

# Linearization via Rewriting (Long Version)

Ugo Dal Lago

Dipartimento di Informatica-Scienza e Ingegneria  
Università di Bologna  
Bologna, Italy  
ugo.dallago@unibo.it

Federico Olimpieri

Dipartimento di Informatica-Scienza e Ingegneria  
Università di Bologna  
Bologna, Italy  
federico.olimpieri@unibo.it

**Abstract**—We introduce the *structural resource*  $\lambda$ -calculus, a new formalism in which strongly normalizing terms of the  $\lambda$ -calculus can naturally be represented, and at the same time any type derivation can be internally rewritten to its linearization. The calculus is shown to be normalizing and confluent. Noticeably, every strongly normalizable  $\lambda$ -term can be represented by a type derivation. This is the first example of a system where the linearization process takes place *internally*, while remaining purely *finitary* and *rewrite-based*.

## I. INTRODUCTION

Slightly less than a century since the pioneering contributions by Church and Gentzen, the study of proof theory, type systems and the  $\lambda$ -calculus has produced a multitude of logical and computational formalisms in which proofs and programs can be represented and the process of either eliminating cuts or evaluating a program by forms of rewriting can be studied directly within the system at hand, often giving rise to confluence and normalization results (see, e.g., [20], [28], [34], [36]). The considered systems are often very expressive, which is why the aforementioned normalization properties become logically non-trivial, and can hardly be carried out combinatorially.

Since the mid-eighties, the situation sketched above has somehow changed: the advent of linear logic [21] has allowed a finer structure to be attributed to the underlying computation process. By identifying *structural* logical rules, and in particular *contraction*, as the bottleneck in the normalization results, linear logic gave rise to the introduction of proof and type systems in which structural rules are *very severely restricted* or *not allowed at all*. As a consequence, normalization properties can be proved by purely combinatorial means: rewriting and cut-elimination have the effect of strictly reducing the size of the objects at hand. Systems of this kind, which we will call *quantitative* — simply to distinguish them from the somehow *qualitative* systems we referred to in the previous paragraph — include not only multiplicative linear logic [11], [21], but also non-idempotent intersection types [12], [19], the resource  $\lambda$ -calculus [8], [14], and certain type systems based on restrictions of exponentials such as the so-called light logics [22], [23], [27]. In quantitative systems, there is an aspect of finiteness absent in qualitative systems, which makes them particularly suitable to the characterization of complexity classes and in general whenever a fine control over the use of resources is considered necessary. A proof or program, in

quantitative systems, is an object that reveals *everything* about its dynamics, *i.e.*, about the evolution it will undergo as a result of the evaluation.

Since the early days of linear logic, there have been many attempts to relate quantitative and qualitative systems by studying how to transform proofs and programs in qualitative systems into equivalent objects in quantitative systems. Already in Girard’s original work [21] there is a trace of all this in the so-called *approximation theorem*, which states that every *cut-free* proof of multiplicative and exponential linear logic can be approximated, in a very natural sense, by a proof of multiplicative linear logic. We should also mention the line of work about *linearizing* the  $\lambda$ -calculus [1]–[3], [25], in which one studies to what extent pure  $\lambda$ -terms can be linearized into linear terms, establishing on one hand that this process can be applied to all normalizing terms following their reduction sequence, and that on the other hand a relationship exists between the emerging process and typeability in non-idempotent intersection types. We should also mention the crucial contribution of Ehrhard and Regnier [14], who through an inductive and very elegant definition, provide a notion of *Taylor expansion* for any pure  $\lambda$ -term as the (infinite) set of its linear approximants. Continuing along the same vein, we must certainly mention the results on the extensional collapse of quantitative denotational models into qualitative models [13]. The collapse induces a categorical understanding of the relationship between idempotent (*i.e.*, qualitative) and non-idempotent (*i.e.*, quantitative) intersection types. However, this result does not directly yield an *explicit* program transformation.

In all these attempts, the infinitary aspects of qualitative systems are either neglected by considering only cut-free proofs, or reflected in the target of the transformation, like in the Taylor expansion. In the case of Kfoury’s linearization [25], in particular, the process of computing the linear approximant of a pure  $\lambda$ -term is delegated to a notion of *expansion* that, even if formally defined in terms of rewriting, is trivially not normalizing, because it consists in increasing the size of the right argument bag of an application, thus matching the number of arguments required by the related abstraction. In a certain sense, therefore, the linearization process remains fundamentally infinitary, similar to what happens in the Taylor expansion [14]. The same holds for the remarkable recent contribution by Pautasso and Ronchi [35], which introduces

a quantitative version of the simple types in which derivations carry the same kind of quantitative information as in non-idempotent intersection types. The system turns out to be equivalent to the usual simply-typed  $\lambda$ -calculus in terms of typeability, but the proof of this fact is carried out externally to the system, and is nontrivial.

Could linearization be made an *integral part* of the underlying system? Is there a way to compute a linear version of a program (or proof) so that this transformation happens *within* the system? The authors are strongly convinced of the interest of these questions and think that understanding to what extent all this is possible can help shedding some light on the relationship between qualitative and quantitative systems.

The main contribution of the present paper is precisely to introduce a language and type system in which qualitative systems can be naturally embedded *and* an essentially quantitative fragment can be isolated. More specifically, terms typable in qualitative systems such as the simply-typed  $\lambda$ -calculus can be compositionally embedded into the calculus. Linearization, this is the real novelty, becomes a form of *rewriting* turning any term into a linear approximation of it. The role of this rewriting process, called exponential reduction, is to eliminate the use of structural rules *without* performing any  $\beta$ -reduction, the latter being part of the system itself. Crucially, exponential reduction is strongly normalizing.

The calculus thus obtained can be seen as an extension of Boudol's resource  $\lambda$ -calculus [8], and we call it the *structural resource  $\lambda$ -calculus*. Technically, the key ingredients in its definition are as follows. At the level of terms, there is both the possibility of having a nonlinear use of variables, but also of passing a bag of terms to an abstraction, in the style of the resource  $\lambda$ -calculus. Crucially, abstractions are labeled so as to identify the structural rules needed to fire the corresponding redex. An intersection type assignment system is an integral part of the calculus and guarantees the termination of each sequence of  $\beta$ -reduction steps. As already mentioned, the latter is not the only available reduction rule, and is complemented in an essential way by exponential reduction, giving rise to what we call *structural reduction*.

In addition to the definition of the structural resource  $\lambda$ -calculus, the key technical contributions of this paper are proofs of confluence and strong normalization for structural reduction. Remarkably, every strongly normalizable  $\lambda$ -term can be represented in a natural way by a structural resource term, the embedding being based on *idempotent* intersection types [6].

## II. A BIRD'S EYE VIEW ON LINEARIZATION AND INTERSECTION TYPES

The purpose of this section is to informally describe the process of linearization in the  $\lambda$ -calculus, anticipating some of the problems we have to face, at the same time introducing the non-specialist reader to the topic. We will proceed by analyzing a very simple example of a  $\lambda$ -term, which will also constitute a sort of running example in the rest of this manuscript.

Let us consider the  $\lambda$ -term  $MN$ , where

$$M := \lambda x.w(x(xy))(xy) \quad N := \lambda z.qzz$$

To the above term we can attribute simple types in the obvious way:

$$w : o \rightarrow o \rightarrow o, y : o, q : o \rightarrow o \rightarrow o \vdash MN : o \quad (1)$$

The term  $MN$  is not a linear term at all, however, in particular due to the nonlinear use of the variables  $x$  and  $z$  by the terms  $M$  and  $N$ , respectively. Understanding the quantitative aspects of the dynamics of the term looking at its typing is not immediate, i.e., it is not possible in a direct way. By  $\beta$ -reducing the term, however, one realizes quite easily that the variable  $y$ , although occurring free only *twice* in the term, will occur free *six* times in its normal form. Depending on the use that  $w$  and  $q$  will make of their arguments, this number could grow or decrease.

Would it be possible to account for the calculations above *from within a type system*? The answer is positive: as is well-known, non-idempotent intersection types [12], [19] allow us to type the same term in the following way:

$$w : \langle o \rangle \multimap \langle o \rangle \multimap o, y : \langle o \rangle^6, q : \langle \langle o \rangle \multimap \langle o \rangle \multimap o \rangle^4 \vdash MN : o$$

where the sequence  $\langle a_1, \dots, a_k \rangle$  stands for the (commutative but not idempotent) intersection type  $a_1 \cap \dots \cap a_k$ , and  $\langle a \rangle^n$  stands for  $\langle a, \dots, a \rangle$ , where  $a$  appears exactly  $n$  times. Once a term is typed in non-idempotent intersection types, its type derivation tells us *everything* there is to know about the dynamics of the term, including its length. If we want the size of the term itself to reflect the size of the derivation, that is, for the former to become a unique and canonical description of the latter, we must move to the so-called resource terms, where the second argument of each application is not a single term, but a *bag* of terms. Resource terms correspond to *linear approximations* of ordinary  $\lambda$ -terms. In our example, we obtain the resource terms  $s := \lambda x.w\langle x\langle y, y \rangle, x\langle y, y \rangle \rangle\langle x\langle y, y \rangle \rangle$  and  $t := \lambda z.q\langle z \rangle\langle z \rangle$ , getting the following typing for the application  $s\langle t \rangle^4$ :

$$w : \langle o \rangle \multimap \langle o \rangle \multimap o, y : \langle o \rangle^6, q : \langle \langle o \rangle \multimap \langle o \rangle \multimap o \rangle^4 \vdash s\langle t \rangle^4 : o \quad (2)$$

How can we compute well-behaved linear approximations (i.e., typed resource terms) of a given  $\lambda$ -term? If one is only interested in the term's semantics, and given the aforementioned connection between  $\lambda$ -term dynamics and its quantitative approximations, one possibility is to first compute the normal form of the term and then reconstruct the corresponding intersection type derivation. This can be seen as a rephrasing of Girard's *approximation theorem*. However, this answer is not completely satisfactory, since it relies on the evaluation of the program. We can thus restate our question as: how do we compute linear approximations without reducing (to  $\beta$ -normal form) the considered program?

Answering this question is the main goal of this paper. Our main contribution are precisely a language and type system in which *all* the above-mentioned type derivations can live and

where we can define an appropriate notion of rewriting which performs linearization. Below, we describe informally how the transition from (1) to (2) can occur *within* the system. First, let us follow a naive proof-search heuristic to compute the non-idempotent intersection typing of our term. In order to do so, we first *embed*  $M$  and  $N$  separately in the resource calculus as terms  $u = \lambda x.x\langle x\langle y \rangle \rangle \langle x\langle y \rangle \rangle$  and  $v = \lambda z.q\langle z \rangle \langle z \rangle$  with

$$\begin{aligned} w : \langle o \rangle \multimap \langle o \rangle \multimap o, y : \langle o \rangle^2 \vdash u : \langle \langle o \rangle \multimap o \rangle^3 \multimap o \\ q : \langle \langle o \rangle \multimap o \rangle \vdash v : \langle o \rangle^2 \multimap o \end{aligned}$$

Now, our proof search gets stuck, since the type of the argument  $v$  is not the one the function  $u$  requires. This happens because  $v$  records the presence of *two* occurrences of  $z$ , while  $u$  only demands *one* input. One possible solution would be to extend the unification algorithm for simple types to the intersection type setting, which is precisely what has been done in [35]. However, we follow a different path. The types  $\langle o \rangle^2 \multimap o$  and  $\langle o \rangle \multimap o$  are different but *coherent*, in the sense that their syntactic tree is the same up to the number of copies of types occurring inside intersections. In particular, the two are connected by a *nested* structural rule, which is defined on the contraction  $c_o : \langle o \rangle \rightarrow \langle o, o \rangle$ . We then extend the syntax and the typing of the resource calculus in order to account for (nested) structural rules. In particular, we shall allow the following type inference:

$$\frac{q : \langle \langle o \rangle \multimap o \rangle \vdash v : \langle o \rangle^2 \multimap o \quad c_o : \langle o \rangle \rightarrow \langle o, o \rangle}{q : \langle \langle o \rangle \multimap o \rangle \vdash p = \lambda z^{c_o}.q\langle z \rangle \langle z \rangle : \langle o \rangle \multimap o}$$

from which we can clearly infer the typing  $w : \langle o \rangle \multimap \langle o \rangle \multimap o, y : \langle o \rangle^2, q : \langle \langle o \rangle \multimap o \rangle \vdash u\langle p \rangle^3 : o$ . The term  $u\langle p \rangle^3$  is not a *linear* approximation of  $MN$ , since we are allowing arbitrary structural rules to be used on bound variables. However, we shall recover the linear resource term  $s\langle t \rangle^4$  as the *normal form* of  $u\langle p \rangle^3$  under a novel notion of labelled rewriting, following which we allow for the transformation

$$\lambda z^{c_o}.q\langle z \rangle \langle z \rangle : \langle o \rangle \multimap o \rightarrow_{c_o \multimap o} \lambda z.q\langle z \rangle \langle z \rangle : \langle o, o \rangle \multimap o.$$

Note that we index the reduction with type morphisms connecting the typing of the two involved terms. We exploit an appropriate notion of *transformation* on type derivations, that consists of an action of the nested structural rule  $c_o \multimap o$  on the term  $u$ , denoted as  $[c_o \multimap o]u$ . This transformation will track all the occurrences of the bound variable  $x$  in  $u$ , replacing their type  $\langle o \rangle \multimap o$  with  $\langle o, o \rangle \multimap o$ . In order to do so, the inputs of  $x$  are *duplicated* accordingly, obtaining the term  $s$ . The process consists then in a *deterministic* and confluent notion of *structural rule* elimination, which is performed only *on-demand*. It is worth noting that the rewriting does not perform any substitution. In particular, if  $s \rightarrow_f t$  the two terms  $s$  and  $t$  are coherent and, hence, are (generally non-linear) approximations of the same ordinary  $\lambda$ -term.

### III. SOME MATHEMATICAL PRELIMINARIES

Following [39], we see intersection types as *lists* of types. This will allow us to obtain a simple and elegant categorical

framework, where intersection plays the role of a (strict) tensor product, and morphisms are given by structural rules (*i.e.*, permutations, copying, and erasing). In order to properly establish the formal framework, we first need to introduce categories of integers, from which the list construction naturally arises.

