all types have identities, leading us to our slightly more general definition.[1]

Let us look at a couple of useful examples. The pure λ-calculus naturally induces a unital monochromatic 2-operad: $\Lambda(n)$ is the category whose objects are λ-terms with free variables in $\{x_1, \dots, x_n\}$ and whose morphisms are β-reduction sequences modulo permutation equivalence; operadic composition is substitution. We remind that *permutation equivalence* [Terese 2003] is the equivalence obtained on reduction sequences by equating all diagrams of the form of Diagram 4 above. Plain β-reduction sequences do not yield a 2-operad: substitution fails to be functorial.

A 2-categorical treatment of the λ-calculus was first advocated by Seely [1987], using the language of cartesian closed 2-categories instead of operads. The 2-CCC perspective was subsequently investigated in depth by Hilken [1996] and, in a much broader framework, by Hirschowitz [2013].

[1] Indeed, what we call "2-operad" here could more traditionally be called "**Cat**-enriched symmetric semimulticategory" or "non-unital colored **Cat**-operad", which are quite verbose and justify our non-standard choice of terminology.

The operadic perspective, albeit **Set**-enriched, is already present in the framework of Fiore et al. [1999] and is at the heart of a recent paper of Hyland [2017].

Actually, we will be more interested in the following bichromatic presentation Λ_{k} of the untyped λ-calculus, which has the advantage of adapting straightforwardly to call-by-value:

- Λ_{k} has two colors, t (for terms) and v (for values);
- $\Lambda_{\mathsf{k}}(\Gamma; A)$, with $A \in \{\mathsf{t}, \mathsf{v}\}$, is non-empty only in case $\Gamma = \mathsf{v}, \ldots, \mathsf{v}$ (with n occurrences of v), in which case it is defined just like $\Lambda(n)$ above;
- operadic composition is again defined via substitution.

Note that Λ_{k} is not unital: there is no identity of type t. For the connoisseur, this presentation derives from taking seriously the idea that the λ-calculus is the Kleisli category of the exponential comonad of linear logic (hence the subscript k): intuitively, $\mathsf{v} = !\mathsf{t}$, and the asymmetry in $\mathsf{v} \to \mathsf{t}$ results from the translation of intuitionistic implication $A \to B$ as $!A \multimap B$.

The 2-operad Λ_{k} has three remarkable suboperads:

- Λ: this is the full suboperad on the color v, which is isomorphic to the monochromatic presentation given above;
- Λ_0: this is obtained by discarding terms of type v, *i.e.*, $\Lambda_0(\Gamma; A) = \Lambda_{\mathsf{k}}(\Gamma; A)$ in case $A = \mathsf{t}$, otherwise it is empty. Operadic composition in Λ_0 is degenerate: all terms are of type t but variables are always of type v, so we can never compose them! In spite of this, we will see that Λ_0 plays an important role in intersection types, as it is the basis of the so-called *strict* type discipline [van Bakel 1995], in which intersections are only allowed to the left of arrows, just like in Gardner-de Carvalho's system (Fig. 1).
- Λ_{v}: this is obtained by restricting the terms of $\Lambda_{\mathsf{v}}(\Gamma; \mathsf{v})$ to be *values*, *i.e.*, variables or abstractions, and by restricting 2-arrows to be those of call-by-value reduction, *i.e.*, $(\lambda x.M)V \to M\{V/x\}$ is only allowed when V is a value.

More 2-operads presenting calculi will be given in Sect. 3, along with examples of morphisms of 2-operads (Sect. 3.3). For the time being, let us just observe how the monochromatic presentation Λ may be straightforwardly be adapted to present the *simply typed* λ-calculus: colors are simple types (generated by $A, B ::= \alpha \mid A \to B$), multimorphisms Church-style simply-typed terms (*i.e.*, with explicit type decorations) and 2-arrows reductions between them. If we call such an operad Λ_{st}, forgetting the type decorations yields a (unital) morphism $(\cdot)^- : \Lambda_{\mathsf{st}} \to \Lambda$, which is a primordial example of type system expressed as morphism of 2-operads.

2.2 Type Systems as Fibrations

It is a standard observation that the type system corresponding to the morphism $(\cdot)^- : \Lambda_{\mathsf{st}} \to \Lambda$ defined above enjoys *subject reduction*: if $\Gamma \vdash M : A$ (*i.e.*, M is simply typable) and $M \to^* M'$, then $\Gamma \vdash M' : A$ too. This is proved by explicitly constructing, for any given derivation δ of $\Gamma \vdash M : A$, a derivation δ' of $\Gamma \vdash M' : A$. In categorical terms, we may reformulate the situation as follows: for every $\delta \in \Lambda_{\mathsf{st}}(\Gamma; A)$ and every 2-arrow $\rho : \delta^- \to^* M'$ in Λ, there exists a 2-arrow $\psi : \delta \to^* \delta'$ in Λ_{st} such that $\psi^- = \rho$. We say that ρ is an *op-lifting* of ψ. The "op" of course tells us that there is a dual notion, called *lifting*, in which sources and targets are reversed: we have $\rho : M' \to^* \delta^-$ and $\psi : \delta' \to \delta$. This, as the reader may have noticed, corresponds to *subject expansion*.

Liftings and op-liftings are at the heart of the categorical notion of *fibration* and *op-fibration*. *Conduché fibrations* [Conduché 1972] encompass both notions, making them an ideal setting to deal with subject expansion and subject reduction. Informally, these are functors lifting *factorizations* of arrows rather than arrows themselves. However, the usual notion of Grothendieck fibration, which is the one subsumed by Conduché fibrations, asks liftings to be minimal in a certain sense (the technical terminology is *cartesian*), a requirement that is quickly seen to be too strong for a

general treatment of type systems. We are therefore led to consider a weakened form of Conduché fibrations, originally studied (albeit in a slightly different form) by Niefield [2004]:

Definition 2.2 (Niefield fibration). Let $\mathcal{B}$ be a small category. A *Niefield fibration* on $\mathcal{B}$ is a functor $p : \mathcal{E} \to \mathcal{B}$, with $\mathcal{E}$ an arbitrary small category, verifying:

(faithfulness) p is faithful;
(identity lifting) for every k of $\mathcal{E}$, if $p(k)$ is an identity, then so is k;
(factorization lifting) for every arrow k of $\mathcal{E}$, if $p(k) = f' \circ f$ for some arrows f, f' of $\mathcal{B}$, then there exists a pair g, g' of arrows of $\mathcal{E}$ such that $k = g' \circ g$, $p(g) = f$ and $p(g') = f'$.[2]

Let $p_1 : \mathcal{E}_1 \to \mathcal{B}$ and $p_2 : \mathcal{E}_2 \to \mathcal{B}$ be Niefield fibrations. A *relational morphism* from p_1 to p_2 is a relation $R \subseteq \mathcal{E}_1 \times \mathcal{E}_2$ on the objects of $\mathcal{E}_1, \mathcal{E}_2$ such that:

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

Definition 2.3 (type system). We say that a morphism of small 2-operads $\mathbf{p} : \mathcal{E} \to \mathcal{B}$ is a *type system* if, for all objects Γ, A of $\mathcal{E}$, the functor $\mathbf{p}_{\Gamma;A}$ is a Niefield fibration.

In terms of programming languages, faithfulness, identity lifting and factorization lifting correspond to the following properties:

- given a reduction $\rho : M \to^* M'$ and type derivations δ of M and δ' of M', there is at most one typed reduction $\psi : \delta \to^* \delta'$ typing ρ (there may exist one such reduction for every couple of type derivations of M, M');
- empty reductions are never typed by non-empty reductions;
- reductions are typed "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 make sense to ask whether they comply with the above restrictions.

The interest of defining type systems as above is that we may build a quite robust framework around them, based on a variant of the *Grothendieck construction*. In its original incarnation, this is a statement relating fibrations and pseudo-presheaves: any Grothendieck fibration $p : \mathcal{E} \to \mathcal{B}$ induces a pseudofunctor $\partial p : \mathcal{B}^{\mathrm{op}} \to \mathbf{Cat}$, where **Cat** is the 2-category of small categories, by setting $\partial p(b) = p^{-1}(b)$, *i.e.*, the fiber over b (the category of all morphisms of $\mathcal{E}$ which are mapped to id_b via p). The Grothendieck construction is the inverse: given any pseudofunctor $F : \mathcal{B}^{\mathrm{op}} \to \mathbf{Cat}$, it constructs a category $\mathcal{E}\ell(F)$ and a fibration $\int F : \mathcal{E}\ell(F) \to \mathcal{B}$. Moreover, $\int$ and ∂ induce an equivalence between the category of fibrations (with suitable morphisms) and the pseudo-presheaf category $\mathbf{Cat}^{\mathcal{B}^{\mathrm{op}}}$. Variants of the Grothendieck construction are available for all usual notions of fibration; for instance, the above is true for Conduché fibrations as long as we replace **Cat** with **Dist**, the bicategory of distributors.

In order to prove the equivalence involving our operadic variant of the Grothendieck construction, we first need to define the category of type systems. A *relational morphism* R between two type systems $p_1 : \mathcal{E}_1 \to \mathcal{B}$ and $p_2 : \mathcal{E}_2 \to \mathcal{B}$ is

- a relation $R_0 \subseteq \mathcal{E}_1 \times \mathcal{E}_2$ between the objects of $\mathcal{E}_1$ and the objects of $\mathcal{E}_2$;

[2] This is the *weak factorization lifting property* of [Niefield 2004]. In Conduché fibrations, a form of minimality is asked.

- for all objects $\Gamma_1 = e_1^1, \ldots, e_1^n$ and e_1 of $\mathcal{E}_1$ and for all objects $\Gamma_2 = e_2^1, \ldots, e_2^n$ and e_2 of $\mathcal{E}_2$, such that for all $1 \leq i \leq n, (e_1^i, e_2^i) \in R_0$ and $(e_1, e_2) \in R_0$, a relational morphism $R^{\Gamma_1;e_1}_{\Gamma_2;e_2}$ between the Niefield fibrations $(\mathbf{p}_1)_{\Gamma_1;e_1}$ and $(\mathbf{p}_2)_{\Gamma_2;e_2}$.

Type systems over a 2-operad $\mathcal{B}$ and relational morphisms between them form a category, which we denote $\mathbf{TypeSys}(\mathcal{B})$.

We now move on to introduce what plays the role of **Dist** in our framework. Let $\mathfrak{Rel}$ be the following (large) bioperad:

- objects are small categories;
- multimorphisms $X_1 \ldots X_n \nrightarrow Y$ are relational distributors, that is functors
$$X_1 \times \cdots \times X_n \times Y^{\mathrm{op}} \to \mathbf{Rel};$$
- composition of multimorphisms is defined as the composition of distributors: given
$$G : Y_1 \ldots Y_m \nrightarrow Z \quad \text{and} \quad F : X_1 \ldots X_n \nrightarrow Y_i,$$
their composite $G \circ^i F$ is defined as the functor $Y_1 \times \cdots \times Y_{i-1} \times (X_1 \times \cdots \times X_n) \times Y_{i+1} \times \cdots \times Y_m \times Z^{\mathrm{op}} \to \mathbf{Rel}$
$$(y_1, \ldots, y_{i-1}, x_1, \ldots, x_n, y_{i+1}, \ldots, y_m; z) \mapsto$$
$$\int^{y_i \in Y_i} G(y_1, \ldots, y_i, \ldots, y_m; z) \times F(x_1, \ldots, x_n; y_i)$$
where the integral sign is the standard notation for a coend (it has nothing to do with the Grothendieck construction). Composition in $\mathfrak{Rel}$ is associative only modulo isomorphism.
- 2-arrows $\theta : F \Rightarrow G : X_1 \ldots X_n \to Y$ are natural transformations of the underlying functors: a family of relations indexed by $X_1 \times \cdots \times X_n \times Y^{\mathrm{op}}$:
$$\forall (x_1, \ldots, x_n, y) \in X_1 \times \cdots \times X_n \times Y^{\mathrm{op}},$$
$$\theta_{x_1,\ldots,x_n,y} \subseteq F(x_1, \ldots, x_n, y) \times G(x_1, \ldots, x_n, y).$$
satisfying naturality conditions.