#### A. Categories from Natural Numbers

We consider the category  $\mathbb{O}_f^c$  whose objects are natural numbers and in which  $\mathbb{O}_f^c(m, n) := [n]^{[m]}$ , where  $[n] = \{1, \dots, n\}$  for  $n \in \mathbb{N}$ . As a consequence,  $[0] = \emptyset$  (so the number 0 is the initial object,  $\emptyset$  being the unique map  $[m]^{[0]}$ ). We generally denote composition of morphisms as either  $g \circ f$  or  $gf$ . The category  $\mathbb{O}_f^c$  is symmetric *strict* monoidal (cocartesian in particular), with tensor product given by addition. We define the subsets of *symmetries* of  $\mathbb{O}_f^c(m, n)$  as  $\mathbb{O}_f^l(m, n) = \{\alpha \in \mathbb{O}_f^c(m, n) \mid \alpha \text{ is bijective}\}$ . Evidently,  $\mathbb{O}_f^l(m, n)$  determines a subcategory of  $\mathbb{O}_f^c$ , the category  $\mathbb{O}_f^l$  of *symmetries* (or *permutations*). In what follows, we work with the parametric category  $\mathbb{O}_f^\spadesuit$ , where  $\spadesuit \in \{c, l\}$ .

#### B. Lists

We often work with (ordered) lists whose elements are taken from a given set  $X$ . Such lists are ranged over by metavariables like  $\vec{a}, \vec{b}, \vec{c}, \dots$ , and one such list can also be written explicitly as  $\langle a_1, \dots, a_k \rangle$ , where  $k \in \mathbb{N}$  and  $a_1, \dots, a_k \in X$ . The *list concatenation* of  $\vec{a}$  and  $\vec{b}$  is indicated as  $\vec{a} \oplus \vec{b}$ . We write  $|\vec{a}|$  to denote the length of  $\vec{a}$ .

Given a list  $\vec{a} = \langle a_1, \dots, a_k \rangle$  and a function  $\alpha : [h] \rightarrow [k]$ , the expression  $\vec{a}^{[\alpha]}$  stands for the list of length  $h$  defined as  $\langle a_{\alpha(1)}, \dots, a_{\alpha(h)} \rangle$ . This turns  $\alpha$  into a contravariant semigroup action on  $\vec{a}$ : given  $\text{len}(\vec{a}) = n, \alpha : [m] \rightarrow [n]$  and  $\beta : [l] \rightarrow [m]$  we have that  $(\vec{a}^{[\alpha]})^{[\beta]} = \vec{a}^{[\alpha\beta]}$ .

Given a small category  $A$ , we define its  $\spadesuit$ -*monoidal completion*,  $!^\spadesuit(A)$  as follows:

- $\text{ob}(!^\spadesuit(A)) = \{\langle a_1, \dots, a_k \rangle \mid k \in \mathbb{N} \text{ and } a_i \in \text{ob}(A)\}$ .
- $!^\spadesuit(A)(\langle a_1, \dots, a_k \rangle, \langle b_1, \dots, b_l \rangle) = \{\langle \alpha; f_1, \dots, f_l \rangle \mid \alpha \in \mathbb{O}_f^\spadesuit(l, k) \text{ and } f_i \in A(a_{\alpha(i)}, b_i) \text{ whenever } i \in [l]\}$ .

The category  $!^l(A)$  is *symmetric* strict monoidal, while  $!^c(A)$  is *cartesian* strict monoidal. In both cases, the tensor product is given by list concatenation.

*Ground morphisms* are morphisms of the shape  $\langle \alpha; \text{id}, \dots, \text{id} \rangle$ , thus representing the composition of permutations, weakenings and contractions without nesting. Given a list  $\vec{a}$ , an integer  $k \in \mathbb{N}$  and a morphism  $\alpha \in \mathbb{O}_f^\spadesuit([|\vec{a}|], [k])$ , we define the *ground morphism* induced by  $\alpha$  as the morphism  $\langle \alpha; \vec{\text{id}} \rangle \in !^\spadesuit(A)(\vec{a}, \vec{a}^{[\alpha]})$ . We shall often abuse the notation and denote  $\langle \alpha; \vec{\text{id}} \rangle$  as just  $\alpha$ . The family  $\alpha_{\vec{a}}$  for  $\vec{a} \in !^\spadesuit(A)$  is, moreover, a *natural transformation*. We use  $\sigma, \tau, \dots$  for ground morphisms induced by permutations.

**Example 1.** We give some examples of morphisms in  $!^c(A)$ , where  $A = \{a, b, c\}$ . The following is a diagonal morphism (that we also call a *contraction*) over  $A$ :

$$c_a = \langle \alpha; \text{id}_a, \text{id}_a \rangle : \langle a \rangle \rightarrow \langle a, a \rangle$$

where  $\alpha$  is the only map from [2] to [1]. For all  $n \geq 1$  we have a  $n$ -ary contraction

$$c_a^n : \langle a \rangle \rightarrow \overbrace{\langle a, \dots, a \rangle}^{n \text{ times}}$$

where for  $n = 1$  we have that  $c^1 = \text{id}_a$ . Projections (that we also call *weakenings*) are given as follows:

$$\pi_{a_i} = \langle \alpha; \text{id}_{a_i} \rangle : \langle a_1, a_2 \rangle \rightarrow \langle a_i \rangle$$

where  $\alpha : [1] \rightarrow [2]$  maps 1 to 1. We denote as  $T_{\vec{a}} : \vec{a} \rightarrow \langle \rangle$  the terminal morphism.

Given sequences of lists  $\gamma = \vec{a}_1, \dots, \vec{a}_n$  and  $\delta = \vec{b}_1, \dots, \vec{b}_n$  we define their *tensor product* as pointwise list concatenation:  $\gamma \otimes \delta = \vec{a}_1 \oplus \vec{b}_1, \dots, \vec{a}_n \oplus \vec{b}_n$ . Typing contexts in our system will consist of sequences of lists. The tensor product between them will model context concatenation. The ground morphism induced by  $\gamma = \vec{a}_1, \dots, \vec{a}_n$  consists of the sequence of ground morphisms of  $\vec{a}_i$ . Abusing the notation, we will use greek letters to denote also ground morphisms induced by sequences.

**Remark 1.** Consider the morphism on lists  $\langle \alpha; \vec{f} \rangle = \langle \alpha; f_1, \dots, f_l \rangle : \vec{a} = \langle a_1, \dots, a_k \rangle \rightarrow \vec{b} = \langle b_1, \dots, b_l \rangle$ . By definition, we have that  $f_i : a_{\alpha(i)} \rightarrow b_i$  with  $i \in [l]$ . We remark that we can always factorize  $\langle \alpha; f_1, \dots, f_l \rangle$  as follows:

$$\langle \alpha; \vec{f} \rangle = \vec{f} \circ \alpha_{\vec{a}}$$

with  $\vec{f} : \vec{a}^{[\alpha]} \rightarrow \vec{b}$  and  $\alpha_{\vec{a}} : \vec{a} \rightarrow \vec{a}^{[\alpha]}$ , defined as  $\alpha_{\vec{a}} = \langle \alpha; \text{id}_{a_{\alpha(1)}}, \dots, \text{id}_{a_{\alpha(l)}} \rangle$  and  $\vec{f} = \langle \text{id}; f_1, \dots, f_l \rangle$ . We call  $\alpha_{\vec{a}}$  the *ground fragment* of  $\langle \alpha; \vec{f} \rangle$ , denoted as  $\text{ground}(\langle \alpha; \vec{f} \rangle)$  and  $\vec{f}$  the *nested fragment* of it, denoted as  $\text{nest}(\langle \alpha; \vec{f} \rangle)$ . We can extend the factorization to sequences of morphisms  $\theta = \langle \alpha_1; \vec{f}_1 \rangle, \dots, \langle \alpha_n; \vec{f}_n \rangle$  in the natural way.

### C. Labelled Transition Systems

In the rewriting systems we deal with in this paper, reduction steps are *labelled*. A *labelled reduction system* consists of a labelled transition system  $R = (\Lambda, \mathbb{S}, \rightarrow)$  whose labels form a partial monoid  $\mathbb{S} = (S, \cdot, 1)$ .<sup>1</sup> Given a labelled reduction system  $R = (\Lambda, (S, \cdot, 1), \rightarrow)$ , we abuse the notation by also writing  $s \rightarrow s'$  to denote the relation  $\rightarrow \subseteq \Lambda \times \Lambda$  s.t.  $s \rightarrow s'$  when there exists  $l \in S$  with  $s \rightarrow_l s'$ . Given a labelled reduction system  $R = (\Lambda, (S, \cdot, 1), \rightarrow)$ , we say that  $R$  is *normalizing* when, given  $s \in \Lambda$ , there exists a reduction sequence  $s = s_0 \rightarrow s_1 \rightarrow \dots \rightarrow s_n$  such that  $s_n \nrightarrow$ , and in this case we call  $s_n$  a *normal form* of  $s$ . We say that  $R$  is *strongly normalizing* (or *terminating*) when it is impossible to form an infinite reduction sequence  $s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow \dots$ .

**Definition 1** (Reflexive and Transitive Extension). Let  $R = (\Lambda, (S, \cdot, 1), \rightarrow)$  be a labelled reduction system. We can thus build its *transitive and reflexive extension*  $R^* =$

<sup>1</sup>We consider partial monoids because the labels of our rewriting systems will be morphisms in appropriate categories of types, where the monoidal product we want to consider is the composition of those morphisms.

$(\Lambda, (S, \cdot, 1), \rightarrow^*)$  where  $\rightarrow^*$  is defined inductively by the following rules:

$$\frac{s \rightarrow_h t \quad t \rightarrow_l^* v \quad l \cdot h \downarrow}{s \rightarrow_{l \cdot h}^* v} \quad \frac{}{s \rightarrow_1^* s}$$

**Definition 2** (Labelled Confluence). Let  $R = (\Lambda, (S, \cdot, 1), \rightarrow)$  be a labelled reduction system. We say that  $R$  enjoys *confluence* if whenever  $t_1 \xleftarrow{l_1}^* s \xrightarrow{l_2}^* t_2$  there exist  $h_1, h_2 \in L$  and  $u \in \Lambda$  such that  $t_1 \xrightarrow{h_1}^* u \xleftarrow{h_2}^* t_2$  with  $h_1 \cdot f_1 = h_2 \cdot f_2$ . If the hypothesis becomes  $t_1 \xleftarrow{l_1} s \xrightarrow{l_2} t_2$ , we say  $R$  is *locally confluent*.

The well-known Newman Lemma continues to hold even in presence of labels: local confluence together with strong normalization imply confluence, something which can be proved exactly as for abstract reduction systems [38].

## IV. TYPES AND MORPHISMS

Intersection types are often defined *equationally*: the intersection type constructor  $a \sqcap b$  is given as a monoidal product satisfying some equations: associativity, commutativity and possibly idempotency. In this paper, we take a more refined viewpoint: the intersection connective is still a strictly associative tensor product, but we replace equations by *concrete* operations on types. For instance, commutativity is replaced by appropriate *symmetries*  $\sigma_{a,b} : a \sqcap b \rightarrow b \sqcap a$ . Idempotency becomes a *copying operation*  $c_a : a \rightarrow a \sqcap a$ . In order to formalize this idea, we identify the intersection type  $a_1 \sqcap \dots \sqcap a_k$  with the list  $\langle a_1, \dots, a_k \rangle$ . Operations on intersection types become then structural operations on lists, seen as a categorical tensor product (see Section III). Moreover, intersection types will appear on the left side of an application  $\langle a_1, \dots, a_k \rangle \multimap a$  only, following the well-established tradition of *strict intersection types* [4]. This allows us to define a syntax-directed type system.

We define the set of *resource types* ( $\mathcal{RT}$ ) and the set of *intersection types* ( $!(\mathcal{RT})$ ) by the following inductive grammar:

$$\begin{aligned} \mathcal{RT} \ni a, b &::= o \mid \vec{a} \multimap a \\ !(\mathcal{RT}) \ni \vec{a} &::= \langle a_1, \dots, a_k \rangle \quad (k \in \mathbb{N}) \end{aligned}$$

We define a notion of *morphism* of resource types by induction in Figure 1b, parametrically on  $\spadesuit \in \{l, c\}$ . *Morphisms* of resource types can be seen as proof-relevant *subtyping*. Composition and identities of these morphisms are defined by induction in a natural way, exploiting the definition of composition induced by the  $!\spadesuit(-)$  construction. Composition is associative and unitary. We can then define a category  $\mathcal{RT}^\spadesuit$ .  $\mathcal{RT}^l$  (resp.  $\mathcal{RT}^c$ ) is said to be the category of *linear* (resp. *cartesian*) resource types in which objects are resource types and morphisms are *linear* (resp. *cartesian*) morphisms of resource types. We remark that the inclusion of categories  $\mathcal{O}_f^l \hookrightarrow \mathcal{O}_f^c$  induces an inclusion  $\mathcal{RT}^l \hookrightarrow \mathcal{RT}^c$ . We will often denote the identity morphism  $\text{id}_a : a \rightarrow a$  by the type  $a$  itself, e.g.,  $\vec{a} \multimap a := \text{id}_{\vec{a}} \multimap \text{id}_a$ .

**Example 2.** We have a morphism  $(\langle o, o \rangle \multimap o) \multimap o \rightarrow (\langle o, \langle o \rangle \multimap o \rangle \multimap o)$  in the category of cartesian resource

types. The morphism is built out of  $c_o : \langle o \rangle \rightarrow \langle o, o \rangle$ , used in a contravariant way, since the intersection type is in negative position.

## V. RESOURCE TERMS

We now introduce the *structural resource*  $\lambda$ -calculus, which can be thought of as a *non-uniform* counterpart to the standard simply-typed  $\lambda$ -calculus in which the structural operations are kept *explicit*. Non-uniformity stems from the application rule, whereas a function can be applied to bags (*i.e.*, lists) of inputs of non-uniform shape and type: *e.g.*, given a function  $s$ , we apply it to a list of arguments obtaining  $s\langle t_1, \dots, t_k \rangle$ , where  $t_1, \dots, t_k$  are arbitrary terms. In this way, our calculus becomes a refinement of both Boudol's [8] and Ehrhard and Regnier's [14] resource calculi. The main difference is that we replace *multisets* of arguments with *lists*, making the operational semantics completely deterministic. In doing so, we build on several previous proposals [29], [33], [39] where resource calculi based on lists are considered. The novelty of our approach relies on the treatment of *structural rules*. The calculus is built in such a way as to separate the *structural* (and in particular *exponential*, in the sense of linear logic) side of  $\beta$ -reduction from the actual substitution operation, which remains linear.

**Definition 3** (Structural Resource Terms). We fix a countable set of variables  $\mathcal{V}$ . We define the set of *structural resource terms*  $(\Lambda_r^\spadesuit)$  by the following inductive grammar, parametrically over  $\spadesuit \in \{l, c\}$

$$\Lambda_r^\spadesuit \ni s ::= x \in \mathcal{V} \mid \lambda x^f.s \mid s\vec{t} \quad \vec{t} \in !^\spadesuit(\Lambda_r^\spadesuit)$$

where  $f \in \text{hom}(\mathcal{RT}^\spadesuit)$ . Resource terms are considered up to renaming of bound variables. Elements of  $!^\spadesuit(\Lambda_r^\spadesuit)$  are called *bags* of resource terms. Typing contexts, consisting of sequences of variable type declarations, are defined in Figure 1b.

A morphism of typing contexts consists of a sequence of morphisms between intersection types. Typing contexts and their morphisms determine a category that we denote as  $\text{Ctx}$ . Terms are considered up to renaming of bound variables. The tensor product of contexts consists of the pairwise concatenation of intersection types assigned to the same variable. Typing rules are given in Figure 1c. Typing contexts in multiple-hypothesis rules are always assumed to be joinable, *i.e.*, they contain the same free variables. The type system admits two kinds of judgments: namely  $\vdash$  and  $\vdash_b$ . The former infers the typing of resource terms; the latter that of bags of resource terms. We note that bags of resource terms are typed with intersection types and they are not formally a part of the grammar of resource terms.

The main novelty of the structural resource calculus is the typing of abstractions: whenever we bound variables, we are allowed to decorate them with morphisms of intersection types. These morphisms express the (nested) structural rules that we are allowing on (the occurrences of) the considered bound variables.

**Remark 2.** The annotation of bound variables in a term like  $\lambda x^f.s$  intuitively retains the information about the (nested) *structural* operations being performed on the occurrences of the variable  $x$ . For instance, in the term  $s = \lambda x^{c_o}.f\langle x, x \rangle$ , where  $c_o : \langle o \rangle \rightarrow \langle o, o \rangle$ , we are performing a *contraction* on the two occurrences of  $x$ . Thus, if we take a term  $t$  of type  $o$  and apply it to  $s$ , it will be *duplicated* along the computation. This means that our reduction semantics should allow for  $s\langle t \rangle \rightarrow f\langle t, t \rangle$ . In the cartesian world, any structural operation is allowed; for example, we also have weakenings:  $\lambda x^{\top \circ z}$ . In the linear world, however, the only allowed operations are *permutations*.

**Example 3** (Typing Some Resource Terms). Consider the ground morphism  $\alpha = (\langle \langle o \rangle \multimap o \rangle \otimes c_o) : \langle \langle o, o \rangle \multimap o, o \rangle \rightarrow \langle \langle o, o \rangle \multimap o, o, o \rangle$ , which performs a contraction on the type  $o$  while keeping the arrow type fixed. We can type the term  $\lambda x^\alpha.x\langle x, x \rangle$  as  $\vdash \lambda x^\alpha.x\langle x, x \rangle : \langle \langle o, o \rangle \multimap o, o \rangle \multimap o$ .

Our system induces a strong correspondence between typed terms and their type derivations, as witnessed by the following result.

**Proposition 1** (Uniqueness of Derivations). *Let  $\pi$  be a derivation of  $\gamma \vdash s : a$  and  $\pi'$  of  $\gamma \vdash s : a'$ . Then  $a = a'$  and  $\pi = \pi'$ .*

Thanks to this result, we can just work with typed terms  $\gamma \vdash s : a$ , without the need of always explicitly referring to the whole type derivation.

**$\eta$ -long forms:** We now introduce a class of structural resource terms, namely the so called  $\eta$ -long forms. Focusing on  $\eta$ -long forms allows for a smoother definition of the rewriting systems we are going to introduce in the next section. The notion of  $\eta$ -long term we consider relies on full application: an application  $s\vec{t}_1 \dots \vec{t}_n$  should be of atomic type  $o$ , both when  $s$  is a variable and when  $s$  is a function. In doing so, we follow [7]. We first define resource *types* in  $\eta$ -long form:

$$\eta(\mathcal{RT}) \ni a ::= o \mid \bar{a} \multimap o \\ \bar{a} ::= (\bar{a}_1 \dots \bar{a}_k) \quad \bar{a} \in !^\spadesuit(\eta(\mathcal{RT}))$$

A resource type in  $\eta$ -long form can be translated to a standard resource type by setting  $((\bar{a}_1 \dots \bar{a}_n) \multimap o)^* = (\bar{a}_1)^* \multimap (\dots \multimap (\bar{a}_n)^* \multimap o) \dots$ . There is also the evident inverse translation, that exploits the fact that any resource type  $a$  can be written uniquely as a sequence of implications  $a = \bar{a}_1 \multimap (\dots \multimap (\bar{a}_n \multimap o) \dots)$  for some  $\bar{a}_i \in !(\mathcal{RT})$ .

Structural resource *terms* in  $\eta$ -long form are those structural resource terms defined by the following inductive grammar:

$$\eta(\Lambda^\spadesuit) \ni s, t ::= x\bar{q} \mid \lambda(x_1^{f_1} \dots x_n^{f_n}).s \mid (\lambda\bar{x}^{\bar{f}}.s)\bar{q} \\ \bar{q} ::= (\bar{s}_1 \dots \bar{s}_k) \quad \bar{s} \in !^\spadesuit(\eta(\Lambda^\spadesuit))$$

The main difference with the standard grammar is the presence of term sequences  $\bar{q}$ , not to be confused with bags. Such a term sequence is in fact a sequence  $\bar{q}_1, \dots, \bar{q}_n$  of *bags*. In the above,  $\bar{f}$  stands for a sequence of type morphisms having the same length as  $\bar{x}$ . Typing rules are in Figure 2. A term in  $\eta$ -long

<div style="text-align: center;"> <math display="block">\frac{}{id_o : o \rightarrow o} \quad \frac{\langle \alpha, \vec{f} \rangle : \vec{b} \rightarrow \vec{a} \quad f : a \rightarrow b}{(\langle \alpha, \vec{f} \rangle \multimap f) : (\vec{a} \multimap a) \rightarrow (\vec{b} \multimap b)}</math> <math display="block">\frac{\alpha \in \mathbb{O}_f^\bullet([m], [n]) \quad f_1 : a_{\alpha(1)} \rightarrow b_1 \dots f_m : a_{\alpha(m)} \rightarrow b_m}{\langle \alpha; f_1, \dots, f_m \rangle : \langle a_1, \dots, a_n \rangle \rightarrow \langle b_1, \dots, b_m \rangle}</math> <p>(a) Type Morphisms.</p> <hr/> <div style="text-align: center;"> <math display="block">\frac{}{() \text{ context}} \quad \frac{\gamma \text{ context} \quad x \text{ fresh variable, } \vec{a} \in !(\mathcal{RT})}{\gamma, x : \vec{a} \text{ context}}</math> <p>(b) Context Formation.</p> </div> </div>	<div style="text-align: center;"> <math display="block">\frac{}{x_1 : \langle \rangle, \dots, x_i : \langle a \rangle, \dots, x_n : \langle \rangle \vdash x_i : a}</math> <math display="block">\frac{\gamma, x : \vec{b} \vdash s : a \quad f : \vec{a} \rightarrow \vec{b} \quad \gamma_0 \vdash s : \vec{a} \multimap b \quad \gamma_1 \vdash_b \vec{t} : \vec{a}}{\gamma \vdash \lambda x^f.s : \vec{a} \multimap a} \quad \frac{}{\gamma_0 \otimes \gamma_1 \vdash s \vec{t} : b}</math> <math display="block">\frac{\gamma_1 \vdash t_1 : a_1 \quad \dots \quad \gamma_k \vdash t_k : a_k}{\gamma_1 \otimes \dots \otimes \gamma_k \vdash_b \langle t_1, \dots, t_k \rangle : \langle a_1, \dots, a_k \rangle}</math> <p>(c) Typing Assignment.</p> </div>
---	---

Fig. 1: Structural Resource Terms.

$$\begin{array}{c}
\frac{}{x_1 : \langle \rangle, \dots, x_i : \langle \vec{a} \multimap o \rangle, \dots, x_n : \langle \rangle \vdash_v x_i : \vec{a} \multimap o} \quad \frac{\gamma, x_1 : \vec{b}_1, \dots, x_n : \vec{b}_n \vdash s : o \quad f_i : \vec{a}_i \rightarrow \vec{b}_i}{\gamma \vdash \lambda(x_1^{f_1} \dots x_n^{f_n}).s : (\vec{a}_1 \dots \vec{a}_n) \multimap o} \\
\\
\frac{\gamma \vdash_v x : \vec{a} \multimap o \quad \gamma' \vdash_s \vec{q} : \vec{a}}{\gamma \otimes \gamma' \vdash x \vec{q} : o} \quad \frac{\gamma_0 \vdash \lambda \vec{x}^{\vec{f}}.s : \vec{a} \multimap o \quad \gamma_1 \vdash_s \vec{t} : \vec{a}}{\gamma_0 \otimes \gamma_1 \vdash (\lambda \vec{x}^{\vec{f}}.s) \vec{t} : o} \\
\\
\frac{\gamma_1 \vdash_b \vec{q}_1 : \vec{a}_1 \quad \dots \quad \gamma_k \vdash_b \vec{q}_n : \vec{a}_n}{\gamma_1 \otimes \dots \otimes \gamma_n \vdash_s (\vec{q}_1 \dots \vec{q}_n) : (\vec{a}_1 \dots \vec{a}_n)} \quad \frac{\gamma_1 \vdash t_1 : a_1 \quad \dots \quad \gamma_k \vdash t_k : a_k}{\gamma_1 \otimes \dots \otimes \gamma_k \vdash_b \langle t_1, \dots, t_k \rangle : \langle a_1, \dots, a_k \rangle}
\end{array}$$

Fig. 2: Structural Resource Terms in  $\eta$ -long Form.

form induces a resource term in the evident way. Conversely, any resource term  $\gamma \vdash s : \vec{a}_1 \multimap (\dots \multimap (\vec{a}_n \multimap o) \dots)$  can be easily attributed a  $\eta$ -long form.

**Actions of type morphisms:** Given a type morphism  $f : a \rightarrow b$  and a typing derivation  $\pi$  of conclusion  $\gamma \vdash s : a$ , it is natural to expect that we can transform  $\pi$  into another derivation  $[f]\pi$  of conclusion  $\gamma \vdash [f]s : b$  for some term  $[f]s$ . Something analogous should hold for context morphisms: if we take  $\theta : \delta \rightarrow \gamma$  we should get a new derivation  $\pi\{\theta\}$  of conclusion  $\delta \vdash s\{\theta\} : a$ . Intuitively, these transformations consist in applying appropriate structural rules at the level of free and bound variables. Consider, for instance, the following typing

$$x : \langle \langle o \rangle \multimap o \rangle, y : \langle o \rangle \vdash x \langle y \rangle : o$$

and the type morphism  $(c_o \multimap o) : (\langle o, o \rangle \multimap o) \rightarrow (\langle o \rangle \multimap o)$ . This morphism *replaces* a function that demands *one* input with another that demands *two* inputs. In order to accommodate this request, the bag of inputs must be *changed*, i.e., the variable  $y$  has to be duplicated. In doing so, however, we are obliged to change the typing context, too. The new context cannot be just  $x : \langle \langle o, o \rangle \multimap o \rangle, y : \langle o \rangle$  but rather  $\delta = x : \langle \langle o, o \rangle \multimap o \rangle, y : \langle o, o \rangle$ . Hence, the naive intuition about our transformation is not entirely correct: when we apply  $\theta$  to  $\pi$ , the resulting conclusion will be  $\delta^{[\nu]} \vdash s\{\theta\} : a$ , where  $\nu$  is an appropriate ground morphism that depends on  $\pi$  and  $\theta$ . The action of  $\nu$  represents the propagation of the action of

$\theta$  on other variables of the considered term. We shall now give the formal definition of these transformations. Our definition is inspired by the notion of actions of symmetries over linear resource terms, introduced in [39] and on its generalization presented in [31].

Given a type derivation  $\pi$  of conclusion  $\gamma \vdash s : a$  and morphisms  $\theta : \delta \rightarrow \gamma$  and  $f : a \rightarrow b$ , we define the *contravariant* and *covariant* actions on  $\pi$  induced by  $\theta$  and  $f$ , as follows

$$\begin{array}{ccc}
\pi\{\theta\} & & [f]\pi \\
\vdots & & \vdots \\
\delta^{[\nu_s^\theta]} \vdash s\{\theta\} : a & & \gamma^{[\mu_s^f]} \vdash [f]s : b
\end{array}$$

where  $\mu_s^f$  and  $\nu_s^\theta$  are ground morphisms. The actions are defined by induction in Figure 3 and Figure 4, respectively. We often write just  $\nu, \mu$  keeping the parameters  $s, f, \theta$  implicit. In Figure 3, we assume that  $\theta$  is of the shape  $\theta = \langle \text{id}; \vec{f}_1 \rangle, \dots, \langle \text{id}; \vec{f}_n \rangle$  for some  $n \in \mathbb{N}$ . The assumption allows us to decompose the morphism  $\theta$  along the multiplicative merging of contexts in the case of multiple-hypothesis rules. We can extend the definition to arbitrary morphisms, simply by setting  $\pi\{\theta\} = \pi\{\text{nest}(\theta)\}$  (see Remark 1) and  $\nu_s^\theta = \nu_s^{\text{nest}(\theta)} \circ \text{ground}(\theta)$ . We have that  $s\{\text{id}\} = s$  and  $[\text{id}]s = s$ . We also have compositionality:

**Proposition 2** (Compositionality). *Let  $\gamma \vdash s : a$  and  $\theta : \delta \rightarrow \gamma, \eta : \delta' \rightarrow \delta, f : a \rightarrow b, g : b \rightarrow c$ . Then, the following*

statements hold.

$$\begin{aligned} (\gamma[\mu_s^f])[\mu_{[f]s}^g] \vdash [g]([f]s) : c &= \gamma[\mu_s^{gf}] \vdash [gf]s : c. \\ (\delta'[\nu_s^\theta])[\nu_{s\{\theta\}}^{\theta\eta}] \vdash s\{\theta\}\{\nu_s^\theta\eta\} : a &= \delta'[\nu_s^{\theta\eta}] \vdash s\{\theta\eta\} : a. \end{aligned}$$

## VI. THE TWO REWRITE RELATIONS

We now discuss the reduction semantics of our calculus, which we call *structural reduction*. Structural reduction is defined as the union of two disjoint rewrite relations, called *linear* and *exponential* reduction. While the former mimics  $\beta$ -reduction, the second should be understood as a process of structural rule elimination. We will show how the two interact, proving that they commute in a certain weak sense. We also prove the strong normalization and confluence of both exponential and linear reduction, from which we obtain the same result for structural reduction as a whole.