A *lax natural transformation* between two lax morphisms $\theta : F \Rightarrow G : \mathcal{B} \to \mathfrak{Rel}$ is as follows:

- for each b in $\mathcal{B}$ a distributor $\theta_b : Fb \nrightarrow Gb$;
- for each $f : b_1 \ldots b_n \to b$ in $\mathcal{B}$, a 2-arrow $\theta_f : Gf \circ (\theta_{b_1}, \ldots, \theta_{b_n}) \Rightarrow \theta_b \circ Ff$, that is a family of relations indexed by the objects of $Fb_1 \times \cdots \times Fb_n \times Gb^{\mathrm{op}}$,
$$\forall (x_1, \ldots, x_n, y) \in Fb_1 \times \cdots \times Fb_n \times Gb^{\mathrm{op}},$$
$$(\theta_f)_{x_1,\ldots,x_n,y} \subseteq (Gf \circ (\theta_{b_1}, \ldots, \theta_{b_n}))(x_1, \ldots, x_n; y) \times (\theta_b \circ Ff)(x_1, \ldots, x_n; y),$$
that satisfy naturality conditions.

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

Theorem 2.4. *Let $\mathcal{B}$ be a small 2-operad. The category $\mathbf{TypeSys}(\mathcal{B})$ is equivalent to the category $\mathfrak{Rel}^{\mathcal{B}}$ of lax morphisms $\mathcal{B} \to \mathfrak{Rel}$ and relational lax natural transformations.*

Proof. $\int : \mathfrak{Rel}^{\mathcal{B}} \to \mathbf{TypeSys}(\mathcal{B})$ is defined by:

- given a lax functor $F : \mathcal{B} \to \mathfrak{Rel}$, we denote by $\mathcal{E}\ell(F)$ the following category:
  - the objects are pairs (b, x), where b is an object of $\mathcal{B}$ and $x \in Fb$;
  - a multimorphism $(b_1, x_1) \ldots (b_n, x_n) \to (b', x')$ is a pair (f, p) where $f : b_1 \ldots b_n \to b'$ is a multimorphism in $\mathcal{B}$ and $p \in Ff(x_1, \ldots, x_n; x')$;

– given

$$(g,q) : (b_1,y_1)\dots(b_m,y_m) \to (c,z),$$
$$(f,p) : (a_1,x_1)\dots(a_n,x_n) \to (b_i,y_i),$$

the composite $(g,q) \circ^i (f,p)$ is the pair $(g \circ^i f,(q,p))$.
– a 2-arrow $\theta : (f,p) \Rightarrow (g,q) : (a_1,x_1)\dots(a_n,x_n) \to (b,y)$ is a family of relations indexed by $Fa_1 \times \cdots \times Fa_n \times Fb$:

$$\forall(\alpha_1,\dots,\alpha_n,\beta) \in Fa_1 \times \cdots \times Fa_n \times Fb,$$
$$\theta_{\alpha_1,\dots,\alpha_n,\beta} \subseteq Ff(\alpha_1,\dots,\alpha_n,\beta) \times Fg(\alpha_1,\dots,\alpha_n,\beta).$$

such that

$$(p,q) \in \theta_{x_1,\dots,x_n,y}.$$

$\int F$ is just the first projection. It is not hard to check that it is a type system in the sense of Definition 2.3.

- Given a relational lax natural transformation $\theta : F \Rightarrow G$, we define the relation $(\int \theta)_0$ from the objects of $\mathcal{E}\ell(F)$ to the objects of $\mathcal{E}\ell(G)$ by $((b,x),(b',y)) \in (\int\theta)_0$ iff $b = b'$ and $(x,y) \in \theta_b$. For every list $\Gamma = (a_1,x_1),\dots,(a_n,x_n)$ of objects of $\mathcal{E}\ell(F)$ and object (b,y) of $\mathcal{E}\ell(F)$, and list $\Gamma' = (a'_1,x'_1),\dots,(a'_n,x'_n)$ of objects of $\mathcal{E}\ell(G)$ and object (b',y') of $\mathcal{E}\ell(G)$, the relation $\left(\int\theta\right)^{\Gamma;(b,y)}_{\Gamma';(b',y')}$ is empty unless $a_1 = a'_1,\dots,a'_n = a_n$ and $b = b'$, in which case, given

$$(f,p) : (a_1,x_1),\dots,(a_n,x_n) \to (b,y)$$
$$(g,q) : (a_1,x'_1),\dots,(a_n,x'_n) \to (b,y')$$

we have $((f,p),(g,q)) \in \left(\int\theta\right)^{\Gamma;(b,y)}_{\Gamma';(b',y')}$ just if $(x_1,x'_1) \in \theta_{a_1},\dots,(x_n,x'_n) \in \theta_{a_n},(y,y') \in \theta_b$. One may check that $\int\theta$ is a relational morphism.

$\partial : \mathbf{TypeSys}(\mathcal{B}) \to \mathfrak{Rel}^{\mathcal{B}}$ is defined by:

- given a type system $\mathbf{p} : \mathcal{E} \to \mathcal{B}$, we set, for b an object of $\mathcal{B}$,
  – $\partial\mathbf{p}(b) := \mathbf{p}^{-1}(b)$, *i.e.*, the subcategory of $\mathcal{E}_1$ whose objects are sent to b and whose morphisms are sent to id_b (usually called the *fiber* over b);
  – for $f : b_1\dots b_n \to b$ in $\mathcal{B}$, $\partial\mathbf{p}(f)$ is the distributor defined by:

$$\forall(e_1,\dots,e_n,e) \in \mathbf{p}^{-1}(b_1) \times \cdots \times \mathbf{p}^{-1}(b_n) \times \mathbf{p}^{-1}(b)^{\mathrm{op}},$$
$$\partial\mathbf{p}(f)(e_1,\dots,e_n;e) := \{g : e_1,\dots,e_n \to e \mid \mathbf{p}(g) = f\}$$

and, for all $1 \le i \le n$, given $k_i : e_i \to e'_i$ arrows of $\mathbf{p}^{-1}(b_i)$ and $k : e' \to e$ an arrow of $\mathbf{p}^{-1}(b)$, respectively,

$$\partial\mathbf{p}(f)(k_1,\dots,k_n;k) := \{(g,g') \mid g = k \circ g' \circ (k_1,\dots,k_n)\};$$

  – for $\theta : f \Rightarrow f' : b_1\dots b_n \to b$ in $\mathcal{B}$, $\partial\mathbf{p}(\theta)_{e_1,\dots,e_n;e}$ is the relation

$$\{(g,g') \in \partial\mathbf{p}(f)(\overline{e};e) \times \partial\mathbf{p}(f')(\overline{e};e) \mid \exists\rho : g \Rightarrow g', \mathbf{p}(\rho) = \theta\}.$$

This defines a lax functor $\partial p : \mathcal{B} \to \mathfrak{Rel}$.

- Given a relational morphism R between type systems $\mathbf{p}_1 : \mathcal{E}_1 \to \mathcal{B}$ and $\mathbf{p}_2 : \mathcal{E}_2 \to \mathcal{B}$,
  – for b an object of $\mathcal{B}$, ∂R_b is R restricted to $\mathbf{p}_1^{-1}(b) \times \mathbf{p}_2^{-1}(b)$, *i.e.*, the relations R_0 and $R^{e_1,e'_1}_{e_2,e'_2}$ (with $e_1,e'_1 \in \mathbf{p}_1^{-1}(b)$ and $e_2,e'_2 \in \mathbf{p}_2^{-1}(b)$) induce a distributor $\partial\mathbf{p}_1(b) \nrightarrow \partial\mathbf{p}_2(b)$ which is a relation (*i.e.*, valued in a subsingleton), and we take this to be ∂R_b.

– with the above definition, given $f : b_1, \ldots, b_n \to b$ in $\mathcal{B}$, we have

$$\partial R_b \circ \partial \mathbf{p}_1(f) \cong \{g_1 : e_1^1, \ldots, e_1^n \to e_1 \mid \mathbf{p}_1(g_1) = f\},$$
$$\partial \mathbf{p}_2(f) \circ (\partial R_{b_1}, \ldots, \partial R_{b_n}) \cong \{g_2' : e_2^1, \ldots, e_2^n \to e_2 \mid \mathbf{p}_2(g_2') = f\}.$$

The family of relations $(\partial R_f)_{e_1^1,\ldots,e_1^n;e_2}$ is then defined to contain all pairs (g_2', g_1) such that there exists g_1' and a 2-arrow $g_1 \Rightarrow g_1'$ such that $(g_1', g_2') \in R^{\overline{e}_1;e_1}_{\overline{e}_2,e_2}$ (which, by definition of relational morphism of Niefield fibrations, is equivalent to the existence of g_2 and a two arrow $g_2 \Rightarrow g_2'$ such that $(g_1, g_2) \in R^{\overline{e}_1;e_1}_{\overline{e}_2,e_2}$).

This defines ∂R as a relational lax natural transformation between $\partial \mathbf{p}_1$ and $\partial \mathbf{p}_2$.

The fact that $\int(-)$ and $\partial(-)$ form an equivalence of categories follows from elementary calculations, applying the above definitions. □

3 POLYADIC CALCULI

3.1 Terms and Reduction

We fix two disjoint, countably infinite sets of *linear* and *polyadic* variables, ranged over by a, b, c and x, y, z, respectively. Polyadic terms are defined as follows:

$$t, u ::= a \mid \lambda a.t \mid tu \mid x \mid \langle t_1, \ldots, t_n \rangle \mid t[\langle x_1, \ldots, x_n \rangle := u] \mid \bot,$$

where:

- in $t[\langle x_1, \ldots, x_n \rangle := u]$, $x_1, \ldots, x_n$ are pairwise distinct and are all bound in t;
- modulo Barendregt's convention (*i.e.*, each binder binds a distinct variable), every linear variable appears at most once and, in $\lambda a.t$, a must appear in t;
- in $\langle t_1, \ldots, t_n \rangle$, each t_i has no linear free variable (polyadic free variables are allowed).

The term $\bot$ is needed for technical reasons which will be clarified momentarily; it will disappear as soon as we will introduced types. As usual, terms are considered up to renaming of bound variables. A term is

- *affine* if, modulo Barendregt's convention, every polyadic variable appears in it at most once;
- *relevant* if, for every of its subterms of the form $t[\langle x_1, \ldots, x_n \rangle := u]$, each x_i appears free in t;
- *linear* if it is both affine and relevant.

Unconstrained terms are also called *cartesian*.

Linear substitution $t\{u/a\}$ of a term u for a linear variable a in t is defined as usual. *Polyadic substitution* $t\{u_1, \ldots, u_n/x_1, \ldots, x_m\}$ is defined by simultaneously substituting each u_i to x_i; if $m < n$, the terms $u_{m+1}, \ldots, u_n$ are discarded; if $m > n$, the variables $x_{n+1}, \ldots, x_m$ are replaced with $\bot$. This explains the presence of $\bot$ in the untyped syntax; the simple types discipline will make it unnecessary.

For the reduction semantics, we adopt the approach introduced by Accattoli [2012], which has the advantage of avoiding commuting conversions. This approach is based on considering reduction rules modulo the presence of arbitrary *substitution contexts*, defined as follows:

$$[-] ::= \{\cdot\} \mid [-][\langle x_1, \ldots, x_n \rangle := u].$$

We write $t[-]$ for the term obtained by replacing the hole $\{\cdot\}$ with the term t. Reduction is the closure under arbitrary contexts of the following rules:

$$(\lambda a.t)[-]u \quad \to \quad t\{u/a\}[-]$$
$$t[\langle x_1, \ldots, x_m \rangle := \langle u_1, \ldots, u_n \rangle[-]] \quad \to \quad t\{u_1, \ldots, u_n/x_1, \ldots, x_m\}[-].$$

In the relevant (and linear) case, we ask $m \geq n$ (because we are not allowed to discard terms).

$$\frac{}{;a:A\vdash a:A}\ \text{var} \qquad \frac{\Gamma;\Delta,a:A\vdash t:B}{\Gamma;\Delta\vdash \lambda a.t:A\multimap B}\ \text{lam} \qquad \frac{\Gamma;\Delta\vdash t:A\multimap B \quad \Gamma';\Delta'\vdash u:A}{\Gamma,\Gamma';\Delta,\Delta'\vdash tu:B}\ \text{app}$$

$$\frac{}{x:A;\vdash x:A}\ \text{pvar} \qquad \frac{\Gamma_1;\vdash t_1:A_1 \quad\dots\quad \Gamma_n;\vdash t_n:A_n}{\Gamma_1,\dots,\Gamma_n;\vdash\langle t_1,\dots,t_n\rangle:\langle A_1,\dots,A_n\rangle}\ \text{box}$$

$$\frac{\Gamma';\Delta'\vdash u:\langle A_1,\dots,A_n\rangle \quad \Gamma,x_1:A_1,\dots,x_n:A_n;\Delta\vdash t:C}{\Gamma,\Gamma';\Delta,\Delta'\vdash t[\langle x_1,\dots,x_n\rangle:=u]:C}\ \text{let}$$

$$\frac{\Gamma;\Delta\vdash t:C}{\Gamma,x:A;\Delta\vdash t:C}\ \text{weak} \qquad \frac{\Gamma,x:A,y:A;\Delta\vdash t:C}{\Gamma,x:A;\Delta\vdash t\{x/y\}:C}\ \text{cntr}$$

Fig. 2. Polyadic simple types derivations. We omit the obvious exchange rules on contexts.

An example: $(\lambda a.a)[\langle x\rangle := u]t \to t[\langle x\rangle := u]$ is a valid reduction; in a more traditional definition, such a term would need the commutation $(\lambda a.a)[\langle x\rangle := u]t \rightsquigarrow ((\lambda a.a)t)[\langle x\rangle := u]$ in order to reduce. While not strictly necessary, Accattoli's approach is extremely convenient for us.

In what follows, we fix a repetition-free sequence $(a^i)_{i\in\mathbb{N}}$ of linear variables, as well as a repetition-free sequence $(x^i_j)_{i,j\in\mathbb{N}}$ of polyadic variables. For a fixed $i\in\mathbb{N}$, the sequence $x^i_1,x^i_2,x^i_3,\dots$ is denoted by $\overline{x}^i$ and called *supervariable*. From now on, in the terms of the form $t[\langle x_1,\dots,x_n\rangle := u]$ we use α-equivalence to assume that $x_1,\dots,x_n$ are always an initial segment of a supervariable. The 2-operad Λ_{p} of (cartesian) polyadic terms is defined as follows:

- it has two objects, l (for "linear") and p (for "polyadic");
- given $\Gamma = \mathrm{s}_1,\dots,\mathrm{s}_n$ with $\mathrm{s}_i\in\{\mathrm{l},\mathrm{p}\}$, $\Lambda_{\mathrm{p}}(\Gamma;\mathrm{l})$ is the category whose
  - objects are all terms t such that, for all $1\le i\le n$: if $\mathrm{s}_i=\mathrm{l}$, then a^i is free in t; if $\mathrm{s}_i=\mathrm{p}$, then any variable of $\overline{x}^i$ may be free in t; and no other variable may be free in t;
  - morphisms are reductions modulo permutation equivalence;
- $\Lambda_{\mathrm{p}}(\Gamma;\mathrm{p})$ is the full subcategory of $\Lambda_{\mathrm{p}}(\Gamma;\mathrm{l})$ restricted to terms of the form $\langle t_1,\dots,t_n\rangle$ (note that this is empty in case Γ contains l);
- operadic composition on the color l is defined by $t\circ^i_{\mathrm{l}} u := t\{u/a^i\}$, while on the color p we set

$$t\circ^i_{\mathrm{p}}\langle u_1,\dots,u_n\rangle := t\{u_1,\dots,u_n/x^i_1,\dots,x^i_m\},$$

  where m is the largest such that x^i_m appears free in t. Note that the second term is of the form $\langle u_1,\dots,u_n\rangle$ because, by definition, every term of color p must be of this form.

3.2 Simple Types

The *polyadic simple types* are defined as follows:

$$A,B ::= \alpha \mid A\multimap B \mid \langle A_1,\dots,A_n\rangle.$$

The simply-typed polyadic calculi are defined by the rules of Fig. 2. Typing judgments are of the form $\Gamma;\Delta\vdash t:A$, where Γ (resp. Δ) is the polyadic (resp. linear) context and contains assignments of the form $x:C$ (resp. $a:B$). Note that, as anticipated, $\bot$ is not typable, nor is any term containing it.

We want to stress that the type system of Fig. 2 is far from original: if one erases all term annotations, one obtains a natural deduction formulation of a logical system such that

- if one reads $\langle A_1, \ldots, A_n \rangle$ as $A_1 \otimes \cdots \otimes A_n$, all rules except weak and cntr are derivable in the $\otimes/\multimap$ fragment of multiplicative linear logic;
- if one reads $\langle A_1, \ldots, A_n \rangle$ as $(A_1 \& 1) \otimes \cdots \otimes (A_n \& 1)$, then weak too is derivable in multiplicative additive linear logic;
- if one reads $\langle A_1, \ldots, A_n \rangle$ as $!A_1 \otimes \cdots \otimes !A_n$, then every rule is derivable in multiplicative exponential linear logic.

We are now going to define a type system $(\cdot)^- : \mathbf{Poly} \longrightarrow \Lambda_{\mathrm{p}}$ corresponding to Fig. 2, where **Poly** is a suitable 2-operad of simply-typed derivations/Church-style terms. To define it, we follow the two-sorted definition of Λ_{p}, declining it over all polyadic simple types:

- the objects are of the form A or $[A]$, where A ranges over polyadic simple types;
- given sequences $[\Gamma]$ and Δ of objects and an interleaving Σ of them, $\mathbf{Poly}(\Sigma; A)$ is the category whose
  - objects are the type derivations (or, equivalently, Church-style terms) of $\Gamma; \Delta \vdash t : A$, where t is a term whose free variables are a^i and x^i_j as in the definition of Λ_{p};
  - morphisms are reductions (of Church-style terms) modulo permutation equivalence;
- $\mathbf{Poly}(\Sigma; [A])$ is the full subcategory of $\mathbf{Poly}(\Sigma; A)$ restricted to derivations of terms of the form $\langle t_1, \ldots, t_n \rangle$;
- operadic composition is substitution, with the rules var and pvar playing the role of identities (on A and $[A]$, respectively).

By disallowing the structural rules (weak and cntr), we have obvious suboperads **RelvPoly**, **AffPoly** and **LinPoly** of **Poly**. Of course, these may only type relevant, affine and linear terms, respectively. To stress the use of unrestricted structural rules, we may write **CartPoly** as a synonym of **Poly**.

The forgetful morphism $(\cdot)^- : \mathbf{Poly} \to \Lambda_{\mathrm{p}}$ is now easy to define: it takes objects of the form A to l and $[A]$ to p, and it is defined in the obvious way for the rest.

The main property ensured by simple types is strong normalization:

Proposition 3.1. *Simply-typed polyadic terms strongly normalize.*

Proof. As observed above, **Poly** may be embedded in propositional multiplicative exponential linear logic, whose strong normalization is well known [Girard 1987]. □

It is worth mentioning that, for **LinPoly** or **AffPoly**, Proposition 3.1 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).

3.3 Intuitionistic Linear Logic and Girard's Embeddings

We consider the following calculus $\Lambda_!$ of proof terms for linear logic:

$$T, U ::= a \mid \lambda a.T \mid TU \mid x \mid !T \mid T[!x := U],$$

where a and x range over disjoint sets of *linear* and *cartesian* variables, respectively. Linear variables must respect the same linearity constraint as in polyadic calculi; in $T[!x := U]$, x is bound in T; and, in $!T$, which is called a *box*, T must not contain free linear variables.

Substitution contexts are defined just like in the polyadic calculus and reduction is the closure under arbitrary contexts of the following rules:

$$(\lambda a.T)[-]U \;\to\; T\{U/a\}[-] \qquad\qquad T[!x := !U[-]] \;\to\; T\{U/x\}[-].$$

The 2-operad $\Lambda_!$ corresponding to the above calculus is defined by adapting the definition of Λ_{p}: its objects are l and c (for "cartesian"); the multimorphisms of source $\Gamma \in \{\mathsf{l}, \mathsf{c}\}^*$ and target

l are terms with free variables matching Γ (cartesian variables need not actually be free); the multimorphisms of target c are restricted to boxes (*i.e.*, of the form $!T$); 2-arrows are reductions modulo permutation equivalence; and operadic composition is defined by $t \circ_{\mathsf{l}}^{i} u := t\{u/a^i\}$ for the color l, and by $t \circ_{\mathsf{c}}^{i} !u := t\{u/x^i\}$ for the color c.

We are now in position of giving a couple of non-trivial examples of encodings of term calculi seen as morphisms of 2-operads. We start with a morphism $\mathbf{G}_0 : \Lambda_0 \longrightarrow \Lambda_!$, defined as follows (the definition of Λ_0 is in the final part of Sect. 2.1):

- on types, $\mathbf{G}_0(\mathsf{t}) := \mathsf{l}$ and $\mathbf{G}_0(\mathsf{v}) := \mathsf{c}$;
- on terms,

$$\mathbf{G}_0(x) := x, \qquad \mathbf{G}_0(\lambda x.M) := \lambda a.\mathbf{G}_0(M)[!x := a], \qquad \mathbf{G}_0(MN) := \mathbf{G}_0(M)!\mathbf{G}_0(N);$$

  an immediate induction shows that $\mathbf{G}_0(M\{N/x\}) = \mathbf{G}_0(M)\{\mathbf{G}_0(N)/x\}$.
- On reductions, it is enough to define $\mathbf{G}_0(\beta)$, where $\beta : (\lambda x.M)N \to^* M\{N/x\}$ is the generating 2-arrow; this is set to be the following reduction:

$$\begin{aligned}\mathbf{G}_0((\lambda x.M)N) &= (\lambda a.\mathbf{G}_0(M)[!x := a])!\mathbf{G}_0(N) \to \mathbf{G}_0(M)[!x := !\mathbf{G}_0(N)] \\ &\to \mathbf{G}_0(M)\{\mathbf{G}_0(N)/x\} = \mathbf{G}_0(M\{N/x\}).\end{aligned}$$

The above encoding may be extended to the whole 2-operad Λ_{k}. We define $\mathbf{G}_{\mathsf{k}} : \Lambda_{\mathsf{k}} \longrightarrow \Lambda_!$ by setting $\mathbf{G}_{\mathsf{k}}(A) := \mathbf{G}_0(A)$ on types; $\mathbf{G}_{\mathsf{k}}(M : \mathsf{t}) := \mathbf{G}_0(M)$ and $\mathbf{G}_{\mathsf{k}}(M : \mathsf{v}) := !\mathbf{G}_0(M)$ on terms; and $\mathbf{G}_{\mathsf{k}}$ is defined like $\mathbf{G}_0$ on reductions.

Both $\mathbf{G}_0$ and $\mathbf{G}_{\mathsf{k}}$ correspond to the call-by-name embedding of the λ-calculus in linear logic, originally presented by Girard [1987]. In that paper, Girard also defined a call-by-value embedding, which induces a morphism $\mathbf{G}_{\mathsf{v}} : \Lambda_{\mathsf{v}} \longrightarrow \Lambda_!$ defined as follows (Λ_{v} is also defined in Sect. 2.1):

- on colors, $\mathbf{G}_{\mathsf{v}}(\mathsf{t}) := \mathsf{l}$ and $\mathbf{G}_{\mathsf{v}}(\mathsf{v}) := \mathsf{c}$;
- on terms, we must distinguish whether the λ-term being encoded is of type t or v:

$$\begin{aligned}\mathbf{G}_{\mathsf{v}}(x : \mathsf{v}) &:= !x & \mathbf{G}_{\mathsf{v}}(MN : \mathsf{t}) &:= \xi[!\xi := \mathbf{G}_{\mathsf{v}}(M : \mathsf{t})]\mathbf{G}_{\mathsf{v}}(N : \mathsf{t}) \\ \mathbf{G}_{\mathsf{v}}(\lambda x.M : \mathsf{v}) &:= !(\lambda a.\mathbf{G}_{\mathsf{v}}(M : \mathsf{t})[!x := a]) & \mathbf{G}_{\mathsf{v}}(V : \mathsf{t}) &:= \mathbf{G}_{\mathsf{v}}(V : \mathsf{v}).\end{aligned}$$

  Note that the last line is not circular, *i.e.*, whatever V is, a $\mathbf{G}_{\mathsf{v}}$ is invoked on strictly smaller subterm of it. It is easy to check that $\mathbf{G}_{\mathsf{v}}(M\{V/x\} : \mathsf{t}) = \mathbf{G}_{\mathsf{v}}(M : \mathsf{t})\{\mathbf{G}_{\mathsf{v}}(V : \mathsf{v})/x\}$.
- on reductions, we set if $\beta_{\mathsf{v}} : (\lambda x.M)V \to^* M\{V/x\}$ is the generating 2-arrow, we take $\mathbf{G}_{\mathsf{v}}(\beta_{\mathsf{v}})$ to be the following reduction:

$$\begin{aligned}\mathbf{G}_{\mathsf{v}}((\lambda x.M)V : \mathsf{t}) &= \xi[!\xi := !(\lambda a.\mathbf{G}_{\mathsf{v}}(M : \mathsf{t})[!x := a])]!\mathbf{G}_{\mathsf{v}}(V : \mathsf{v}) \\ &\to (\lambda a.\mathbf{G}_{\mathsf{v}}(M : \mathsf{t})[!x := a])!\mathbf{G}_{\mathsf{v}}(V : \mathsf{v}) \to \mathbf{G}_{\mathsf{v}}(M : \mathsf{t})[!x := !\mathbf{G}_{\mathsf{v}}(V : \mathsf{v})] \\ &\to \mathbf{G}_{\mathsf{v}}(M : \mathsf{t})\{\mathbf{G}_{\mathsf{v}}(V : \mathsf{v})/x\} = \mathbf{G}_{\mathsf{v}}(M\{V/x\} : \mathsf{t}).\end{aligned}$$

In all cases, we glossed over a detail: technically, since the 2-arrows of our operads are defined only up to permutation equivalence, we need to check that the image of Diagram 4 of Sect. 2.1 via the embeddings commutes. Fortunately, this is just a straightforward verification.