The reduction of structural resource terms relies on *linear substitution*. Given  $\gamma, x : \vec{a} \vdash s : b$  and  $\delta \vdash_b \vec{t} : \vec{a}$ , we define a permutation  $\sigma_{s, \vec{t}}$  and a term  $(\gamma \otimes \delta)^{[\sigma_{s, \vec{t}}]} \vdash s\{\vec{t}/x\} : b$ , called the linear substitution of  $x$  in  $s$  by  $\vec{t}$ , by induction, in Figure 5.

### A. Structural Reduction

We define a relation

$$\rightarrow \subseteq \Lambda^c \times (\text{hom}(\text{Ctx}^c) \times \text{hom}(\mathcal{RT}^c)) \times \Lambda^c,$$

called *structural reduction* by induction, see Figure 6. The relation  $\rightarrow$  naturally induces a labelled reduction system, simply by equipping  $(\text{hom}(\text{Ctx}^c) \times \text{hom}(\mathcal{RT}^c))$  with the partial monoid structure induced by composition. We have that if  $(\gamma \vdash s : a) \rightarrow_{\theta; f} (\gamma' \vdash s' : a')$  then  $\theta : \gamma \rightarrow \gamma'$  and  $f : a' \rightarrow a$ . The *exponential* (resp. *linear*) reduction  $\Rightarrow$  (resp.  $\Rightarrow$ ) is defined as the contextual closure of the left-hand-side (resp. right-hand-side) ground step. We observe that  $\rightarrow = \Rightarrow \cup \Rightarrow$ . In the following we shall often write  $s \rightarrow_{\theta; f} t$ , assuming the reduction well-typed and keeping the typing implicit.

**Remark 3.** Exponential reduction does not strictly speaking enjoy subject reduction, since types can indeed change. However, the typings of the redex and reduct are related through the type morphisms labeling the reduction step. Typing is not preserved under linear substitution, either. This is due to the possible permutations of variable occurrences. For instance, consider  $s = ((\lambda y^{\langle o \rangle} . \lambda x^{\langle \langle o \rangle \multimap o \rangle} . xy)\langle z \rangle)\langle z \rangle$  with the following typing:

$$z : \langle o, \langle o \rangle \multimap o \rangle \vdash ((\lambda y^{\langle o \rangle} . \lambda x^{\langle \langle o \rangle \multimap o \rangle} . xy)\langle z \rangle)\langle z \rangle : o$$

Now,  $s \rightarrow z\langle z \rangle$ . However, the typing context must be permuted, becoming  $z : \langle \langle o \rangle \multimap o, o \rangle \vdash z\langle z \rangle : o$ . As for exponential reduction, the evident permutation  $\langle o, \langle o \rangle \multimap o \rangle \rightarrow \langle \langle o \rangle \multimap o, o \rangle$  is recorded as part of the underlying label.

**Example 4.** We give an example of structural reduction. Consider the terms from Section II, namely

$$\begin{aligned} u &= \lambda x^{\langle \langle o \rangle \multimap o \rangle^3} . w\langle x\langle xy \rangle \rangle\langle x\langle y \rangle \rangle; \\ v &= \lambda z^{c_o} . q\langle z \rangle\langle z \rangle; \\ s &= \lambda x^{\langle \langle o \rangle \multimap o \rangle^4} . w\langle x\langle x\langle y, y \rangle, x\langle y, y \rangle \rangle\langle x\langle y, y \rangle \rangle; \\ t &= \lambda z^{\langle o, o \rangle} . q\langle z, z \rangle. \end{aligned}$$

We can normalize  $u\langle v \rangle^3$  into  $s\langle t \rangle^4$ . Since  $q : \langle \langle \langle o \rangle \cdot \langle o \rangle \rangle \multimap o \rangle \vdash \lambda z^{c_o} . q\langle z \rangle\langle z \rangle : \langle o \rangle \multimap o \rightarrow_{\text{id}; c_o \multimap o} (q : \langle \langle \langle o \rangle \cdot \langle o \rangle \rangle \multimap o \rangle \vdash \lambda z^{\langle o \rangle^2} . q\langle z \rangle\langle z \rangle : \langle o, o \rangle \multimap o)$ . Then we have

$$\begin{aligned} [c_o \multimap o](w : \langle \langle \langle o \rangle \cdot \langle o \rangle \rangle \multimap o \rangle \vdash s : \langle o \rangle^3 \multimap o) &\rightarrow_{\text{id}; (c_o \oplus \text{id}) \multimap o} \\ w : \langle \langle \langle o \rangle \cdot \langle o \rangle \rangle \multimap o \rangle \vdash v : \langle o \rangle^4 \multimap o \end{aligned}$$

Hence, we can conclude that  $(w : \langle \langle \langle o \rangle \cdot \langle o \rangle \rangle \multimap o \rangle, q : \langle \langle \langle o \rangle \cdot \langle o \rangle \rangle \multimap o \rangle \vdash u\langle v \rangle^3 : o) \rightarrow_{\text{id}; \text{id}} (w : \langle \langle \langle o \rangle \cdot \langle o \rangle \rangle \multimap o \rangle, q : \langle \langle \langle o \rangle \cdot \langle o \rangle \rangle \multimap o \rangle \vdash s\langle t \rangle^4 : o)$ .

### B. Termination and Confluence

We are interested in proving strong normalization and confluence of the reduction relations introduced in the last section.

We first consider strong normalization of exponential reduction. This will be proved by means of a reducibility argument [37]. Given that structural reduction is a labelled reduction system, we have to somehow take type morphisms into account. The crucial lemma will then be to prove that if  $\gamma \vdash s : a$  and  $\theta : \delta \rightarrow \gamma$ , then  $s\{\theta\}$  is reducible for the type  $a$ . In order to do so, we need to adjust the classic notion of saturated set of terms. Normally, we would ask for a set to be closed by antireduction: if  $s\{\vec{t}/x\} \in X$ , then  $(\lambda x.s)\vec{t} \in X$ . We will do the same in our case, taking into account the action of morphisms instead of substitution.

We set  $\text{SN} = \{s \mid s \text{ is strongly normalizable for } \Rightarrow\}$ . A set  $X$  of structural resource terms is said to be *saturated* when, given terms  $\gamma, x : \vec{a} \vdash s : b$  and  $\delta \vdash \vec{q} : \vec{b}$  with morphism  $\vec{g} : \vec{b} \rightarrow \vec{a}$  the following condition holds: if  $\vec{q} \in \text{SN}$  and for all  $\vec{f}$  s.t.  $\vec{q} \rightarrow_{\theta; \vec{f}} \vec{q}'$  we have that  $s\{\text{id}; \vec{g}\vec{f}\} \in X$ , then  $(\lambda x^{\vec{g}}.s)\vec{q} \in X$ .

**Proposition 3.** *SN is saturated.*

Given a resource type  $a$ , we define the *reducibility candidates* for  $a$ ,  $\text{Red}(a) \subseteq \Lambda_r$ , by induction on the structure of  $a$ , as follows:

$$\begin{aligned} \text{Red}(o) &= \text{SN}; \\ \text{Red}(\langle a_1, \dots, a_k \rangle) &= \{\langle t_1, \dots, t_k \rangle \mid t_i \in \text{Red}(a_i)\}; \\ \text{Red}(\vec{a} \multimap o) &= \{s \in \Lambda_r \mid \text{for all } \vec{q} \in \text{Red}(\vec{a}), s\vec{q} \in \text{Red}(o)\}; \\ \text{Red}(\langle \vec{a}_1 \cdots \vec{a}_n \rangle) &= \{\langle \vec{t}_1 \cdots \vec{t}_n \rangle \mid \vec{t}_i \in \text{Red}(\vec{a}_i)\} \end{aligned}$$

**Lemma 1.** *We have that  $\text{Red}(a) \subseteq \text{SN}$ .*

**Lemma 2.** *Let  $\gamma \vdash s : a$  and  $\theta : \delta \rightarrow \gamma, f : a \rightarrow b$ . Then  $s\{\theta\} \in \text{Red}(a)$  and  $[f]s \in \text{Red}(b)$ .*

$$\begin{aligned}
& \left( \overline{x_1 : \langle \rangle, \dots, x_i : \langle \bar{a} \multimap o \rangle, \dots, x_n : \langle \rangle \vdash_v x_i : \bar{a} \multimap o} \right) \{ \text{id}_{\langle \rangle}, \dots, \langle \bar{f} \multimap o \rangle, \dots, \text{id}_{\langle \rangle} \} = \overline{x_1 : \langle \rangle, \dots, x_i : \langle \bar{b} \multimap o \rangle, \dots, x_n : \langle \rangle \vdash_v x_i : \bar{b} \multimap o} \\
& \left( \frac{\gamma_1 \vdash_v x : \bar{a} \multimap o \quad \gamma_2 \vdash_s \bar{q} : \bar{a}}{\gamma_1 \otimes \gamma_2 \vdash x \bar{q} : o} \right) \{ \eta \otimes \theta \} = \frac{\delta_1 \vdash_v x : \bar{b} \multimap o \quad (\delta_2^{[\nu_2]})^{[\mu]} \vdash_s [\bar{f}] (\bar{q} \{ \theta \}) : \bar{b}}{\delta_1 \otimes (\delta_2^{[\nu_2 \mu]}) \vdash x [\bar{f}] (\bar{q} \{ \theta \}) : o} \\
& \left( \frac{\gamma, \bar{y} : \bar{b} \vdash s : o \quad \bar{f} : \bar{a} \rightarrow \bar{b}}{\gamma \vdash \lambda \bar{y} \bar{f}.s : \bar{a} \multimap o} \right) \{ \theta \} = \frac{\delta^{[\nu]}, \bar{y} : \bar{b}^{[\nu']} \vdash s \{ \theta, \bar{b} \} : o \quad \bar{f} \nu' : \bar{a} \rightarrow \bar{b}^{[\nu']}}{\delta^{[\nu]} \vdash \lambda \bar{y} \bar{f} \nu'.s \{ \theta, \bar{b} \} : \bar{a} \multimap o} \\
& \left( \frac{\gamma_0 \vdash \lambda \bar{x} \bar{f}.s : \bar{a} \multimap o \quad \gamma_1 \vdash \bar{t} : \bar{a}}{\gamma_0 \otimes \gamma_1 \vdash (\lambda \bar{x} \bar{f}.s) \bar{t} : o} \right) \{ \theta_0 \otimes \theta_1 \} = \frac{\delta_0^{[\nu_0]} \vdash (\lambda \bar{x} \bar{f}.s) \{ \theta_0 \} : \bar{a} \multimap o \quad \delta_1^{[\nu_1]} \vdash_s \bar{t} \{ \theta_1 \} : \bar{a}}{(\delta_0 \otimes \delta_1)^{[\nu_0 \otimes \nu_1]} \vdash (\lambda \bar{x} \bar{f}.s) \{ \theta_0 \} \bar{t} \{ \theta_1 \} : o} \\
& \left( \frac{(\gamma_i \vdash \bar{q}_i : \bar{a}_i)_{i=1}^n}{\otimes \gamma_i \vdash_s (\bar{q}_1 \dots \bar{q}_n) : (\bar{a}_1 \dots \bar{a}_n)} \right) \{ \otimes \theta_i \} = \frac{(\delta_i^{[\nu_i]} \vdash \bar{q}_i \{ \theta_i \} : \bar{a}_i)_{i=1}^n}{(\otimes \delta_i)^{[\sum \nu_i]} \vdash_s (\bar{q}_1 \{ \theta_1 \} \dots \bar{q}_n \{ \theta_n \}) : (\bar{a}_1 \dots \bar{a}_n)} \\
& \left( \frac{(\gamma_i \vdash t_i : a_i)_{i=1}^k}{\otimes \gamma_i \vdash_b \langle t_1, \dots, t_k \rangle : \langle a_1, \dots, a_k \rangle} \right) \{ \otimes \theta_i \} = \frac{(\delta_i^{[\nu_i]} \vdash t_i \{ \theta_i \} : a_i)_{i=1}^k}{(\otimes \delta_i)^{[\sum \nu_i]} \vdash_b \langle t_1 \{ \theta_1 \}, \dots, t_k \{ \theta_k \} \rangle : \langle a_i \rangle}
\end{aligned}$$

Fig. 3: Contravariant action on resource terms. We omit the explicit typing of morphisms to improve readability. In the variable rule we assume that  $\bar{f} : \bar{a} \rightarrow \bar{b}$ . In the application rule for the variable, we assume that the value of  $\eta$  on  $x$  is  $\bar{f} \multimap o$ .