3.4 Polyadic Approximations

Definition 3.2 (approximation relations). The approximation relations on $\Lambda_{\mathsf{p}} \times \Lambda_!$ are defined inductively, as in Fig. 3. The judgments are of the same form given in the introduction (Sect. 1.1). There are four variants of the relation, one for each of the four variants of Λ_{p}, depending on which structural rule(s) (*i.e.*, weak and cntr) are allowed. We write $t \sqsubset T$ to mean that $\Gamma \vdash t \sqsubset T$ is derivable from the (suitable variant of) the rules of Fig. 3 for some context Γ.

$$\frac{}{\vdash a \sqsubset a}\ \text{var} \qquad \frac{\Gamma \vdash t \sqsubset T}{\Gamma \vdash \lambda a.t \sqsubset \lambda a.T}\ \text{lam} \qquad \frac{\Gamma \vdash t \sqsubset T \quad \Gamma' \vdash u \sqsubset U}{\Gamma,\Gamma' \vdash tu \sqsubset TU}\ \text{app}$$

$$\frac{x_0 \text{ polyadic}, x \text{ cartesian}}{x_0 \sqsubset x \vdash x_0 \sqsubset x}\ \text{pvar} \qquad \frac{\Gamma_1 \vdash t_1 \sqsubset T \quad \dots \quad \Gamma_n \vdash t_n \sqsubset T}{\Gamma_1,\dots,\Gamma_n \vdash \langle t_1,\dots,t_n\rangle \sqsubset\ !T}\ \text{box}$$

$$\frac{\Gamma' \vdash u \sqsubset U \quad \Gamma, x_1 \sqsubset x,\dots,x_n \sqsubset x \vdash t \sqsubset T}{\Gamma,\Gamma' \vdash t[\langle x_1,\dots,x_n\rangle := u] \sqsubset T[!x := U]}\ \text{let}$$

$$\frac{\Gamma \vdash t \sqsubset T}{\Gamma, x_0 \sqsubset x \vdash t \sqsubset T}\ \text{weak} \qquad \frac{\Gamma, x_0 \sqsubset x, x_1 \sqsubset x \vdash t \sqsubset T}{\Gamma, x_0 \sqsubset x \vdash t\{x_0/x_1\} \sqsubset T}\ \text{cntr}$$

Fig. 3. Polyadic approximations. In the let rule, x does not appear in Γ.

$$\frac{t \sqsubset T}{\mathrm{id}_t \sqsubset \mathrm{id}_T}\ \text{id} \qquad \frac{\rho \sqsubset \varphi \quad \rho' \sqsubset \varphi'}{\rho;\rho' \sqsubset \varphi;\varphi'}\ \text{comp} \qquad \frac{\rho \sqsubset \varphi \quad \mathrm{S}_0 \sqsubset \mathrm{S}}{\mathrm{S}_0\{\rho\} \sqsubset \mathrm{S}\{\varphi\}}\ \text{sh ctxt} \qquad \frac{\rho_1 \sqsubset \varphi \quad \dots \quad \rho_n \sqsubset \varphi}{\langle \rho_1,\dots,\rho_n\rangle \sqsubset\ !\varphi}\ \text{box}$$

$$\frac{\begin{array}{l}\rho : (\lambda a.t)[-]_0 u \to t\{u/a\}[-]_0 \\ \varphi : (\lambda a.T)[-]U \to T\{U/a\}[-] \qquad t \sqsubset T \qquad u \sqsubset U \qquad [-]_0 \sqsubset [-]\end{array}}{\rho \sqsubset \varphi}\ \text{mul}$$

$$\frac{\begin{array}{l}\rho : t[\langle \vec{x}\rangle := \langle \vec{u}\rangle[-]_0] \to t\{\vec{u}/\vec{x}\}[-]_0 \\ \varphi : T[!x := !U[-]] \to T\{U/x\}[-] \qquad t \sqsubset T \qquad u_i \sqsubset U \qquad [-]_0 \sqsubset [-]\end{array}}{\rho \sqsubset \varphi}\ \text{exp}$$

Fig. 4. Approximation relation on reductions.

The approximation relation may be extended to reduction sequences. We first need to fix some notations and terminology. A context is *shallow* if

- in Λ_{p}, the hole does not appear inside a $\langle \dots \rangle$;
- in $\Lambda_!$, the hole does not appear under a $!(-)$.

Substitution contexts are example of shallow contexts. The approximation relation is extended to shallow contexts in the obvious way (the hole is treated like a linear variable).

The identity (*i.e.*, empty) reduction sequence on a term t is denoted by id_t; if $\rho : t \to^* t'$ and $\rho' : t' \to^* t''$, their concatenation is denoted by $\rho;\rho'$. If $\rho : t \to^* t'$ and S is a shallow context, there is an obvious reduction sequence $\mathrm{S}\{t\} \to^* \mathrm{S}\{t'\}$, which we denote by $\mathrm{S}\{\rho\}$. Similar notations are used for $\Lambda_!$. If $\rho_i : t_i \to^* t_i'$ are reduction sequences in Λ_{p}, with $1 \le i \le n$, there are several obvious reduction sequences $\langle t_1,\dots,t_n\rangle \to^* \langle t_1',\dots,t_n'\rangle$; these are all permutation equivalent and are denoted by $\langle \rho_1,\dots,\rho_n\rangle$. Similarly, if $\varphi : T \to^* T'$ is a reduction sequence in $\Lambda_!$, there is an obvious reduction sequence $!T \to^* !T'$, which we denote by $!\varphi$.

Definition 3.3 (approximating reductions). The approximation relation (in any of its variants) for reductions is defined by means of the inductive rules of Fig. 4, where:

- in the comp rule, the sequences are assumed to be composable;

- in the sh ctxt rule, S_0 and S are shallow contexts;
- in the mul and exp rules, ρ and φ are the obvious one-step sequences reducing the redex on the left hand side.

4 INTERSECTION TYPES FROM POLYADIC APPROXIMATIONS

4.1 The Approximation Presheaf

Fix a suboperad $\mathcal{D} \hookrightarrow \mathbf{Poly}$.

Definition 4.1 (category of types). We define the following sets, seen as discrete categories:

$$\mathcal{T}_{\mathsf{l}}[\mathcal{D}] := \{A \mid \text{the polyadic type } A \text{ is a color of } \mathcal{D}\};$$
$$\mathcal{T}_{\mathsf{c}}[\mathcal{D}] := \{\langle A_1, \dots, A_n\rangle \mid A_i \in \mathcal{T}_{\mathsf{l}}[\mathcal{D}]\}.$$

Definition 4.2 (approximation presheaf). We define a lax morphism of bioperads

$$\mathrm{Apx}[\mathcal{D}] : \Lambda_! \longrightarrow \mathfrak{Rel}$$

as follows:

- on objects, $\mathrm{Apx}[\mathcal{D}](\mathsf{l}) := \mathcal{T}_{\mathsf{l}}[\mathcal{D}]$ and $\mathrm{Apx}[\mathcal{D}](\mathsf{c}) := \mathcal{T}_{\mathsf{c}}[\mathcal{D}]$;
- given $T \in \Lambda_!(\mathsf{c}^m, \mathsf{l}^n; \mathsf{s})$ with $\mathsf{s} \in \{\mathsf{l}, \mathsf{c}\}$, we must define a functor $\mathrm{Apx}[\mathcal{D}](T) : \mathcal{T}_{\mathsf{c}}[\mathcal{D}]^m \times \mathcal{T}_{\mathsf{l}}[\mathcal{D}]^n \times \mathcal{T}_{\mathsf{s}}[\mathcal{D}]^{\mathrm{op}} \to \mathbf{Rel}$; since the source categories are discrete, this is just a map assigning a set to each element of $\mathcal{T}_{\mathsf{c}}[\mathcal{D}]^m \times \mathcal{T}_{\mathsf{l}}[\mathcal{D}]^n \times \mathcal{T}_{\mathsf{s}}[\mathcal{D}]$. Let $\Theta \in \mathcal{T}_{\mathsf{c}}[\mathcal{D}]^m$, *i.e.*,
$$\Theta = \langle B_1^1, \dots, B_{k_1}^1\rangle, \dots, \langle B_1^m, \dots, B_{k_m}^m\rangle;$$
we define $\overline{\Theta}$ to be the polyadic context containing exactly the judgments $x_j^i : B_j^i$ for all $1 \le i \le m$, $1 \le j \le k_i$. Then, for $\Gamma \in \mathcal{T}_{\mathsf{l}}[\mathcal{D}]^n$ and $A \in \mathcal{T}_{\mathsf{s}}[\mathcal{D}]$, we set
$$\mathrm{Apx}[\mathcal{D}](T)(\Theta, \Gamma; A) := \left\{\delta \in \mathcal{D}(\overline{\Theta}, \Gamma; A) \mid \Xi \vdash \delta^- \sqsubset T\right\},$$
where Ξ consists of exactly $x_j^i \sqsubset x^i$ for all $1 \le i \le m$, $1 \le j \le k_i$.
- given $T, T' \in \Lambda_!(\mathsf{c}^m, \mathsf{l}^n; \mathsf{s})$ with $\mathsf{s} \in \{\mathsf{l}, \mathsf{c}\}$ and $\varphi : T \to^* T'$, $\mathrm{Apx}[\mathcal{D}](\varphi)$ must be a natural transformation from $\mathrm{Apx}[\mathcal{D}](T)$ to $\mathrm{Apx}[\mathcal{D}](T')$; again, since the source categories of these distributors are discrete, this is just a family of relations indexed by $\mathcal{T}_{\mathsf{c}}[\mathcal{D}]^m \times \mathcal{T}_{\mathsf{l}}[\mathcal{D}]^n \times \mathcal{T}_{\mathsf{s}}[\mathcal{D}]$; we define it as follows:
$$\mathrm{Apx}[\mathcal{D}](\varphi)_{\Theta,\Gamma;A} := \Big\{(\delta, \delta') \in \mathrm{Apx}[\mathcal{D}](T)(\overline{\Theta}, \Gamma, A) \times \mathrm{Apx}[\mathcal{D}](T')(\overline{\Theta}, \Gamma, A) \mid \exists \tau : \delta \to^* \delta' \text{ in } \mathcal{D}(\overline{\Theta}, \Gamma; A) \text{ s.t. } \Xi \vdash \tau^- \sqsubset \varphi\Big\},$$
with $\overline{\Theta}$ and Ξ defined as above. So, $(\delta, \delta') \in \mathrm{Apx}[\mathcal{D}](\varphi)_{\Theta,\Gamma;A}$ if these are related by a typed reduction approximating φ.

Suppose now that we have a morphism of operads $\mathbf{G} : \mathcal{L} \to \Lambda_!$, *i.e.*, $\mathcal{L}$ is a programming language admitting a semantic-preserving embedding in intuitionistic linear logic. Then, we have a lax morphism

$$\mathcal{L} \xrightarrow{\mathbf{G}} \Lambda_! \xrightarrow{\mathrm{Apx}[\mathcal{D}]} \mathfrak{Rel}$$

to which we may apply the Grothendieck construction of Theorem 2.4, yielding a type system for $\mathcal{L}$ in the sense of Definition 2.3:

Definition 4.3 ($(\mathcal{D}, \mathbf{G})$-type system). Let $\mathcal{D} \hookrightarrow \mathbf{Poly}$ and $\mathbf{G} : \mathcal{L} \to \Lambda_!$. We define the abbreviations
$$\mathbf{p}[\mathcal{D}, \mathbf{G}] := \textstyle\int(\mathrm{Apx}[\mathcal{D}] \circ \mathbf{G}), \qquad \mathcal{E}[\mathcal{D}, \mathbf{G}] := \mathcal{E}\ell(\mathrm{Apx}[\mathcal{D}] \circ \mathbf{G}),$$

and say that a multimorphism M of $\mathcal{L}$ is $(\mathcal{D},\mathbf{G})$-typable if it is in the image of $\mathbf{p}[\mathcal{D},\mathbf{G}]$ or, equivalently, there are Γ, A such that $(\mathrm{Apx}[\mathcal{D}] \circ \mathbf{G})(M)(\Gamma; A) \neq \emptyset$.

In Sect. 4.3 and 4.4 we will give ample evidence that this is an intersection type system for $\mathcal{L}$, in accord with the Equivalence 3 of Sect. 1.2. For the time being, let us give an idea of what the 2-operad $\mathcal{E}[\mathcal{D},\mathbf{G}]$ looks like. By applying the definition given in the proof of Theorem 2.4, we have:

- an object of $\mathcal{E}[\mathcal{D},\mathbf{G}]$ is a pair (a,A) where a is a type of $\mathcal{L}$ and A a type of $\mathcal{T}_{\mathsf{I}}[\mathcal{D}]$ or $\mathcal{T}_{\mathsf{c}}[\mathcal{D}]$, depending on whether the image of a through $\mathbf{G}$ is I or c; in other words, it is a type of $\mathcal{L}$ refined by a polyadic simple type declared "valid" by $\mathcal{D}$.
- Given types $(a,A),(\gamma,\Gamma) = (c_1,C_1),\dots,(c_n,C_n)$ and
$$(\theta,\Theta) = (b_1,\langle B^1_1,\dots,B^1_{k_1}\rangle),\dots,(b_m,\langle B^m_1,\dots,B^m_{k_m}\rangle),$$
where we assume that $\mathbf{G}(c_i) = \mathsf{I}$ ad $\mathbf{G}(b_i) = \mathsf{c}$ for all i, a multimorphism of
$$\mathcal{E}[\mathcal{D},\mathbf{G}]((\theta,\Theta),(\gamma,\Gamma);(a,A))$$
is a pair (M,δ) composed of a term $M \in \mathcal{L}(\theta,\gamma;a)$ and a "valid" simply-typed derivation δ of $\overline{\Theta};\Gamma \vdash t : A$ such that $t \sqsubset \mathbf{G}(M)$.
- Similarly, a 2-arrow $(M,\delta) \to (M',\delta')$ is a pair (ψ,τ) such that $\psi : M \to^* M'$ in $\mathcal{L}$, $\tau : \delta \to^* \delta'$ in $\mathcal{D}$ and $\tau^- \sqsubset \mathbf{G}(\psi)$, *i.e.*, $M \to^* M'$ via ψ and there is a polyadic reduction approximating ψ which may be given a "valid" type as a reduction $\delta \to^* \delta'$.

In particular,
$$M \text{ is } (\mathcal{D},\mathbf{G})\text{-typable iff there exists } \delta \text{ in } \mathcal{D}(\Gamma;A) \text{ such that } \delta^- \sqsubset \mathbf{G}(M),$$
exactly as in Equivalence 3 of Sect. 1.2.

4.2 Capturing Dynamic Properties

Intersection types are known for their ability to capture dynamic (or runtime) properties of programs, most notably various kinds of termination. We will start by giving a somewhat general definition of what a "dynamic property" may be in our context.

Definition 4.4 (dynamic property). Let $\mathcal{L}$ be a 2-operad. A *strong dynamic property* for $\mathcal{L}$ is a set of 2-arrows of $\mathcal{L}$. Given such a set $\mathcal{R}$, we let $\mathcal{S}(\mathcal{R})$ be the set of all multimorphisms M of $\mathcal{L}$ such that there is no sequence $(\psi_i : M_i \to^* M_{i+1})_{i\in\mathbb{N}}$ of non-identity 2-arrows of $\mathcal{R}$ with $M_0 = M$.

A *weak dynamic property* of $\mathcal{L}$ consists of a triple $(\mathcal{R},\mathcal{N},\mathrm{Ctxt})$ such that

- $\mathcal{R}$ is a set of 2-arrows of $\mathcal{L}$;
- $\mathcal{N}$ is a set of multimorphisms of $\mathcal{L}$;
- Ctxt is a set of functions on multimorphisms of $\mathcal{L}$.

We write $\mathsf{C}\{M\}$ for the multimorphism of $\mathcal{L}$ resulting from the application of $\mathsf{C} \in \mathrm{Ctxt}$ to a multimorphism M. Given such a triple, we let $\mathcal{W}(\mathcal{R},\mathcal{N},\mathrm{Ctxt})$ be the set of all multimorphisms M of $\mathcal{L}$ such that there exist $\mathsf{C} \in \mathrm{Ctxt}$, $N \in \mathcal{N}$ and $\psi : \mathsf{C}\{M\} \to^* N$ in $\mathcal{R}$.

The intuition between strong and weak dynamic properties is that they express some kind of strong or weak normalization. For what concerns the first, this is clear from the definition: $\mathcal{R}$ represents a notion of reduction and $\mathcal{S}(\mathcal{R})$ is the set of strongly $\mathcal{R}$-normalizing terms (seen as multimorphisms). For what concerns the latter, $\mathcal{R}$ is still a notion of reduction, $\mathcal{N}$ represents a notion of normal form and Ctxt a notion of "legal" contexts, so $\mathcal{W}(\mathcal{R},\mathcal{N},\mathrm{Ctxt})$ is the set of terms having a $\mathcal{N}$-form reachable via a reduction in $\mathcal{R}$, modulo an initial manipulation in Ctxt.

Definition 4.5 (faithful reduction, full expansion). Let $\mathcal{D} \hookrightarrow \mathbf{Poly}$ and $\mathbf{G} : \mathcal{L} \to \Lambda_!$. Let $\mathcal{R}$ be a strong dynamic property for $\mathcal{L}$. We say that the pair $(\mathcal{D},\mathbf{G})$ is *faithfully reductive* with respect to $\mathcal{R}$ if, for all $\rho \in \mathcal{R}$ and for all types Γ, A

subject reduction: $\mathbf{p}[\mathcal{D},\mathbf{G}]$ oplifts every 2-arrow of $\mathcal{R}$;
faithfulness: if $(\mathrm{Apx}[\mathcal{D}]\circ\mathbf{G})\,(\rho)_{\Gamma;A}\neq\emptyset$ and is the identity (*i.e.*, the diagonal relation), then ρ is an identity.

Let $(\mathcal{R},\mathcal{N},\mathrm{Ctxt})$ be a weak dynamic property for $\mathcal{L}$. We say that the pair $(\mathcal{D},\mathbf{G})$ is *fully expansive* if, for all $\rho\in\mathcal{R}$ and for all types Γ,A

subject expansion: $\mathbf{p}[\mathcal{D},\mathbf{G}]$ lifts every arrow of $\mathcal{R}$;
fullness: for all $u\in\mathcal{N}$, u is $(\mathcal{D},\mathbf{G})$-typable and, for all $\mathrm{C}\in\mathrm{Ctxt}$, if t is $(\mathcal{D},\mathbf{G})$-typable, then so is $\mathrm{C}\{t\}$.

Lemma 4.6. *Let $\mathcal{D}\hookrightarrow\mathbf{Poly}$ and $\mathbf{G}:\mathcal{L}\to\Lambda_!$ and let $\mathcal{R}$ be a strong dynamic property for $\mathcal{L}$ with respect to which $(\mathcal{D},\mathbf{G})$ is faithfully reductive. Then, if $\psi\in\mathcal{R}$ is not an identity, there is at least one oplifting (ψ,τ) of ψ with respect to $\mathbf{p}[\mathcal{D},\mathbf{G}]$ such that τ is not an identity.*

Proof. Let $\psi:M\Rightarrow M'\in\mathcal{R}$ be a non-identity arrow with M in the image of $\mathbf{p}[\mathcal{D},\mathbf{G}]$, *i.e.*, there exist Γ,A and $\delta\in\mathcal{D}(\Gamma;A)$ such that $(M,\delta)\in\mathcal{E}[\mathcal{D},\mathbf{G}](\Gamma;A)$, which means that $\delta\in(\mathrm{Apx}[\mathcal{D}]\circ\mathbf{G})(M)(\Gamma;A)$. Now, by subject reduction, ψ has an oplifting, which by definition is of the form (ψ,τ) with $\tau:\delta\Rightarrow\delta'$ and $(\delta,\delta')\in(\mathrm{Apx}[\mathcal{D}]\circ\mathbf{G})(\psi)_{\Gamma;A}$. By faithfulness, we may choose $\delta'\neq\delta$ and, in particular, $\tau\neq\mathrm{id}_\delta$. □

We may now state and prove our main result:

Theorem 4.7. *Let $\mathcal{L}$ be a programming language, presented as a 2-operad. Let $\mathcal{R}$ and $(\mathcal{R}_0,\mathcal{N},\mathrm{Ctxt})$ be a strong and a weak dynamic property for $\mathcal{L}$ such that $\mathcal{S}(\mathcal{R})\subseteq\mathcal{W}(\mathcal{R}_0,\mathcal{N},\mathrm{Ctxt})$, and suppose that $(\mathcal{D}\hookrightarrow\mathbf{Poly},\mathbf{G}:\mathcal{L}\to\Lambda_!)$ is both faithfully reductive with respect to $\mathcal{R}$ and fully expansive with respect to $(\mathcal{R}_0,\mathcal{N},\mathrm{Ctxt})$. Then, for every program M of $\mathcal{L}$, the following are equivalent:*

(1) M is $(\mathcal{D},\mathbf{G})$-typable;
(2) $M\in\mathcal{S}(\mathcal{R})$;
(3) $M\in\mathcal{W}(\mathcal{R}_0,\mathcal{N},\mathrm{Ctxt})$.

Proof. (1) ⇒ (2): let M be $(\mathcal{D},\mathbf{G})$-typable and suppose, for the sake of absurdity, that there is a sequence $(\psi_i:M_i\Rightarrow M_{i+1})_{i\in\mathbb{N}}$ of non-identity 2-arrows of $\mathcal{R}$ with $M_0=M$. Since M is $(\mathcal{D},\mathbf{G})$-typable, there exist Γ,A and a derivation δ such that $(M,\delta)\in\mathcal{E}[\mathcal{D},\mathbf{G}](\Gamma;A)$. In particular, M is the image of (M,δ) via $\mathbf{p}[\mathcal{D},\mathbf{G}]_{\Gamma;A}$. Now, since $\psi_0\in\mathcal{R}$, this functor oplifts it by the subject reduction hypothesis, so there is $(\psi_0,\tau_0):(M,\delta)\Rightarrow(M_1,\delta_1)$ in $\mathcal{E}[\mathcal{D},\mathbf{G}](\Gamma;A)$. In particular, M_1 too is in the image of $\mathbf{p}[\mathcal{D},\mathbf{G}]_{\Gamma;A}$. Furthermore, by Lemma 4.6, we may suppose τ_1 not to be an identity. We may now re-apply the reasoning to M_1 and ψ_1, then to M_2 and ψ_2, and so on, obtaining a sequence $(\tau_i:\delta_i\Rightarrow\delta_{i+1})_{i\in\mathbb{N}}$ of non-identity reductions in $\mathcal{D}(\Gamma;A)$, with $\delta_0=\delta$. We have therefore shown that δ is not strongly normalizing, contradicting Proposition 3.1 (remember that $\mathcal{D}\hookrightarrow\mathbf{Poly}$).
(2) ⇒ (3): by hypothesis.
(3) ⇒ (1): suppose $M\in\mathcal{W}(\mathcal{R}_0,\mathcal{N},\mathrm{Ctxt})$, *i.e.*, there exist $\mathrm{C}\in\mathrm{Ctxt}$, $N\in\mathcal{N}$ and $\psi:\mathrm{C}\{M\}\Rightarrow N$ in $\mathcal{R}_0$. By fullness, there are types Γ,A and a derivation ε such that $(N,\varepsilon)\in\mathcal{E}[\mathcal{D},\mathbf{G}](\Gamma;A)$. In particular, N is the image of (N,ε) via $\mathbf{p}[\mathcal{D},\mathbf{G}]_{\Gamma;A}$. Now, since $\psi\in\mathcal{R}_0$, this functor lifts it by the subject expansion hypothesis, so there is $(\psi,\tau):(\mathrm{C}\{M\},\delta)\Rightarrow(N,\varepsilon)$, showing in particular that $\mathrm{C}\{M\}$ is $(\mathcal{D},\mathbf{G})$-typable, so we conclude by fullness. □

Before moving on to examples and applications, let us highlight the modularity of the proof:

- (3)⇒(1) uses only full expansiveness and follows a completely standard argument: we apply subject expansion to typability of normal forms;
- (1)⇒(2) uses only faithful reductiveness and is an abstraction of an argument due to Bucciarelli et al. [2003], who reduced the soundness of the intersection type system of Coppo et al. [1981] to the strong normalization of the simply-typed λ-calculus;

- hence, (3)⇒(2) holds independently the hypothesis $\mathcal{S}(\mathcal{R}) \subseteq \mathcal{W}(\mathcal{R}_0, \mathcal{N}, \mathrm{Ctxt})$, *i.e.*, of (2)⇒(3).