$$\begin{aligned}
& [\bar{f} \multimap o] \left( \overline{x_1 : \langle \rangle, \dots, x_i : \langle \bar{a} \multimap o \rangle, \dots, x_n : \langle \rangle \vdash_v x_i : \bar{a} \multimap o} \right) = \overline{x_1 : \langle \rangle, \dots, x_i : \langle \bar{b} \multimap o \rangle, \dots, x_n : \langle \rangle \vdash_v x_i : \bar{b} \multimap o} \\
& [id_o] \left( \frac{\gamma_0 \vdash s : \bar{a} \multimap o \quad \gamma_1 \vdash \bar{t} : \bar{a}}{\gamma_0 \otimes \gamma_1 \vdash s \bar{t} : o} \right) = \frac{\gamma_0 \vdash s : \bar{a} \multimap o \quad \gamma_1 \vdash \bar{t} : \bar{a}}{\gamma_0 \otimes \gamma_1 \vdash s \bar{t} : o} \\
& [\bar{f} \multimap o] \left( \frac{\gamma, \bar{x} : \bar{c} \vdash s : o \quad \bar{g} : \bar{a} \rightarrow \bar{c}}{\gamma \vdash \lambda \bar{x} \bar{g}.s : \bar{a} \multimap o} \right) = \frac{\gamma, \bar{x} : \bar{b} \vdash s : o \quad \bar{g} \bar{f} : \bar{b} \rightarrow \bar{c}}{\gamma \vdash \lambda \bar{x} \bar{g} \bar{f}.s : \bar{b} \multimap o} \\
& [(\langle \alpha_i; \bar{f}_i \rangle)_{i=1}^n] \left( \frac{(\gamma_i \vdash \bar{q}_i : \bar{a}_i)_{i=1}^n}{\otimes \gamma_i \vdash_s (\bar{q}_1 \dots \bar{q}_n) : (\bar{a}_1 \dots \bar{a}_n)} \right) = \frac{(\gamma_i^{[\mu_i]} \vdash [(\langle \alpha_i; \bar{f}_i \rangle) \bar{q}_i : \bar{b}_i]_{i=1}^n)}{\otimes \gamma_i^{[\mu_i]} \vdash_s ([(\langle \alpha_1; \bar{f}_1 \rangle) \bar{q}_1 \dots (\langle \alpha_n; \bar{f}_n \rangle) \bar{q}_n] : (\bar{b}_1 \dots \bar{b}_n)} \\
& [(\langle \alpha, f_i \rangle)_{i=1}^l] \left( \frac{(\gamma_i \vdash t_i : a_i)_{i=1}^k}{\otimes \gamma_i \vdash_b \langle t_1, \dots, t_k \rangle : \langle a_1, \dots, a_k \rangle} \right) = \frac{(\gamma_{\alpha(j)}^{[\mu_j]} \vdash [f_j] t_{\alpha(j)} : b_j)_{j=1}^l}{(\otimes \gamma_j)^{[(\sum \mu_j) \alpha^*]} \vdash_b \langle [f_1] t_{\alpha(1)}, \dots, [f_l] t_{\alpha(l)} \rangle : \langle b_1, \dots, b_l \rangle}
\end{aligned}$$

Fig. 4: Covariant action on resource terms. We omit the explicit typing of morphisms to improve readability. We assume that  $\bar{f} : \bar{b} \rightarrow \bar{a}$ . The ground morphism  $\alpha^*$  is the canonical one of type  $\alpha^* : \otimes \gamma_i \rightarrow \otimes \gamma_{\alpha(j)}$ .

*Proof.* By mutual induction on the structure of  $s$ , exploiting the saturation property in the abstraction cases.  $\square$

**Theorem 1** (Adequacy). *If  $\gamma \vdash s : a$  then  $s \in \text{Red}(a)$ .*

*Proof.* Corollary of the former lemma, by taking  $\theta = \text{id}_\gamma$ .  $\square$

**Theorem 2.** *Exponential reduction is strongly normalizing.*

Strong normalization of linear reduction can be proved in a combinatorial way, since the size of derivation strictly *decreases* under linear reduction.

**Proposition 4.** *If  $s \Rightarrow^* s'$  then  $\text{size}(s') < \text{size}(s)$ . As a consequence, linear reduction is strongly normalizing.*

**Remark 4.** Normal forms for the exponential reduction are a special kind of linear resource terms, the *planar* ones. Normal forms for linear reduction, instead, are not linear in general, since non-linear redexes are never fired. Normal forms for general structural reduction are exactly the planar terms in  $\beta$ -normal form.

**Remark 5.** Exponential steps can generate linear redexes. For instance,  $(\lambda x^{c_o}.w\langle x \rangle \langle x \rangle) \langle y \rangle$  is a linear normal form;



$$\begin{aligned}
& \frac{}{(x_1 : \langle \rangle, \dots, x_i : \langle a \rangle, \dots, x_n : \langle \rangle \vdash x : a)} \{ \langle t \rangle / x_i \} = \gamma \vdash t : a \\
& \frac{\left( \frac{\gamma, x : \vec{a}, y : \vec{b} \vdash s : b \quad f : \vec{a} \rightarrow \vec{b}}{\gamma, x : \vec{a} \vdash \lambda y^f . s : \vec{a} \multimap b} \right) \{ \vec{t} / x \}}{(\gamma \otimes \delta)^{[\sigma]}, y : (\vec{b})^{[\sigma']} \vdash s \{ \vec{t} / x \} : b \quad \sigma' f} = \frac{(\gamma \otimes \delta)^{[\sigma]}, y : (\vec{b})^{[\sigma']} \vdash s \{ \vec{t} / x \} : b \quad \sigma' f}{(\gamma \otimes \delta)^{[\sigma]} \vdash \lambda y^{\sigma' f} . (s \{ \vec{t} / x \}) : \vec{a} \multimap b} \\
& \frac{\left( \frac{\gamma_0, x : \vec{a}_0 \vdash s : \vec{a} \multimap b \quad \gamma_1, x : \vec{a}_1 \vdash \vec{q} : \vec{a}}{\gamma_0 \otimes \gamma_1, x : \vec{a}_0 \oplus \vec{a}_1 \vdash s \vec{q} : b} \right) \{ \vec{t}_0 \oplus \vec{t}_1 \}}{(\gamma_0 \otimes \delta_0)^{[\sigma_0]} \vdash s \{ \vec{t}_0 / x \} : \vec{a} \multimap b \quad (\gamma_1 \otimes \delta_1)^{[\sigma_1]} \vdash \vec{q} \{ \vec{t}_1 / x \} : \vec{a}} = \frac{(\gamma_0 \otimes \delta_0)^{[\sigma_0]} \vdash s \{ \vec{t}_0 / x \} : \vec{a} \multimap b \quad (\gamma_1 \otimes \delta_1)^{[\sigma_1]} \vdash \vec{q} \{ \vec{t}_1 / x \} : \vec{a}}{((\gamma_0 \otimes \gamma_1) \otimes (\delta_0 \otimes \delta_1))^{[(\sigma_0 \oplus \sigma_2) \tau]} \vdash s \{ \vec{t}_0 / x \} \vec{q} \{ \vec{t}_1 / x \} : b} \\
& \frac{\left( \frac{(\gamma_i, x : \vec{a}_i \vdash s_i : b_i)_{i=1}^k}{\bigotimes \gamma_i, x : \bigoplus \vec{a}_i \vdash_b \langle s_1, \dots, s_k \rangle : \vec{b}} \right) \{ \bigoplus \vec{t}_i / x \}}{((\gamma_i \otimes \delta_i)^{[\sigma_i]} \vdash s_i \{ \vec{t}_i / x \} : b_i)_{i=1}^k} = \frac{((\gamma_i \otimes \delta_i)^{[\sigma_i]} \vdash s_i \{ \vec{t}_i / x \} : b_i)_{i=1}^k}{(\bigotimes \gamma_i \otimes \bigotimes \delta_i)^{[(\bigoplus \sigma_i) \tau]} \vdash_b \langle s_1 \{ \vec{t}_1 / x \}, \dots, s_k \{ \vec{t}_k / x \} \rangle : \vec{b}}
\end{aligned}$$

Fig. 5: Substitution operation on typed resource terms. In the application and list rules, the list terms  $\vec{t}_i$  are of type  $\vec{a}_i$ . They correspond to the decomposition of the list  $\vec{t}$ , following the multiplicative decomposition of the context. The permutation  $\tau$  is the evident one of type  $\tau : (\bigotimes \gamma_i) \otimes (\bigotimes \delta_i) \rightarrow \bigotimes (\gamma_i \otimes \delta_i)$ .

Ground Steps:

$$\frac{\gamma, \vec{x} : \vec{a} \vdash s : o \quad \vec{f} : \vec{b} \rightarrow \vec{a} \quad \vec{f} \text{ not identity}}{(\gamma \vdash \lambda \vec{x}^{\vec{f}} . s : \vec{b} \multimap o) \rightarrow_{\nu, \nu' \multimap o} (\gamma^{[\nu]} \vdash \lambda \vec{x}^{\vec{b} \nu'} . s \{ \gamma, \vec{f} \} : \vec{b}^{[\nu']} \multimap o)} \quad \frac{\gamma, \vec{x} : \vec{a} \vdash s : o \quad \delta \vdash \vec{t} : \vec{b} \quad \tau \text{ a permutation}}{(\gamma \otimes \delta \vdash (\lambda \vec{x}^{\tau} . s) \vec{t} : o) \rightarrow_{\sigma, [\tau] \vec{t} : o} ((\gamma \otimes \delta)^{[\sigma_{s, [\tau] \vec{t}}]} \vdash s \{ [\tau] \vec{t} : o / \vec{x} \})}$$

Contextual Closure:

$$\begin{aligned}
& \frac{(\gamma, x : \vec{a} \vdash s : o) \rightarrow_{\theta, \vec{f} : o} (\gamma', x : \vec{a}' \vdash s' : o) \quad g : \vec{b} \rightarrow \vec{a}}{(\gamma \vdash \lambda x^g . s : \vec{b} \multimap o) \rightarrow_{\theta, \vec{b} \multimap o} (\gamma' \vdash \lambda x^{\vec{f} g} . s' : \vec{b} \multimap o)} \quad \frac{(\gamma \vdash s : \vec{a} \multimap o) \rightarrow_{\theta, \vec{f} \multimap o} (\gamma' \vdash s' : \vec{a}' \multimap o) \quad \delta \vdash_s \vec{t} : \vec{a}}{(\gamma \otimes \delta \vdash s \vec{t} : o) \rightarrow_{\theta \otimes \mu_{\vec{t}}^{\vec{f}} : o} (\gamma' \otimes \delta^{[\mu_{\vec{t}}^{\vec{f}}]} \vdash s' [\vec{f}] \vec{t} : o)} \\
& \frac{(\delta \vdash \vec{t} : \vec{a}) \rightarrow_{\theta, \vec{f}} (\delta' \vdash \vec{t}' : \vec{a}') \quad \gamma \vdash s : \vec{a} \multimap o}{(\gamma \otimes \delta \vdash s \vec{t} : o) \rightarrow_{\mu_{\vec{t}}^{\vec{f} \multimap o} \otimes \theta : o} (\gamma^{\mu_s^{[\vec{f} \multimap o]}} \otimes \delta' \vdash ([\vec{f} \multimap o] s) \vec{t}' : o)} \\
& \frac{(\gamma_1 \vdash s_1 : a_1) \rightarrow_{\theta_1, f_1} (\gamma'_1 \vdash s'_1 : a'_1) \quad \dots \quad (\gamma_k \vdash s_k : a_k) \rightarrow_{\theta_k, f_k} (\gamma'_k \vdash s'_k : a'_k)}{(\bigotimes \gamma_i \vdash_b \langle s_1, \dots, s_k \rangle) \rightarrow_{\bigotimes \theta_i ; \langle id; f_1, \dots, f_k \rangle} (\bigotimes \gamma'_i \vdash_b \langle s'_1, \dots, s'_k \rangle : \langle a'_1, \dots, a'_k \rangle)} \\
& \frac{(\gamma \vdash_b \vec{q} : \vec{a}) \rightarrow_{\theta, \langle \alpha, \vec{f} \rangle} (\gamma' \vdash \vec{q}' : \vec{a}') \quad \delta_1 \vdash_s \vec{q}_1 : \vec{a}_1 \quad \delta_2 \vdash_s \vec{q}_2 : \vec{a}_2}{(\delta_1 \otimes \gamma \otimes \delta_2 \vdash_s (\vec{q}_1 \cdot \vec{q} \cdot \vec{q}_2) : (\vec{a}_1 \cdot \vec{a} \cdot \vec{a}_2)) \rightarrow_{\delta_1 \otimes \theta \otimes \delta_2 ; \vec{a}_1 \cdot \langle \alpha, \vec{f} \rangle \cdot \delta_2} (\delta_1 \otimes \gamma' \otimes \delta_2 \vdash_s (\vec{q}_1 \cdot \vec{q}' \cdot \vec{q}_2) : (\vec{a}_1 \cdot \vec{a}' \cdot \vec{a}_2))}
\end{aligned}$$

Fig. 6: Structural reduction.

however, if we perform a step of exponential reduction we get the term  $(\lambda x^{(o, o)}. w \langle x \rangle \langle x \rangle) \langle y, y \rangle$  that is now a linear redex. Linear steps too can generate exponential redexes. This is due to the permutations generated by substitution. If you take  $s = \lambda z^{(o, (o) \multimap o)}. (\lambda x. \lambda y. y \langle x \rangle) \langle z, z \rangle$ , the term  $s$  is an exponential normal form. However, by performing the leftmost linear reduction to its normal form, we obtain the term  $t = \lambda z^\sigma. z \langle z \rangle$ , since the substitution forces the permutation of the two occurrences of  $z$ . Now  $t$  is not an exponential normal form anymore. Hence, the two reductions do not strictly commute with each other.

We are now going to prove a form of weak commutation, mediated by an isomorphism. We define  $\cong \subseteq (\Lambda_r \times \text{hom}(\text{Ctx}) \times \Lambda_r)$  called the *isomorphism relation* be-

tween resource terms, by induction in Figure 7. We note that if  $s \cong_\theta s'$  then  $\theta$  is a permutation. We write  $s \cong s'$  iff  $s \cong_\theta s'$  for some  $\theta$ .