The last remark brings to light how, contrarily to common understanding, intersection type systems do not characterize dynamic properties *per se* but, rather, *relate* them to one another. That is, intersection types actually build a bridge from an "existential" property (weak normalization) to a "universal" property (strong normalization), and their apparent ability to characterize dynamic properties results from judiciously choosing such properties so that the "universal" implies (as is usually the case) the "existential", thus closing the circle. It would be interesting to pursue a more general approach in this direction.

Also note that we are not exploiting the full generality of Theorem 2.4: in Definition 4.1, we could add non-trivial arrows to $\mathcal{T}_!\mathcal{D}$, which would lead to non-trivial subtyping relations in the resulting intersection type system. Exploring this possibility is left to future work.

4.3 A Worked Out Example

We will now show that the multimorphisms of $\mathcal{E}[\mathbf{LinPoly}, \mathbf{G}_0]$ are isomorphic to the derivations of Gardner-de Carvalho's non-idempotent intersection type system of Fig. 1. According to the general description of $\mathcal{E}[\mathbf{LinPoly}, \mathbf{G}_0]$ given after Definition 4.3, its objects are either polyadic simple types (color l), or sequences of such (color c). Its derivations are pairs (M, δ) consisting of a λ-term M and a linear polyadic simply-typed derivation δ of $\Theta; \vdash t : A$ such that $t \sqsubset \mathbf{G}_0(M)$. We know that the context Θ is entirely polyadic because λ-terms only have variables of type v, and $\mathbf{G}_0(\mathsf{v}) = \mathsf{c}$. Moreover, we know that $t : \mathsf{l}$ because $\mathbf{G}_0$ has only terms of type t and $\mathbf{G}_0(\mathsf{t}) = \mathsf{l}$.

Let $\mathrm{fv}(M) \subseteq \{x^1, \dots, x^n\}$. If we make explicit the approximation context, *i.e.*, which free polyadic variables of t approximate which free variables of $\mathbf{G}_0(M)$ (which coincide with those of M), we get $\overline{x}^1 \sqsubset x^1, \dots, \overline{x}^n \sqsubset x^n \vdash t \sqsubset \mathbf{G}_0(M)$. This also means that $\Theta = \overline{x}^1 : \overline{C}^1, \dots, \overline{x}^n : \overline{C}^n$. Superposing approximation judgments and typing judgments (as we did in the introduction), we get judgments of the form $\overline{x}^1 \sqsubset x^1 : \overline{C}^1, \dots, \overline{x}^n \sqsubset x^n : \overline{C}^n \vdash t \sqsubset \mathbf{G}_0(M) : A$, where $\overline{x}^i \sqsubset x^i : \overline{C}^i$ abbreviates $x^i_1 \sqsubset x^i : C^i_1, \dots, x^i_{k_i} \sqsubset x^i : C^i_{k_i}$. Now, by inspecting Fig. 2 and Fig. 3, we see that linear approximations and linear derivations are both syntax-directed. Therefore, the structure of M guides the structure of δ, and we have only three possibilities:

$$\frac{}{x_0 \sqsubset x : A; \vdash x_0 \sqsubset \mathbf{G}_0(x) : A}\ \text{pvar} \qquad \frac{\dfrac{\dfrac{}{; a : \vec{A} \vdash a : \vec{A}}\ \text{var} \qquad \Gamma, \overline{x} \sqsubset x : \overline{A}; \vdash t \sqsubset \mathbf{G}_0(M) : B}{\Gamma; a : \vec{A} \vdash t[\langle \overline{x} \rangle := a] \sqsubset \mathbf{G}_0(M)[!x := a] : B}\ \text{let}}{\Gamma; \vdash \lambda a.t[\langle \overline{x} \rangle := a] \sqsubset \mathbf{G}_0(\lambda x.M) : \vec{A} \multimap B}\ \text{lam}$$

$$\frac{\Gamma; \vdash t \sqsubset \mathbf{G}_0(M) : \vec{A} \multimap B \qquad \dfrac{\Gamma_1; \vdash u_1 \sqsubset \mathbf{G}_0(N) : A_1 \quad \dots \quad \Gamma_n; \vdash u_n \sqsubset \mathbf{G}_0(N) : A_n}{\Gamma'_1, \dots, \Gamma'_n; \vdash \langle u_1, \dots, u_n \rangle \sqsubset !\mathbf{G}_0(N) : \vec{A}}\ \text{box}}{\Gamma, \Gamma'_1, \dots, \Gamma'_n; \vdash t\langle u_1, \dots, u_n \rangle \sqsubset \mathbf{G}_0(MN) : B}\ \text{app}$$

where $\vec{A} = \langle A_1, \dots, A_n \rangle$. If we only retain the purple decorations, forget the intermediate steps not typing terms of the form $\mathbf{G}_0(-)$ and if, in the context, we write $x : \langle A_1, \dots, A_n \rangle$ instead of $x_1 \sqsubset x : A_1, \dots, x_n \sqsubset x : A_n$, we obtain precisely Fig. 1: the lam_0 rule is a special case of the above in which there are zero approximations of x, and the perm rule results from th exchange rule on **LinPoly** contexts. To be exact, the system obtained is in Fig. 5, discarding the rules weak and cntr.

4.4 Applications

We will now present a series of interesting instances of Theorem 4.7. First, however, let us point out the intuition behind faithfulness (Definition 4.5). The idea is that not only do we want subject reduction, we also want it to reflect the computation being performed. Indeed, subject reduction

$$\frac{}{x : \langle A \rangle \vdash x : A}\ \text{var} \qquad \frac{\Gamma, x : \vec{A} \vdash M : B}{\Gamma \vdash \lambda x.M : \vec{A} \multimap B}\ \text{lam} \qquad \frac{\Gamma \vdash M : \langle A_1, \dots, A_n \rangle \multimap B \quad \dots \Delta_i \vdash N : A_i \dots}{\Gamma \cdot \Delta_1 \cdots \Delta_n \vdash MN : B}\ \text{app}$$

$$\frac{\Gamma, x : \vec{A} \vdash M : C}{\Gamma, x : \sigma(\vec{A}) \vdash M : C}\ \text{exch} \qquad \frac{\Gamma \vdash M : C}{\Gamma, x : \langle\rangle \vdash M : C}\ \underset{x \notin \Gamma}{\text{weak}_0} \qquad \frac{\Gamma, x : \vec{B} \vdash M : C}{\Gamma, x : \langle \vec{B}, A \rangle \vdash M : C}\ \text{weak} \qquad \frac{\Gamma, x : \langle \vec{B}, A, A \rangle \vdash M : C}{\Gamma, x : \langle \vec{B}, A \rangle \vdash M : C}\ \text{cntr}$$

Fig. 5. Cartesian intersection type system characterizing head normalization. In the app rule, $\Gamma \cdot \Delta$ is concatenation as in Fig. 1. Non-idempotent or relevant variants are obtained by removing the rule cntr or weak, respectively (but not weak_0). Gardner-de Carvalho's system (Fig. 1) is obtained by removing both.

may hold because, when $M \to M'$, the derivation δ typing M' is the same as that typing M. This may happen because the redex fired in M to obtain M' is contained, via the embedding in $\Lambda_!$, inside a box approximated by $\langle\rangle$, hence it is not typed by δ, and the modifications induced by the reduction are invisible to the typing. Faithfulness says that this must *not* happen for the reductions of which we are trying to capture termination (*i.e.*, those in $\mathcal{R}$).

Head normalization. Consider again the pair $(\mathbf{LinPoly}, \mathbf{G}_0)$. We know that the induced type system is isomorphic to Fig. 1. It is easy to see that such a system enjoys both subject reduction and subject expansion with respect to *every* reduction. However, it is *faithfully* reductive only with respect to *head* reduction, which we denote here by $\mathcal{R}$. This is a consequence of what observed above: head redexes never appear under a $!(-)$ via $\mathbf{G}_0$, so they may never be "forgotten" by approximations.

Let now $\mathcal{R}_0 :=$ all reductions, $\mathcal{N} :=$ head normal forms and $\mathrm{Ctxt} := \{\mathrm{id}\}$. Then, $\mathcal{W}(\mathcal{R}_0, \mathcal{N}, \mathrm{Ctxt})$ is the set of terms having a head normal form. It is straightforward to see that head normal forms are typable (just assign a type of shape $\langle\rangle \multimap \cdots \multimap \langle\rangle \multimap \alpha$ to the head variable), so $(\mathbf{LinPoly}, \mathbf{G}_0)$ is fully expansive. It is obviously the case that $\mathcal{S}(\mathcal{R}) \subseteq \mathcal{W}(\mathcal{R}_0, \mathcal{N}, \mathrm{Ctxt})$ (if head reduction terminates for M, then M certainly has a head normal form). Therefore, by Theorem 4.7, $(\mathbf{LinPoly}, \mathbf{G}_0)$-typability characterizes having a head normal form and, moreover, we get for free that having a head normal form is the same as saying that head reduction terminates. This latter fact is not immediate: syntactic proofs require a form of standardization.

More generally, it is easy to see that $(\mathcal{D}, \mathbf{G}_0)$ for any $\mathcal{D} \in \{\mathbf{LinPoly}, \mathbf{AffPoly}, \mathbf{RelvPoly}, \mathbf{CartPoly}\}$ induces a type system characterizing head normalization. The least restrictive, when $\mathcal{D} = \mathbf{CartPoly}$, is depicted in Fig. 5. This is just a (strict, in the sense of van Bakel [1995]) reformulation of the standard system called $D\Omega$ by Krivine [1993], the variant with Ω of Coppo et al. [1981]. The other systems are obtained by discarding one or both of the rules weak and cntr: the former enables basic subtyping (*i.e.*, $A \wedge B \leq A$) and makes the system non-relevant; the latter makes the system idempotent. The rule weak_0 *cannot* be discarded, it is necessary to make the system complete (*e.g.* it is needed to type $\lambda x.y$).

Solvability. The above systems actually characterize solvability. It is a classic result of Wadsworth [1971] that solvability and head normalization coincide, so this would be a trivial remark if it were not for the fact that we may prove it *independently* of Wadsworth's result, thus yielding an alternative, type-theoretic proof of his theorem. Indeed, let $\mathcal{R}$ be head reductions, as above, and let $\mathcal{R}_0 :=$ all reductions, $\mathcal{N} := \{I\}$ and $\mathrm{Ctxt} :=$ applicative contexts, where I is the identity λ-term and applicative contexts are of the form $\{\cdot\}N_1 \cdots N_n$. By definition, $\mathcal{W}(\mathcal{R}_0, \mathcal{N}, \mathrm{Ctxt})$ is the set of solvable λ-terms.

Now, it is immediate that $\mathcal{S}(\mathcal{R}) \subseteq \mathcal{W}(\mathcal{R}_0, \mathcal{N}, \mathrm{Ctxt})$ (if a λ-term has a head normal form, it is solvable). We know that any of the systems of Fig. 5 is faithfully reductive with respect to $\mathcal{R}$. It is

easy to see that it is also fully expansive with respect to $(\mathcal{R}_0, \mathcal{N}, \mathrm{Ctxt})$: we know we have subject expansion, I is obviously typable and it is immediate to see that, if $MN_1 \cdots N_n$ is typable, then so must be M. So Theorem 4.7 applies, and we have that a term is solvable iff it has a head normal form iff it is typable in one of the systems of Fig. 5. The possibility of using intersection type systems to give an alternative proof of Wadsworth's result was already pointed out by Bucciarelli, Kesner and Ronchi Della Rocca [Bucciarelli et al. 2014] (specifically, they used Gardner-de Carvalho's system).