**Lemma 3.** *Let  $s \cong s'$  and  $s \Rightarrow p$  (resp.  $s \Rightarrow p$ ). Then there exists a term  $p'$  s.t.  $s' \Rightarrow p'$  (resp.  $s' \Rightarrow p'$ ) and  $p \cong p'$ .*

**Theorem 3 (Commutation).** *Let  $s \Rightarrow^* t \Rightarrow^* t'$ . Then there exist terms  $u, u'$  s.t.  $s \Rightarrow^* u \Rightarrow^* u'$  with  $t' \cong u'$ .*

From these commutation results, we can lift strong normalization of  $\Rightarrow$  and  $\Rightarrow$  to the whole structural reduction:

**Theorem 4.** *Structural reduction is strongly normalizing.*

*Proof.* By Theorem 8, given an infinite reduction sequence for structural reduction, we could build an infinite one for either

$$\begin{array}{c}
\frac{}{(x : \langle a \rangle \vdash x : a) \cong_{\langle a \rangle} (x : \langle a \rangle \vdash x : a)} \quad \frac{\sigma : \gamma \rightarrow \gamma^{[\sigma]} \text{ is a permutation}}{(\gamma \vdash s : a) \cong_{\sigma} (\gamma^{[\sigma]} \vdash s : a)} \quad \frac{(\gamma, x : \vec{a} \vdash s : a) \cong_{\theta, \sigma} (\gamma', x : \vec{a}' \vdash s' : a') \quad f : \vec{b} \rightarrow \vec{a}}{(\gamma \vdash \lambda x^f. s : \vec{b} \multimap a) \cong_{\theta} (\gamma' \vdash \lambda x^{\sigma f}. s' : \vec{b} \multimap a)} \\
\\
\frac{(\gamma_0 \vdash s : \vec{a} \multimap a) \cong_{\theta_0} (\gamma'_0 \vdash s' : \vec{a} \multimap a) \quad (\gamma_1 \vdash \vec{t} : \vec{a}) \cong_{\theta_1} (\gamma'_1 \vdash \vec{t}' : \vec{a})}{(\gamma_0 \otimes \gamma_1 \vdash s \vec{t} : a) \cong_{\theta_0 \otimes \theta_1} (\gamma'_0 \otimes \gamma'_1 \vdash s' \vec{t}' : a)} \\
\\
\frac{((\gamma_i \vdash t_i : a_i) \cong_{\theta_i} (\gamma'_i \vdash t'_i : a_i))_{i=1}^k}{(\bigotimes \gamma_i \vdash \langle t_1, \dots, t_k \rangle : \langle a_1, \dots, a_k \rangle) \cong_{\bigotimes \theta_i} (\bigotimes \gamma'_i \vdash \langle t'_1, \dots, t'_k \rangle : \langle a_1, \dots, a_k \rangle)}
\end{array}$$

Fig. 7: Isomorphism of Resource Terms.

exponential or linear reductions, which are instead known to be strongly normalizing.  $\square$

Let us now consider confluence. To prove that, we employ a standard method: first we prove local confluence, from which we infer general confluence as a corollary of Newman's Lemma, exploiting the fact that structural reduction terminates.

**Theorem 5.** *Structural reduction is confluent.*

## VII. ON APPROXIMATIONS, COLLAPSE, AND ALL THAT

### A. Linearizing Type Systems

The structural resource  $\lambda$ -calculus can be seen as a calculus of approximations of ordinary  $\lambda$ -terms, generalizing what happens in the Taylor expansion [5], [14]. We define an *approximation relation*  $\triangleleft \subseteq \Lambda_r^c \times \Lambda$  by induction in Figure 8a. In general, approximations are *not* linear. However, we can *compute* a linear approximation starting from a non-linear one, just by normalizing through exponential reduction. We shall use this fact to get linearizations of every strongly normalizing  $\lambda$ -terms.

In this section we will assume that a typed term  $\Gamma \vdash M : A$  is in  $\eta$ -long form. We remark that any term, typed with either simple or intersection types, has a  $\eta$ -long form.

**Qualitative Fragment and Uniformity:** We can define a fragment of the structural resource calculus into which we shall *embed* every strongly normalizing  $\lambda$ -term. First, we formally define a *coherence relation* on resource terms  $s \subset s'$ , that is a direct generalization of coherence as defined in [14]:

$$\begin{array}{c}
\frac{}{x \subset x} \quad \frac{s \subset s'}{\lambda x^f. s \subset \lambda x^g. s'} \quad \frac{s \subset s' \quad \vec{t} \subset \vec{t}'}{s \vec{t} \subset s' \vec{t}'} \\
\\
\frac{t_i \subset t'_j \quad \text{for all } i \in [n], j \in [m]}{\langle t_1, \dots, t_n \rangle \subset \langle t'_1, \dots, t'_m \rangle}
\end{array}$$

Given an ordinary  $\lambda$ -term  $M$ , we have that  $s, s' \triangleleft M$  iff  $s \subset s'$ . If  $s \subset s$ , we say that  $s$  is *uniform*. We remark that if we have an exponential step  $s \Rightarrow s'$ , then  $s \subset s'$ , while substitution notoriously breaks coherence. We extend the notion of coherence and uniformity to resource type, in

the natural way. We define the *qualitative fragment* of the cartesian resource calculus by induction as follows:

$$\Lambda_{\text{qual}}^c \ni s ::= x \mid \lambda x^\alpha. s \mid s \overbrace{\langle t, \dots, t \rangle}^{k \text{ times}}$$

Where  $\alpha$  is a ground morphism. If  $s$  is qualitative, then  $s$  is uniform. Given a qualitative term  $\gamma \vdash s : a$  we call it *strongly uniform* if  $\gamma \subset \gamma$ ,  $a \subset a$ , and the bags occurring in  $s$  are singletons. In what follows, we give embeddings of strongly normalizing  $\lambda$ -terms into cartesian resource  $\lambda$ -terms which targets the qualitative fragment.

**Simple types:** We can easily embed the simply-typed  $\lambda$ -calculus into the structural resource calculus. We set  $\mathbf{l}(\ast) = o$  and  $\mathbf{l}(A \Rightarrow B) = \langle \mathbf{l}(A) \rangle \multimap \mathbf{l}(B)$ . Given  $n \in \mathbb{N}$  and  $A$  a simple type, we define  $\text{cart}_A^n : \langle \mathbf{l}(A) \rangle \rightarrow \langle \mathbf{l}(A) \rangle^n$  as follows:

$$\text{cart}_A^0 = \mathbf{T}_{\mathbf{l}(A)} \quad \text{cart}_A^{n+1} = c_{\mathbf{l}(A)}^n$$

Given a simply-typed term  $x_1 : A_1, \dots, x_n : A_n \vdash M : A$ , we associate to it a resource term  $x_1 : \langle \mathbf{l}(A_1) \rangle^{n_1^M}, \dots, x_n : \langle \mathbf{l}(A_n) \rangle^{n_n^M} \vdash \text{qt}(M) : \mathbf{l}(A)$  with  $n_i^M \in \mathbb{N}$ , called its *coarse approximation*, by induction, as shown in Figure 9.

We prove that the embedding preserves  $\beta$ -reduction, *factorizing it* into exponential and linear steps:

**Theorem 6.** *Let  $M \rightarrow_\beta N$  then there exists a cartesian resource term  $t$  s.t.  $\text{qt}(M) \Rightarrow t \Rightarrow \text{qt}(N)$ .*

From the former theorem and strong normalization of structural reduction (Theorem 4), we obtain strong normalization of simply-typed terms as a corollary. We observe that if  $M$  is simply-typed, then  $\text{qt}(M)$  is a strongly uniform resource term. From this, we obtain the following characterization:

**Proposition 5.**  *$\gamma \vdash s : a$  is strongly uniform iff there exists  $\Gamma \vdash M : A$  s.t.  $\text{qt}(M) = s$ .*

**Idempotent Intersection Types:** We shall now show that every strongly normalizing  $\lambda$ -term can be represented as a cartesian resource term, by exploiting a classic result about idempotent intersection types. We consider the system of idempotent intersection types in Figure 8b, which corresponds to the one from [4]. Intersection types can be seen as a finite set of types. The system enjoys both subject reduction and expansion, from which one can prove that any strongly

$\frac{}{x \triangleleft x} \quad \frac{s \triangleleft M}{\lambda x^f. s \triangleleft \lambda x. M} \quad \frac{s \triangleleft M \quad \vec{t} \triangleleft N}{s \vec{t} \triangleleft MN}$ $\frac{t_1 \triangleleft M \cdots t_k \triangleleft M}{\langle t_1, \dots, t_k \rangle \triangleleft M}$	$\text{Idem} \ni A ::= o \mid \tilde{A} \Rightarrow B \quad \tilde{A} := A_1 \cap \cdots \cap A_k \ (k \neq 0)$ $\frac{A \in \tilde{A}_i}{x_1 : \tilde{A}_1, \dots, x_i : \tilde{A}_i, \dots, x_n : \tilde{A}_n \vdash x_i : A} \quad \frac{\Gamma, x : \tilde{A} \vdash M : B}{\Gamma \vdash \lambda x. M : \tilde{A} \Rightarrow B}$ $\frac{\Gamma \vdash M : \tilde{A} \Rightarrow B \quad \Gamma \vdash_{\cap} N : \tilde{A}}{\Gamma \vdash MN : B} \quad \frac{(\Gamma \vdash M : A_i)_{i=1}^k}{\Gamma \vdash_{\cap} M : A_1 \cap \cdots \cap A_k}$
(a) Approximation Relation for Ordinary $\lambda$ -terms.	(b) Idempotent Intersection Type Assignment.

Fig. 8: Approximation of Terms and Idempotent Intersection Types.

$$\begin{aligned} \text{qt} \left( \frac{}{x_1 : A_1, \dots, x_i : A_i, \dots, x_n : A_n \vdash x_i : A_i} \right) &= \frac{}{x_1 : \langle \rangle, \dots, x_i : \langle I(A_i) \rangle, \dots, x_n : \langle \rangle \vdash x_i : I(A_i)} \\ \text{qt} \left( \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x^A. M : A \Rightarrow B} \right) &= \frac{I(\Gamma)^{\overline{\pi}}, x : \langle I(A) \rangle^m \vdash \text{qt}(M) : I(B) \quad \text{cart}_A^m : \langle I(A) \rangle \rightarrow \langle I(A) \rangle^m}{I(\Gamma)^{\overline{\pi}} \vdash \lambda x^{\text{cart}_A^m}. \text{qt}(M) : \langle I(A) \rangle \multimap I(B)} \\ \text{qt} \left( \frac{\Gamma \vdash M : A \Rightarrow B \quad \Gamma \vdash N : A}{\Gamma \vdash MN : B} \right) &= \frac{I(\Gamma)^{\overline{\pi}} \vdash \text{qt}(M) : I(A) \multimap I(B) \quad I(\Gamma)^{\overline{\pi}} \vdash \text{qt}(N) : I(A)}{I(\Gamma)^{\overline{\pi}} \otimes I(\Gamma)^{\overline{\pi}} \vdash \text{qt}(M) \langle \text{qt}(N) \rangle : I(B)} \end{aligned}$$

Fig. 9: Translation of simply typed terms into cartesian resource terms.

normalizing  $\lambda$ -term can be typed in it (see, e.g., [26]). We fix an enumeration of types as the bijection  $\text{enum} : \text{idem} \rightarrow \mathbb{N}$ . Then we set

$$I(*) = o \quad I(\tilde{A} \Rightarrow B) = I(\tilde{A}) \multimap I(B)$$

$$I(A_1 \cap \cdots \cap A_k) = \langle I(A_{\sigma(1)}), \dots, I(A_{\sigma(k)}) \rangle$$

where  $\sigma$  is the unique permutation s.t.  $\sigma(i) \leq \sigma(j)$  whenever  $\text{enum}(A_i) \leq \text{enum}(A_j)$ . Given a typed term  $x_1 : \tilde{A}_1, \dots, x_n : \tilde{A}_n \vdash M : A$  we associate to it a resource term  $x_1 : I(\tilde{A}_1)^{\vec{n}_1^M}, \dots, x_n : I(\tilde{A}_n)^{\vec{n}_n^M} \vdash \text{qt}(M) : I(A)$  with  $\vec{n}_i^M$  being list of integers of the same length as  $I(\tilde{A}_i)$ , called its *coarse approximation*, by a straightforward extension of the embedding of simple types. The embedding, again, preserves  $\beta$ -reduction. We then get strong normalization of intersection typed terms as a corollary.

We observe that if  $M$  is typed in the idempotent system, we get that  $\text{qt}(M)$  is a qualitative term. However,  $\text{qt}(M)$  is not strongly uniform as in the simply-typed case. (Consider for instance,  $M = \lambda x. xx$ .) We then obtain the following characterization:

**Proposition 6.**  $\gamma \vdash s : a$  is a qualitative term iff there exists  $\Gamma \vdash M : A$  s.t.  $\text{qt}(M) = s$ .

### B. The Extensional Collapse, Operationally

Our rewriting system allows to explicitly connect linear and non-linear approximations of ordinary  $\lambda$ -terms. Another way of establishing the connection has been pursued in the context of the *relational semantics* of linear logic by, e.g., Ehrhard [13]. It is well-established that, in this setting, the

interpretation of a  $\lambda$ -term can be presented by means of *multitypes*, whereas the intersection connective is seen as a multiset  $[A_1, \dots, A_k]$ . In particular, we can present both the traditional relational semantics and the Scott semantics in this way (cfr. [13]). Given an untyped term  $M$  with  $\text{fv}(M) \subseteq \vec{x}$ , its denotation is a monotonic relation  $\llbracket M \rrbracket_{\clubsuit} \subseteq D^{\text{len}(\vec{x})} \times D$  defined as

$$\llbracket M \rrbracket_{\clubsuit} = \{(\Gamma, A) \mid \Gamma \vdash_{\clubsuit} M : A\}$$

for  $\clubsuit \in \{\text{scott}, \text{rel}\}$  and  $D$  is (the underlying carrier of) an appropriate preorder on multitypes. The judgment  $\Gamma \vdash_{\clubsuit} M : A$  is obtained through an intersection type system, that is idempotent (resp. non-idempotent) in the Scott semantics (resp. relational semantics). The *extensional collapse of non-idempotent intersection types* is the equality

$$\llbracket M \rrbracket_{\text{scott}} = \downarrow \llbracket M \rrbracket_{\text{rel}} := \{(\Gamma, A) \mid \text{there exists } (\Gamma', B) \in \llbracket M \rrbracket_{\text{rel}}$$

$$\text{s.t. } \Gamma \leq_{\text{scott}} \Gamma', B \leq_{\text{scott}} A\}$$

where  $A \leq_{\text{scott}} B$  is a preorder on intersection types generated by the rules:  $[A] \leq [A, A]$ ,  $[A_1, A_2] \leq [A_i]$  and  $[A_1, \dots, A_k] \leq []$ . The result has been proved by Ehrhard semantically [13]. We can here give an operational proof. If we endow the structural resource  $\lambda$ -calculus with the following subtyping rule:

$$\frac{x_1 : \vec{a}_1, \dots, x_n : \vec{a}_n \vdash s : b \quad f_i : \vec{b}_i \rightarrow \vec{a}_i}{x_1^{f_1} : \vec{b}_1, \dots, x_n^{f_n} : \vec{b}_n \vdash_{\text{sub}} s : a}$$

we can characterize both forms of semantics by exploiting our calculus. We defined the approximation relation as

$$\llbracket M \rrbracket_{\clubsuit} = \{(\underline{\gamma}, \underline{a}) \mid \gamma \vdash_{\clubsuit} s : a \text{ for some } s \triangleleft M\} \quad (3)$$

where  $\spadesuit = c$  (resp.  $l$ ) if  $\clubsuit = \text{scott}$  (resp.  $\text{rel}$ ) and  $\underline{a}$  is obtained from  $a$  by replacing any list with the corresponding multiset. We can then rephrase Ehrhard's result into our setting, obtaining an operational and proof-relevant version of the collapse, bridging categorical constructions and our rewriting system. We can perform the collapse for both typed and untyped  $\lambda$ -calculi. Given a  $\lambda$ -term  $M$ , we set

$$\text{Appr}(M)_{\spadesuit}(\gamma; a) = \{s \in \Lambda_r \mid s \triangleleft M \text{ and } \gamma \vdash_{\spadesuit} s : a\}.$$

Thanks to the normalization and confluence result for our exponential reduction, we have that the map  $s \mapsto \text{nf}^e(s)$  defines the function:

$$\text{Appr}(M)_{\spadesuit}^{\eta}(\gamma; a) \rightarrow \sum_{\delta, b} \text{Appr}(M)_{\spadesuit}^{\eta}(\delta; b) \times \text{Ctx}(\gamma, \delta) \times \mathcal{RT}(b, a)$$

where  $\text{Appr}(M)_{\spadesuit}^{\eta}(\gamma; a)$  denotes the set of  $\eta$ -long forms of approximations of  $M$ . We call this function the *proof-relevant collapse* of linear intersection types into cartesian ones. From this, we can give an alternative proof of the semantic collapse.