Note that this particular application uses a non-trivial set of functions Ctxt. In the sequel, we will always use Ctxt = {id} (the identity function), so we will never specify it again, and we will write $\mathcal{W}(\mathcal{R}, \mathcal{N})$ for $\mathcal{W}(\mathcal{R}, \mathcal{N}, \{\mathrm{id}\})$.

Strong normalization. Let now $\mathcal{D} \in \{\mathbf{CartPoly}, \mathbf{AffPoly}\}$ and let $\mathcal{D}_{\mathrm{sn}}$ be its full suboperad on the $\langle\rangle$-free types, *i.e.*, the empty sequence $\langle\rangle$ is not allowed, and consider the pair $(\mathcal{D}_{\mathrm{sn}}, \mathbf{G}_0)$. The resulting type system is obtained from Fig. 5 by disallowing the use of $\langle\rangle$ in types, which means that the weak$_0$ rule must be modified to derive $\Gamma, x : \langle A\rangle \vdash M : C$ from $\Gamma \vdash M : C$ (also, in the affine case, the rule cntr must be dropped). The idempotent system (*i.e.*, with contraction) is just a reformulation of the system originally introduced by Coppo et al. [1981].

Since the type $\langle\rangle$ is the only way to type the term $\langle\rangle$, empty polyadic approximations are not allowed in $\mathcal{D}_{\mathrm{sn}}$; in particular, whenever $t \sqsubset \mathbf{G}_0(M)$ with t typable in $\mathcal{D}_{\mathrm{sn}}$, no redex of M may be "forgotten" by t. This means that the identity reduction cannot approximate a non-identity reduction, ensuring faithfulness with respect to all reductions (subject reduction is easy to show).

It is also easy to show that all normal forms are typable. However, subject expansion only holds for *non-erasing* reductions. In the λ-calculus, a reduction step firing a redex $(\lambda x.M)N$ is *non-erasing* if $x \notin \mathrm{fv}(M)$ implies N normal.

Indeed, if $(\lambda x.M)N \to M$ because $x \notin \mathrm{fv}(M)$, and if $t \sqsubset M$ is a typable approximation, we cannot use the approximation $(\lambda a.t[\langle\rangle := a])\langle\rangle \sqsubset \mathbf{G}_0((\lambda x.M)N)$ in order to expand, because this is not typable in $\mathcal{D}_{\mathrm{sn}}$. However, we know that N is normal, hence typable, hence there exists $u \sqsubset N$ typable in $\mathcal{D}_{\mathrm{sn}}$, so we may use the approximation $(\lambda a.t[\langle z\rangle := a])\langle u\rangle$, where z does not appear in t and may be given the type of u. This latter point shows the necessity of weakening.

So, if $\mathcal{R}$ denotes the set of all reductions, $\mathcal{R}_0$ the set of non-erasing reductions and $\mathcal{N}$ the set of normal forms, we have that $\mathcal{S}(\mathcal{R})$ is the set of strongly normalizable λ-terms, whereas $\mathcal{W}(\mathcal{R}_0, \mathcal{N})$ is the set of λ-terms having a normal form via a non-erasing reduction. Since being strongly normalizable implies having a normal form under any kind of reduction, Theorem 4.7 applies and we have that a λ-term is strongly normalizable iff its normal form may be found by non-erasing reduction iff it is $(\mathcal{D}_{\mathrm{sn}}, \mathbf{G}_0)$-typable. This subsumes the classic result that the *perpetual strategy* (which is a particular non-erasing strategy) terminates on M iff M is strongly normalizable [Barendregt 1984].

Weak normalization. Let $\mathcal{D}_{\mathrm{wn}}$ be the suboperad of $\mathcal{D} \in \{\mathbf{LinPoly}, \mathbf{AffPoly}, \mathbf{RelvPoly}, \mathbf{CartPoly}\}$ in which typing judgments $\Theta; \Gamma \vdash t : A$ are restricted so that A (resp. a type in Θ, Γ) may only contain occurrences of $\langle\rangle$ in negative (resp. positive) position, and consider the pair $(\mathcal{D}_{\mathrm{wn}}, \mathbf{G}_0)$. The corresponding type systems are obtained from Fig. 5 simply by restrict typing judgments.

In this case, subject reduction and expansion are unproblematic for all reductions. It is also easy to show that all normal forms are typable, so the pair is fully expansive with respect to $(\mathcal{R}_0, \mathcal{N})$ where $\mathcal{R}_0$ is all reductions and $\mathcal{N}$ are the normal forms. Note that $\mathcal{W}(\mathcal{R}_0, \mathcal{N})$ is just the set of (weakly) normalizable λ-terms.

This time, what fails in general is faithfulness. Indeed, it holds for reductions which only fire redexes whose applicative depth is minimal among all redexes, the *applicative depth* of a subterm N of M being the number of times one must cross the argument position of an application to reach N from the root of the syntactic tree of M. Such reductions are called *Böhm reductions*, because they

$$\frac{}{x : A \vdash x : A}\ \text{var} \qquad \frac{\Gamma_1, x : A_1 \vdash N : B_1 \quad \dots \quad \Gamma_n, x : A_n \vdash N : B_n}{\Gamma_1 \cdots \Gamma_n \vdash \lambda x.N : \langle A_1 \multimap B_1, \dots, A_n \multimap B_n \rangle}\ \text{lam} \qquad \frac{\Gamma \vdash M : \langle A \multimap B \rangle \quad \Delta \vdash N : A}{\Gamma \cdot \Delta \vdash MN : B}\ \text{app}$$

Fig. 6. Intersection types for the call-by-value λ-calculus. The are also rules exch, weak$_0$, weak and cntr, not shown because identical to Fig. 5.

iterate head reduction and gradually reveal the Böhm tree of a term: the head redex is at minimum depth; once the head variable is found, one starts entering in its arguments, and so on.

The intuition behind faithfulness for Böhm reductions is the following. Let $\psi : M \to M'$ be a non-identity reduction such that $H\psi$ is a Böhm reduction and suppose that h is a polyadic term typable in $\mathcal{D}_{\mathrm{wn}}$ such that $h\langle\rangle \sqsubset \mathbf{G}_0(H\psi)$, contradicting faithfulness. Since $H\psi$ is a Böhm reduction, H must be of the form $xN_1 \cdots N_k$ (if instead of x we had an abstraction, there would be a redex at strictly lower applicative depth). But then the type of x must be of the form $\vec{C}_1 \multimap \cdots \multimap \vec{C}_k \multimap \langle\rangle \multimap A$ with $\langle\rangle$ appearing negatively, which is not allowed in a context of $\mathcal{D}_{\mathrm{wn}}$.

So, if we take $\mathcal{R}$ to be Böhm reductions, we have that $\mathcal{S}(\mathcal{R})$ is obviously contained in the set of normalizable terms, and Theorem 4.7 gives us that a term is (weakly) normalizable iff every Böhm reduction starting from it terminates iff it is $(\mathcal{D}_{\mathrm{wn}}, \mathbf{G}_0)$-typable. A syntactic proof of the first double implication requires a non-trivial standardization theorem [Barendregt 1984].

Non-strict intersection types. Systems using "non-strict" intersection types (*i.e.*, with intersection also to the right of arrows, as in $A \to A \wedge A$) may be obtained by considering the unrestricted encoding $\mathbf{G}_{\mathrm{k}} : \Lambda_{\mathrm{k}} \longrightarrow \Lambda_!$, which takes as source the full bichromatic presentation of the λ-calculus, with terms of type v as well. In this embedding, the image of a λ-term M seen as value is $!\mathbf{G}_0(M)$, so its approximations will all be of the form $\langle t_1, \dots, t_1 \rangle$ with $t_i \sqsubset \mathbf{G}_0(M)$. In other words, we have systems with the rules of Fig. 5 augmented with

$$\frac{\Gamma_1 \vdash M : A_1 \quad \dots \quad \Gamma_n \vdash M : A_n}{\Gamma_1, \dots, \Gamma_n \vdash M : \langle A_1, \dots, A_n \rangle}\ \text{inter}$$

In particular, the derivations of $\mathcal{E}[\mathcal{D}_{\mathrm{sn}}, \mathbf{G}_{\mathrm{k}}]$, with $\mathcal{D}_{\mathrm{sn}}$ as defined above (the cartesian version), are just a different presentation of the original system of Coppo and Dezani-Ciancaglini [1980].

The call-by-value λ-calculus. Our construction has two parameters: a sub-operad $\mathcal{D} \hookrightarrow \mathbf{CartPoly}$ and an embedding $\mathbf{G} : \mathcal{L} \to \Lambda_!$ of a calculus $\mathcal{L}$ in linear logic. The first wave of examples fixed $\mathbf{G}$ and showed several possibilities for $\mathcal{D}$; the last example provided a different $\mathbf{G}$, but the calculus was the same. Let us give an instance of our framework with a slightly different calculus.

In Sect. 3.3, we recalled Girard's call-by-value embedding $\mathbf{G}_{\mathrm{v}} : \Lambda_{\mathrm{v}} \longrightarrow \Lambda_!$. If $\mathcal{R}$ denotes head reduction in Λ_{v} (also known as the *weak call-by-value* strategy, which is the standard evaluation strategy of many practical programming languages), and if $\mathcal{R}_0$ denotes all reductions and $\mathcal{N}$ weak head normal forms (*i.e.*, arbitrary abstractions or terms of the form $xN_1 \cdots N_k$ with N_i arbitrary), then the pair $(\mathcal{D}, \mathbf{G}_{\mathrm{v}})$ for any $\mathcal{D} \in \{\mathbf{LinPoly}, \mathbf{AffPoly}, \mathbf{RelvPoly}, \mathbf{CartPoly}\}$ is faithfully reductive with respect to $\mathcal{R}$ and fully expansive with respect to $(\mathcal{R}_0, \mathcal{N})$, so Theorem 4.7 proves that weak head normal forms are reachable with the weak call-by-value strategy iff they are reachable at all, and gives us intersection type systems characterizing weak call-by-value normalization.

The types for these systems are given by $A, B ::= \langle A_1 \multimap B_1, \dots, A_n \multimap B_n \rangle$ ($n = 0$ 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 $D = !(D \multimap D)$, and remembering that $\langle - \rangle$ approximates $!(-)$. The rules are given in Fig. 6. As usual, one has four versions of the system, idempotent or not, relevant or not, by keeping or discarding weak and cntr.

$$\frac{}{x : \langle A \rangle \vdash x : A}\ \text{var} \qquad \frac{\Gamma, x : \Theta \vdash M : B}{\Gamma \vdash \lambda x.M : \langle\!\langle \Theta \multimap B \rangle\!\rangle}\ \text{lam} \qquad \frac{\Gamma, \alpha : A, \beta : B \vdash M : C}{\Gamma, \alpha : C \cdot A \vdash \mu\beta.\lceil\alpha\rceil M : B}\ \text{name}$$

$$\frac{\Gamma \vdash M : \langle\!\langle \Theta_1 \multimap B_1, \dots, \Theta_n \multimap B_n \rangle\!\rangle \quad \Delta_i^j \vdash N : \Theta_i(j)}{\Gamma \cdot \Delta_1^1 \cdots \Delta_1^{k_1} \cdots \Delta_n^1 \cdots \Delta_n^{k_n} \vdash MN : \langle\!\langle B_1, \dots, B_n \rangle\!\rangle}\ \text{app} \qquad \begin{array}{l} 1 \le i \le n \\ 1 \le j \le k_i \end{array}$$

Fig. 7. Intersection types for the $\lambda\mu$-calculus. The notation $\Gamma \cdot \Delta$ is concatenation as in Fig. 1, extended to sequences of the form $\langle\!\langle - \rangle\!\rangle$ as well. There are also rules exch, weak_0, weak and cntr, similar to Fig. 5.

The $\lambda\mu$-calculus. The 2-operad $\Lambda_!$ and the polyadic calculi that approximate it are not the most general for which one may define an approximation presheaf as in Sect. 4.1. In fact, it is possible to reformulate the whole theory of polyadic approximations (Sect. 3.4) using proof nets, which are the most general syntax for linear logic proofs, without intuitionistic restrictions. The details of this have been developed by the second author in his Ph.D. thesis [Pellissier 2017]. The development is based on *cyclic 2-operads*, of which the proof net syntax is a paradigmatic example, which we denote by **LL**.

In this augmented set-up, we may consider the $\lambda\mu$-calculus of Parigot [1992], a well-known extension of the λ-calculus capturing classical reasoning/control operators. This admits a natural presentation as a cyclic 2-operad ΛM, which may be embedded in **LL** by means of an embedding $\mathbf{L} : \Lambda M \longrightarrow \mathbf{LL}$ due to Laurent [2003]. Then, for any "flavor" $\mathcal{D}$ of simply-typed polyadic proof nets (with weakening and/or contraction), the Grothendieck construction applied to $\mathrm{Apx}[\mathcal{D}] \circ \mathbf{L}$ gives us a type system for the $\lambda\mu$-calculus. Its types, ranged over by A, B, C, are as follows:

$$A, B, C ::= \langle\!\langle P_1, \dots, P_n \rangle\!\rangle, \qquad P ::= \Theta \multimap B, \qquad \Theta ::= \langle A_1, \dots, A_n \rangle,$$

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 $D = ?(!D \multimap D)$, keeping in mind that $\langle - \rangle$ approximates $!(-)$ and $\langle\!\langle - \rangle\!\rangle$ approximates $?(-)$. In the literature on intersection types for the $\lambda\mu$-calculus, these latter are known as *union types* [Laurent 2004]. Given a sequence $\Theta = \langle A_1, \dots, A_n \rangle$, we write $\Theta(i)$ for A_i. Given two types A, B, which are always sequences, we write $A \cdot B$ for their concatenation. The typing judgments are of the form

$$x_1 : \Theta_1, \dots, x_m : \Theta_m, \alpha_1 : B_1, \dots, \alpha_n : B_n \vdash M : A,$$

where x_i are λ-variables and α_j are μ-variables. The typing rules are given in Fig. 7.

One may check that the right conditions for applying (the augmented version of) Theorem 4.7 are met by this system with respect to head reduction of $\lambda\mu$-terms, which is therefore characterized by typability. Interestingly, the linear version of this system (without weak and cntr) turns out to coincide with a system for the $\lambda\mu$-calculus recently introduced by Kesner and Vial [2017].

5 CONCLUSIONS

5.1 Related work

It is has been known for a long time that intersection types and linear logic are related. Regnier (crediting Duquesne) started this line of investigation in his Ph.D. thesis [Regnier 1992]; later on, Kfoury [2000] discovered an intriguing relationship between intersection types and a linearization of the λ-calculus. Retrospectively, his is a sort of an embryo of our linear polyadic calculus. We already mentioned the line of work concerning the link between non-idempotency and linearity [de Carvalho 2009; Neergaard and Mairson 2004]. However, to the best of our knowledge no

previous work described the correspondence between approximations and intersection types in the sharp, synthetic and broad framework we propose.

The affinities between our approximations and the Taylor expansion of Ehrhard and Regnier [2008] are obvious and we believe that our work may be entirely reformulated in that context. As a starting point, it is immediate to define an embedding $(-)^\circ$ of the linear polyadic calculus into the resource λ-calculus such that, for T in $\Lambda_!$ and t linear, $t \sqsubset T$ iff $t^\circ \in \mathrm{Taylor}(T)$, where $\mathrm{Taylor}(T)$ is the support of the Taylor expansion of T.

We already mentioned the work of Bucciarelli et al. [2003]. Here, we (vastly) generalize their approach, reducing the soundness of every well-known system of intersection types to the strong normalization of propositional linear logic (Proposition 3.1) and replacing plenty of *ad hoc* reducibility/logical relation arguments [Krivine 1993] or, in the non-idempotent case, *ad hoc* combinatorial arguments [Bernadet and Lengrand 2013; Kesner and Vial 2017; Kfoury 2000] with a uniform proof.

Finally, the work of Melliès and Zeilberger [2015] obviously had a major influence on our paper. It is worth mentioning that the fibrational perspective on type systems is already explored to some depth in their work, although it has a different meaning there, because they focus on the **Set**-enriched case and do not consider subject reduction/expansion.

5.2 Discussion and perspectives

Pragmatically, the message of this paper is: *as soon as a programming language may be meaningfully embedded in linear logic, there is an intersection type discipline for it.* Of course, what this intersection type discipline may do depends on the embedding and its "meaningfulness". However, we hope that we have shown enough applications to justify the claim that our approach is rather broad.

Our framework also gives a systematic explanation to certain aspects of intersection types:

- non-idempotency and relevance are just reflections of the absence of structural rules (weakening and contraction, respectively) in polyadic systems;
- strictness of intersections arises from the asymmetric version of Girard's encoding;
- as mentioned above, soundness is ultimately a consequence of strong normalization of propositional linear logic.

We would also like to stress that our construction does not have only an "explanatory" power but also has a "predictive" value: the last two instances given in Sect. 4.4, albeit simple, exemplify how our framework may be used to almost automatically synthesize intersection type systems without necessarily knowing them in advance (the call-by-value system is possibly folklore, but we did not know of it; the system for $\lambda\mu$ had been developed independently of our work). The encoding of the π-calculus in proof nets by Ehrhard and Laurent [2010] offers here an intriguing perspective.

Of course, we should also mention what our construction *cannot* currently do: it fails to capture infinitary systems such as those of the third author [Vial 2017], or probabilistic systems as those being developed by Breuvart and Dal Lago [2016]; and, although we do capture the reformulation by Grellois and Melliès [2015] of the system of Kobayashi and Ong [2009] for higher order recursion schemes, its main property is not a consequence of our Theorem 4.7. The question of capturing these systems and their properties is left for future work.

ACKNOWLEDGMENTS

This work was partially supported by ANR grant Elica (ANR-14-CE25-0005).

REFERENCES

Beniamino Accattoli. 2012. An Abstract Factorization Theorem for Explicit Substitutions. In *Proceedings of RTA*. 6–21.
Henk P. Barendregt. 1984. *The Lambda Calculus, Its Syntax and Semantics.* Elsevier.

Alexis Bernadet and Stéphane Lengrand. 2013. Non-idempotent intersection types and strong normalisation. *Logical Methods in Computer Science* 9, 4 (2013).

Gérard Boudol. 1993. The Lambda-Calculus with Multiplicities (Abstract). In *Proceedings of CONCUR*. 1–6.

Flavien Breuvart and Ugo Dal Lago. 2016. ntersection Types and Probabilistic Lambda Calculi. In *Presented at ITRS*.

Antonio Bucciarelli, Delia Kesner, and Simona Ronchi Della Rocca. 2014. The Inhabitation Problem for Non-idempotent Intersection Types. In *Proceedings of IFIP TCS*. 341–354.

Antonio Bucciarelli, Adolfo Piperno, and Ivano Salvo. 2003. Intersection Types and lambda-Definability. *Mathematical Structures in Computer Science* 13, 1 (2003), 15–53.

Sébastien Carlier, Jeff Polakow, J. B. Wells, and A. J. Kfoury. 2004. System E: Expansion Variables for Flexible Typing with Linear and Non-linear Types and Intersection Types. In *Proceedings of ESOP 2004*. 294–309.

François Conduché. 1972. Au sujet de l'existence d'adjoints à droite aux foncteurs 'image reciproque' dans la catégorie des catégories. *C. R. Acad. Sci. Paris* Série A, 275 (1972), 891–894.

Mario Coppo and Mariangiola Dezani-Ciancaglini. 1980. An extension of the basic functionality theory for the λ-calculus. *Notre Dame Journal of Formal Logic* 21, 4 (1980), 685–693.

Mario Coppo, Mariangiola Dezani-Ciancaglini, and Betti Venneri. 1981. Functional Characters of Solvable Terms. *Math. Log. Q.* 27, 2-6 (1981), 45–58.

Pierre-Louis Curien and Jovana Obradovic. 2017. *Categorified cyclic operads*. Technical Report 1706.06788 [math.CT]. ArXiv.

Daniel de Carvalho. 2009. Execution Time of lambda-Terms via Denotational Semantics and Intersection Types. *CoRR* abs/0905.4251 (2009).

Thomas Ehrhard and Olivier Laurent. 2010. Interpreting a Finitary Pi-Calculus in Differential Interaction Nets. *Information and Computation* 208, 6 (2010), 606–633.

Thomas Ehrhard and Laurent Regnier. 2008. Uniformity and the Taylor expansion of ordinary lambda-terms. *Theor. Comput. Sci.* 403, 2–3 (2008), 347–372.

Marcelo P. Fiore, Gordon D. Plotkin, and Daniele Turi. 1999. Abstract Syntax and Variable Binding. In *Proceedings of LICS*. 193–202.

Philippa Gardner. 1994. Discovering Needed Reductions Using Type Theory. In *Proceedings of TACS*. 555–574.

Jean-Yves Girard. 1987. Linear Logic. *Theor. Comput. Sci.* 50, 1 (1987), 1–102.

Charles Grellois and Paul-André Melliès. 2015. Relational Semantics of Linear Logic and Higher-order Model Checking. In *Proceedings of CSL*. 260–276.

Barney P. Hilken. 1996. Towards a Proof Theory of Rewriting: The Simply Typed 2lambda-Calculus. *Theor. Comput. Sci.* 170, 1-2 (1996), 407–444.

Tom Hirschowitz. 2013. Cartesian closed 2-categories and permutation equivalence in higher-order rewriting. *Logical Methods in Computer Science* 9, 3 (2013).

Martin Hyland. 2017. Classical lambda calculus in modern dress. *Math. Struct. Comput. Sci.* 27, 5 (2017), 762–781.

Delia Kesner and Pierre Vial. 2017. Types as Resources for Classical Natural Deduction. *Proceedings of FSCD (to appear)* (2017).

Assaf J. Kfoury. 2000. A linearization of the Lambda-calculus and consequences. *J. Log. Comput.* 10, 3 (2000), 411–436.

Naoki Kobayashi and C.-H. Luke Ong. 2009. A Type System Equivalent to the Modal Mu-Calculus Model Checking of Higher-Order Recursion Schemes. In *Proceedings of LICS*. 179–188.

Jean-Louis Krivine. 1993. *Lambda Calculus, Types and Models*. Ellis Horwood.

Olivier Laurent. 2003. Polarized proof-nets and lambda-mu-calculus. *Theor. Comput. Sci.* 290, 1 (2003), 161–188.

Olivier Laurent. 2004. On the denotational semantics of the untyped lambda-mu calculus. (2004). Unpublished note.

Tom Leinster. 2004. *Higher Operads, Higher Categories*. Cambridge University Press.

Damiano Mazza. 2012. An Infinitary Affine Lambda-Calculus Isomorphic to the Full Lambda-Calculus. In *Proceedings of LICS*. 471–480.

Paul-André Melliès, Nicolas Tabareau, and Christine Tasson. 2009. An Explicit Formula for the Free Exponential Modality of Linear Logic. In *Proc. ICALP 2009*. 247–260.

Paul-André Melliès and Noam Zeilberger. 2015. Functors are Type Refinement Systems. In *Proceedings of POPL*. 3–16.

Peter Møller Neergaard and Harry G. Mairson. 2004. Types, potency, and idempotency: why nonlinearity and amnesia make a type system work. In *Proceedings of ICFP*. 138–149.

Susan Niefield. 2004. Change of Base for Relational Variable Sets. *Theory Appl. Categ.* 12, 7 (2004), 248–261.

Michel Parigot. 1992. *$\lambda\mu$-Calculus: An algorithmic interpretation of classical natural deduction*. Springer Berlin Heidelberg, Berlin, Heidelberg, 190–201. https://doi.org/10.1007/BFb0013061

Luc Pellissier. 2017. *Réductions et approximations linéaires*. PhD. Thesis. Université Paris 13.

Laurent Regnier. 1992. *Lambda-calcul et réseaux*. PhD Thesis. Université Paris 7.

R. A. G. Seely. 1987. Modelling Computations: A 2-Categorical Framework. In *Proceedings of LICS*. 65–71.

Terese. 2003. *Term Rewriting Systems*. Cambridge Tracts in Theoretical Computer Science, Vol. 55. Cambridge University Press.

Steffen van Bakel. 1995. Intersection Type Assignment Systems. *Theor. Comput. Sci.* 151, 2 (1995), 385–435.

Pierre Vial. 2017. Infinitary intersection types as sequences: A new answer to Klop's problem. In *Proceedings of LICS*. 1–12.

Christopher P. Wadsworth. 1971. *Semantics and Pragmatics of the Lambda Calculus*. PhD Thesis. University of Oxford.