**Theorem 7.** *Let  $M$  be a  $\lambda$ -term. We have that*

$$\llbracket M \rrbracket_{\text{scott}} = \downarrow \llbracket M \rrbracket_{\text{rel}}.$$

*Proof.* ( $\subseteq$ ) Let  $(\Gamma, A) \in \llbracket M \rrbracket_{\text{scott}}$ . By (3), there exists  $\gamma \vdash_c s : a$  with  $\Gamma = \underline{\gamma}$ ,  $A = \underline{a}$ . We consider a normalization rewriting  $(\gamma \vdash_l s : a) \rightarrow_{\theta; f} (\gamma' \vdash_l \text{nf}^e(s) : a')$ . Now, again by (3) we have that  $(\underline{\gamma}, \underline{a}) \in \llbracket M \rrbracket_l$ . Since  $\theta : \gamma \rightarrow \gamma'$  and  $f : a' \rightarrow a$  we can infer that  $\underline{\gamma} \leq_{\text{scott}} \underline{\gamma'}$  and  $\underline{a'} \leq_{\text{scott}} \underline{a}$ . Then we can conclude that  $(\Gamma, A) \in \downarrow \llbracket M \rrbracket_{\text{rel}}$ .  $\square$

## VIII. RELATED WORK

Linearization — seen as a program transformation aiming at bringing  $\lambda$ -terms into a linear form — was first considered by Kfoury [25], whose notion of linearization does not eliminate structural rules *on demand* but somehow *blindly*, resulting in a notion of reduction which is neither strongly normalizing nor confluent. This contrasts with the structural resource  $\lambda$ -calculus, in which instead linearization and  $\beta$ -reduction commute (cfr. Theorem 8). Alves and Florido [1], [2] consider a *weakly* linear fragment of the  $\lambda$ -calculus and a transformation of all terms of the ordinary  $\lambda$ -calculus into this fragment which preserves normal forms; the transformation, however, does not occur internally to the calculus, being based on Levy's legal paths. More recently, Alves and Ventura [3] revisit weakly linear terms by characterizing them in terms of non-idempotent intersection types, this way simplifying the translation. Non-idempotent typing being itself a quantitative system, the bottleneck of the translation is somehow transferred to typing.

The work most closely related to ours is certainly the recent one by Pautasso and Ronchi [35]. They present a unification algorithm that, given a simply typed  $\lambda$ -term  $M$ , produces a typing for  $M$  in a restricted (uniform) non-idempotent

intersection type system. We obtain a similar result as a corollary of our work, as shown in Section VII-A. However, our result is rewriting-based, and stems from structural rule elimination, not unification. While the typing they obtain is affine, ours is linear, by normalization of the exponential reduction. Moreover, while they need strong normalization of simply-typed terms to prove the correctness of their algorithm, we prove it independently, as a corollary of strong normalization.

The Taylor expansion as defined by Ehrhard and Regnier [14] can itself be seen as a form of linearization. It consists of assigning to every  $\lambda$ -term, seen as an *analytic function*, its Taylor expansion, i.e., the *infinite* sum of its linear approximations, defined as resource terms. Only a finite number of such approximations (or one, in the rigid case) are correct. In this sense, what we do in this paper can be seen as a way to build such an approximation *from within* a calculus. A related approach is the one of *polyadic approximations* [29], whereas both linear and non-linear approximations are allowed. The type of approximation is closely related to the structural rules allowed. Our structural resource  $\lambda$ -calculus can be seen as a further refinement of that framework, where *nested* structural rules are allowed.

While we presented our results without relying on abstract constructions, our work has been strongly influenced by the 2-dimensional categorical semantics of the  $\lambda$ -calculus. In particular, the definition of structural resource  $\lambda$ -terms, the action of morphisms and their operational semantics come from a fine-grained analysis of the semantics of  $\lambda$ -terms in the bicategory of *generalized species of structures* [15], [16], [24], [32], [39], [40].

## IX. CONCLUSION

In this paper, we introduced the structural resource  $\lambda$ -calculus with its terminating and confluent rewriting system. Two reduction relations are present, namely the *exponential* one, which performs linearization by structural rule elimination, and the *linear* one, which instead performs linear  $\beta$ -reduction. We have shown how to embed qualitative type systems into our calculus and how to recover a classic result in the denotational semantics of linear logic, namely the *extensional collapse*.

We believe that the structural resource  $\lambda$ -calculus can be suitably extended to encode other qualitative systems of interest. As a first goal we will work on obtaining an encoding of Gödel's  $\mathcal{T}$ , then targeting dependent types and polymorphism.

We have shown how the reduction semantics of some qualitative systems can be factorized into exponential and linear steps. Since the normalization of linear reduction holds by a simple combinatorial argument, the proof of normalization for those qualitative systems depends on the well-foundedness of exponential reduction, which we proved by a suitable reducibility argument. Would it be possible to obtain a more combinatorial proof? A positive answer to this question could also shed some new light on the so called Gödel's Koan [30],

that is about finding natural measures that decrease under reduction of typed  $\lambda$ -terms.

From the point of view of denotational semantics, we plan to investigate the relationship of our linearization with a 2-dimensional version of the extensional collapse. We speculate that the normalization function  $s \rightarrow \text{nf}^e(s)$  could be the syntactic presentation of a natural isomorphism expressing a *proof-relevant* version of the collapse, arising from a 2-dimensional orthogonality construction [17], [18]. The interactive flavour of our rewriting system also suggests a strong connection with *game semantics*. In that context, it would be interesting to relate our construction to the collapse of concurrent games [9]. From this, we could then hope to exploit the connection between generalized species and concurrent games established in [10] to formalize the relationship between the two collapse results.

## REFERENCES

- [1] Sandra Alves and Mário Florido. Linearization by program transformation. In *Proc. of LOPSTR 2003*, volume 3018 of *LNCS*, pages 160–175. Springer, 2003.
- [2] Sandra Alves and Mário Florido. Weak linearization of the lambda calculus. *Theor. Comput. Sci.*, 342(1):79–103, 2005.
- [3] Sandra Alves and Daniel Ventura. Quantitative weak linearisation. In *Proc. of ICTAC 2022*, volume 13572 of *LNCS*, pages 78–95. Springer, 2022.
- [4] Steffen Van Bakel. Strict intersection types for the lambda calculus. *ACM Comput. Surv.*, 2011.
- [5] Davide Barbarossa and Giulio Manzonetto. Taylor subsumes scott, berry, kahn and plotkin. *Proc. ACM Program. Lang.*, (POPL), 2020.
- [6] Henk Barendregt, Mario Coppo, and Mariangiola Dezani-Ciancaglini. A filter lambda model and the completeness of type assignment. *Journal of Symbolic Logic*, 48(4):931–940, 1983.
- [7] Lison Blondeau-Patissier, Pierre Clairambault, and Lionel Vaux Auclair. Strategies as resource terms, and their categorical semantics. In Marco Gaboardi and Femke van Raamsdonk, editors, *8th International Conference on Formal Structures for Computation and Deduction, FSCD, 2023*.
- [8] Gérard Boudol. The lambda-calculus with multiplicities. In *Proc. of CONCUR 1993*, pages 1–6. Springer, 1993.
- [9] Pierre Clairambault. The qualitative collapse of concurrent games. *CoRR*, abs/2410.11389, 2024.
- [10] Pierre Clairambault, Federico Olimpieri, and Hugo Paquet. From thin concurrent games to generalized species of structures. In *38th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS, 2023*.
- [11] Vincent Danos and Laurent Regnier. The structure of multiplicatives. *Arch. Math. Log.*, 28(3):181–203, 1989.
- [12] Daniel de Carvalho. Execution time of  $\lambda$ -terms via denotational semantics and intersection types. *Math. Struct. Comput. Sci.*, 28(7):1169–1203, 2018.
- [13] Thomas Ehrhard. The Scott model of linear logic is the extensional collapse of its relational model. *Theoretical Computer Science*, 2012.
- [14] Thomas Ehrhard and Laurent Regnier. Uniformity and the taylor expansion of ordinary lambda-terms. *Theor. Comput. Sci.*, 403(2-3):347–372, 2008.
- [15] Marcelo Fiore, Nicola Gambino, Martin Hyland, and Glynn Winskel. The cartesian closed bicategory of generalised species of structures. *Journal of the London Mathematical Society*, 2008.
- [16] Zeinab Galal. A Profunctorial Scott Semantics. In *5th International Conference on Formal Structures for Computation and Deduction, FSCD, 2020*.
- [17] Zeinab Galal. *Bicategorical Orthogonality Constructions for Linear Logic*. PhD thesis, Université Paris Cité, 2021.
- [18] Zeinab Galal. Proof-relevant normalization for intersection types with profunctors. In *TYPES, 2022*.
- [19] Philippa Gardner. Discovering needed reductions using type theory. In *Proc. of TACS 1994*, volume 789 of *Lecture Notes in Computer Science*, pages 555–574. Springer, 1994.
- [20] Jean-Yves Girard. Une extension de L’interprétation de Gödel à l’analyse, et son application à l’élimination des coupures dans l’analyse et la théorie des types. 63:63–92, 1971.
- [21] Jean-Yves Girard. Linear logic. *Theor. Comput. Sci.*, 50:1–102, 1987.
- [22] Jean-Yves Girard. Light linear logic. *Inf. Comput.*, 143(2):175–204, 1998.
- [23] Jean-Yves Girard, Andre Scedrov, and Philip J. Scott. Bounded linear logic: A modular approach to polynomial-time computability. *Theor. Comput. Sci.*, 97(1):1–66, 1992.
- [24] Axel Kerinec, Giulio Manzonetto, and Federico Olimpieri. Why are proofs relevant in proof-relevant models? *Proc. ACM Program. Lang.*, (POPL), 2023.
- [25] A. J. Kfoury. A linearization of the lambda-calculus and consequences. *J. Log. Comput.*, 10(3):411–436, 2000.
- [26] J. L. Krivine. *Lambda-calculus, types and models*. Ellis Horwood, USA, 1993.
- [27] Yves Lafont. Soft linear logic and polynomial time. *Theor. Comput. Sci.*, 318(1-2):163–180, 2004.
- [28] Per Martin-Löf. *Intuitionistic type theory*, volume 1 of *Studies in proof theory*. Bibliopolis, 1984.
- [29] Damiano Mazza, Luc Pellissier, and Pierre Vial. Polyadic approximations, fibrations and intersection types. In *Proc. ACM Program. Lang.*, POPL, 2018.
- [30] TLCA List of Open Problems. Problem 26.
- [31] Federico Olimpieri. *Intersection Types and Resource Calculi in the Denotational Semantics of Lambda-Calculus*. PhD thesis, Aix-Marseille Université, 2020.
- [32] Federico Olimpieri. Intersection type distributors. In *Proceedings of the 36th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS, 2021*.
- [33] Federico Olimpieri and Lionel Vaux Auclair. On the taylor expansion of  $\lambda$ -terms and the groupoid structure of their rigid approximants. *Log. Methods Comput. Sci.*, 2022.
- [34] Michel Parigot. Lambda-mu-calculus: An algorithmic interpretation of classical natural deduction. In *Proc. of LPAR 1992*, volume 624 of *Lecture Notes in Computer Science*, pages 190–201. Springer, 1992.
- [35] Daniele Pautasso and Simona Ronchi Della Rocca. A quantitative version of simple types. In *Proc. of FSCD 2023*, volume 260 of *LIPIcs*, pages 29:1–29:21, 2023.
- [36] John C. Reynolds. Towards a theory of type structure. In *Proceedings of Colloque sur la Programmation*, volume 19 of *LNCS*, pages 408–423. Springer, 1974.
- [37] W. W. Tait. Intensional interpretations of functionals of finite type i. *Journal of Symbolic Logic*, 32(2):198–212, 1967.
- [38] Terese. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.
- [39] Takeshi Tsukada, Kazuyuki Asada, and C.-H. Luke Ong. Generalised species of rigid resource terms. In *Proceedings of the 32nd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS, 2017*.
- [40] Takeshi Tsukada, Kazuyuki Asada, and C.-H. Luke Ong. Species, profunctors and taylor expansion weighted by smcc: A unified framework for modelling nondeterministic, probabilistic and quantum programs. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS, 2018*.

## APPENDIX

We sketch some of the proofs of the main results. Proofs are divided thematically.

### A. Typing

**Proposition 7** (Uniqueness of Type Derivations). *Let  $\pi$  be a derivation of  $\gamma \vdash s : a$  and  $\pi'$  of  $\gamma \vdash s : a'$ . Then  $a = a'$  and  $\pi = \pi'$ .*

*Proof.* By induction on  $s$ . The only interesting cases are the application and list terms. We prove the application case, the list one being completely analogous. Let  $s = p\vec{q}$  and  $\pi =$

$$\frac{\gamma_0 \vdash p : \vec{a} \multimap a \quad \gamma_1 \vdash \vec{q} : \vec{a}}{\gamma_0 \otimes \gamma_1 \vdash p\vec{q} : a}$$

and  $\pi' =$

$$\frac{\gamma'_0 \vdash p : \vec{a}' \multimap a \quad \gamma'_1 \vdash \vec{q}' : \vec{a}}{\gamma_0 \otimes \gamma_1 \vdash p\vec{q}' : a}$$

with  $\gamma_0 \otimes \gamma_1 = \gamma'_0 \otimes \gamma'_1$ . We want to prove that  $\gamma_i = \gamma'_i$ . Let  $\gamma_i = x_1 : \vec{a}_1, \dots, x_n : \vec{a}_n$  and  $\gamma'_i = x_1 : \vec{a}'_1, \dots, x_n : \vec{a}'_n$ . The context is relevant, meaning that  $\vec{a}_i \oplus \vec{a}'_i$  contains exactly the typing of all occurrences of  $x_i$  in  $p\vec{q}'$ , ordered from the leftmost to the rightmost. Hence the same is true for  $\gamma_i, \gamma'_i$  and then we can conclude.  $\square$

**Proposition 8 (Compositionality).** *Let  $\gamma \vdash s : a$  and  $\theta : \delta \rightarrow \gamma, \theta' : \delta' \rightarrow \delta, f : a \rightarrow b, g : b \rightarrow c$ . The following statements hold.*

$$\begin{aligned} (\gamma^{\mu_s^f})^{\mu_{[f]s}^g} \vdash [g]([f]s) : c &= \gamma^{\mu_s^{gf}} \vdash [gf]s : c. \\ (\delta^{\nu_s^{\theta}})^{\nu_s^{\theta\theta'}} \vdash s\{\theta\}\{\nu_s^{\theta\theta'}\} : a &= \delta^{\nu_s^{\theta\theta'}} \vdash s\{\theta\theta'\} : a. \end{aligned}$$

*Proof.* We prove the interceding cases.

1) By induction on  $s$ . If  $s = \lambda\vec{x}^{\vec{h}}.p$  with  $f = \vec{f} \multimap o$  and  $h = \vec{g} \multimap o$ , we have that  $[g]([f]s) = \lambda\vec{x}^{((\vec{h}f)\vec{g})}.p$  and  $[gf]s = \lambda\vec{x}^{\vec{h}(\vec{f}g)}.p$ . We then conclude by associativity of morphism composition.

If  $s = \langle s_1, \dots, s_k \rangle$  with  $f = \langle \alpha; f_1, \dots, f_l \rangle$  and  $g = \langle \beta; g_1, \dots, g_m \rangle$  we have that  $[g]([f]s) = \langle [f_{\beta(1)}g_1]s_{\alpha(\beta(1))}, \dots, [f_{\beta(m)}g_m]s_{\alpha(\beta(m))} \rangle$ . We conclude again by definition of morphism composition and by applying the IH.

2) By induction on  $s$ , exploiting the former result. If  $s = x\vec{q}$  then  $\theta = \vec{f} \multimap o$  and  $\theta' = \vec{f}' \multimap o$ . By IH we have that  $\vec{q}\{\theta\}\{\nu_{\vec{q}}^{\theta\theta'}\} = \vec{q}\{\theta\theta'\}$ . By definition we have that

$$s\{\theta\}\{\nu_s^{\theta\theta'}\} = x[f']([f]\vec{q}\{\theta\}\{\nu_{\vec{q}}^{\theta\theta'}\})$$

and

$$s\{\theta\theta'\} = [f']\vec{q}\{\theta\theta'\}$$

we then conclude by applying the IH and the former point of this lemma.  $\square$

We also observe that covariant and contravariant action are interchangeable:  $[f](s\{\theta\}) = ([f]s)\{\mu_s^f\theta\}$ .

### B. Morphisms under reduction

We study the behaviour of type morphism action under reduction. Given  $x_1 : \vec{a}_1, \dots, x_n : \vec{a}_n \vdash s : a$  and  $\theta : \delta \rightarrow (\vec{a}_1, \dots, \vec{a}_n)$ , we have that  $\nu_s^\theta = \nu_1, \dots, \nu_n$  for appropriate ground morphisms  $\nu_i$ . We denote  $\nu_{x_i, s}^\theta$  the morphism  $\nu_i$ , that is the one acting on the type of  $x_i$ . We will often shorten it to  $\nu_{x_i}^\theta$ , when the remaining information is clear from the context. Given a sequence of list morphisms of the shape  $\langle \rangle, \dots, \langle \alpha; \vec{f} \rangle, \dots, \langle \rangle$  we shall often abuse the language and just denote it by  $\langle \alpha; \vec{f} \rangle$ .

Given a reduction step  $(\gamma \vdash s : a) \rightarrow_{\theta; f} (\gamma' \vdash s' : a')$  and a morphism  $\theta' : \delta \rightarrow \gamma$ , we would expect that we could infer some reduction step the following shape:

$$(\delta^{\nu_s^{\theta'}} \vdash s\{\theta'\} : a) \rightarrow_{\eta; f} (\delta^{\nu_{s'}^{\theta'\theta}} \vdash s'\{\theta\theta'\} : a')$$

for some ground morphism  $\eta$ . However, it is easy to see that this is not the case, due to the contextual rule of reduction for  $\lambda$ -abstraction. We can still recover a convergence though: there exists  $t$  and appropriate morphisms  $\eta_1, \eta_2, i_1, i_2$  s.t.

$$s\{\theta'\} \rightarrow_{\eta_1; i_1} t \leftarrow_{\eta_2; i_2} s\{\theta\theta'\}$$

We formalize this result in the following lemma.

**Lemma 4 (Actions under exp).** *Let  $(\gamma, x : \vec{a} \vdash s : a) \Rightarrow_{\theta; g; f} (\gamma', x : \vec{a}' \vdash s' : a')$  and  $g' : \vec{b} \rightarrow \vec{a}, h : a \rightarrow b$ .*

*Then there exist ground morphisms  $\mu_1, \mu_2$ ,*

$$\zeta_1 : \gamma^{\nu_s^{id; g'} \mu_{s\{id, g'\}}^h} \rightarrow \delta$$

$$\zeta_2 : \gamma^{\nu_{s'}^{id, gg'} \mu_{s'\{id, gg'\}}^{hf}} \rightarrow \delta$$

*and  $i : c \rightarrow b$  and a term  $\delta, x : \vec{b}^{\mu'} \vdash t : c$  s.t.  $[h]s\{id, g'\} \Rightarrow_{\zeta_1, \mu_1; if} t$  and  $[hf]s'\{id, gg'\} \Rightarrow_{\zeta_2, \mu_2; i} t$  with*

$$(\zeta_1, \mu_1)\nu_s^{id, g'} \mu_{s\{id, g'\}}^h = (\zeta_2, \mu_2)\nu_{s'}^{id, gg'} \mu_{s'\{id, gg'\}}^{hf} \theta$$

*Proof.* By induction on the step  $s \Rightarrow s'$ . We prove the interesting cases.

Let

$$\lambda\vec{x}^{\vec{f}'} . s \Rightarrow_{\nu_{\vec{f}'}^{\vec{f}'}, \nu_{\vec{y}}^{\vec{f}'}, \nu_{\vec{x}}^{\vec{f}' \multimap o}} \lambda x.s\{id, id, \vec{f}'\}$$

with  $f = \nu_{\vec{x}}^{\vec{f}'} \multimap o$  and  $\theta = \nu_{\vec{f}'}^{\vec{f}'}$ . We have that  $h$  is of the shape  $\vec{h} \multimap o$ ,  $g = \nu_{\vec{y}}^{\vec{f}'}$  and

$$[h]s\{id, g'\} = \lambda\vec{x}^{\vec{f}'\vec{h}} . s\{id, g', id\}$$

Then

$$[hf]s'\{id, gg'\} = \lambda\vec{x}^{\nu_{\vec{x}}^{\vec{f}'\vec{h}} \vec{h}} . s\{id, \nu_{\vec{y}}^{\vec{f}'} g', \vec{f}'\}$$

By compositionality of the actions and by their interchange, we have the following reductions:

$$\begin{aligned} &[hf]s'\{id, gg'\} \\ &\quad \downarrow \nu_{\nu_{\vec{f}'\vec{h}}^{\vec{f}'\vec{h}} \nu_{\vec{y}}^{g'}, (\nu_{\vec{x}}^{\vec{f}'\vec{h}} \vec{h} \nu_{\vec{x}}^{g'}) ; \nu_{\nu_{\vec{f}'\vec{h}}^{\vec{f}'\vec{h}} \nu_{\vec{x}}^{g'} \multimap o}} \\ &[h]s\{id, g'\} \xrightarrow{\nu_{\vec{f}'\vec{h}}^{\vec{f}'\vec{h}} \nu_{\vec{y}}^{g'}, \nu_{\vec{x}}^{\vec{f}'\vec{h}} \nu_{\vec{y}}^{g'} ; \nu_{\vec{x}}^{\vec{f}'\vec{h}} \nu_{\vec{x}}^{g'} \multimap o} \lambda\vec{x}.s\{id, g', \vec{f}'\vec{h}\} \end{aligned}$$

We set

$$t = \lambda\vec{x}.s\{id, g', \vec{f}'\vec{h}\}$$

$$\zeta_1 = \nu_{\vec{f}'\vec{h}}^{\vec{f}'\vec{h}} \nu_{\vec{y}}^{g'} \quad \zeta_2 = \nu_{\nu_{\vec{f}'\vec{h}}^{\vec{f}'\vec{h}} \nu_{\vec{x}}^{g'}}^{\vec{f}'\vec{h}} \nu_{\vec{y}}^{g'}$$

$$\mu_1 = \nu_{\vec{x}}^{\vec{f}'\vec{h}} \nu_{\vec{y}}^{g'} \quad \mu_2 = \nu_{\vec{x}}^{\vec{f}'\vec{h}} \nu_{\vec{x}}^{g'}$$

We remark that in this case  $\mu_{s\{id,g'\}}^f, \mu_{s\{gg'\}}^{hf}$  are identities, since covariant action on lambda abstraction does not change the context. The equations are then satisfied by compositionality (Lemma 8).

Let  $s = x\bar{q}$  with  $x : \langle \bar{a} \multimap o \rangle \vdash_v x : \bar{a} \multimap o$  and  $\delta, x : \bar{a} \vdash \bar{q} : \bar{a}$ . Then  $s' = x\bar{q}'$  with  $\bar{q} \Rightarrow_{\zeta, \bar{f}} \bar{q}'$ . Since the term has an atomic type, we get  $h = f = id$ . There are two possibilities.

Let  $x = y$ . Then  $g = \langle \bar{f} \multimap o \rangle \oplus \zeta_x$ ,  $g' = \langle \bar{h} \multimap o \rangle \oplus g''$  for some morphism  $g''$  and  $s\{id, g'\} = x[\bar{h}]\bar{q}\{g''\}$ . Also  $s'\{id, gg'\} = x[\bar{h}f]q'\{\zeta_x g''\}$ . By IH we get morphisms  $\eta_1, \eta_2, i$  and a sequence term  $\bar{t}$  s.t.

$$\begin{array}{ccc} [\bar{h}]\bar{q}\{id, g''\} & & [hf]\bar{q}'\{\zeta_x g''\} \\ \eta_1, i\bar{f} \searrow & & \swarrow \eta_2, \bar{i} \\ & \bar{t} & \end{array}$$

then we can infer

$$x[\bar{h}]\bar{q}\{id, g''\} \Rightarrow_{(\langle \bar{i}\bar{f} \multimap o \rangle \oplus \eta_1); id} x\bar{t}$$

and

$$x[hf]\bar{q}'\{\zeta_x g''\} \Rightarrow_{(\langle \bar{i} \multimap o \rangle \oplus \eta_2); o} x\bar{t}$$

Then we set  $\zeta_1, \mu_1 = \langle \bar{i}\bar{f} \multimap o \rangle \oplus \eta_1$  and  $\zeta_2, \mu_2 = \langle \bar{f} \multimap o \rangle \oplus \eta_2$ . The equations are satisfied by applying the IH and compositionality.

If  $x \neq y$ , then the result is a direct consequence of the IH.  $\square$

**Lemma 5** (Associativity of Substitution). *let  $\gamma, x : \bar{a}_1, y : \bar{b} \vdash s : a$ ,  $\delta_1, y : \bar{a}_2 \vdash \bar{t} : \bar{b}$  and  $\delta_2 \vdash \bar{q} : \bar{a}_1 \oplus \bar{a}_2$ . We have that*

$$s\{\bar{t}/x\}\{\bar{q}^{[\sigma_s, \bar{t}]} / y\} = s\{\bar{q}_1/y\}\{(\bar{t}\bar{q}_2/y)\}^{[\sigma_s, \bar{q}_1]} / x\}$$

where  $\bar{q} = \bar{q}_1 \oplus \bar{q}_2$ , the decomposition being the one induced by the typing (Figure 5).

*Proof.* By induction on the structure of  $s$ . We prove the interesting cases.

If  $s = x$  then  $\bar{t} = \langle t \rangle \cdot \bar{t}'$  for some term  $t$  and list term  $\bar{t}'$ , with  $s\{\bar{t}/x\} = t$ . Hence

$$s\{\bar{t}/x\}\{\bar{q}^{[\sigma_s, \bar{t}]} / y\} = t\{\bar{q}/y\}$$

and

$$s\{\bar{q}_1/y\}\{(\bar{t}\bar{q}_2/y)\}^{[\sigma_s, \bar{q}_1]} / x\} = t\{\bar{q}_2/y\} = t\{\bar{q}/y\}$$

since  $s$  does not contain any occurrence of the variable  $y$ .  $\square$

**Lemma 6.** *Let  $\gamma, x : \bar{a} \vdash s : a$  and  $\delta \vdash \bar{t} : \bar{a}$  with  $\theta : \gamma' \rightarrow \gamma$  and  $\theta' : \delta' \rightarrow \delta$ . we have that*

$$s\{\theta, id\}\{\nu_x^\theta \bar{t}\{\theta'\}/x\} = s\{\bar{t}/\bar{x}\}\{\theta \otimes \theta'\}$$

*Proof.* By induction on the structure of  $s$ . We prove the interesting cases.

If  $s = x$  then  $\theta$  is a sequence of terminal morphisms and  $s\{\theta, id\} = x$ . Then the result is immediate since  $s\{\theta, id\}\{\nu_x^\theta \bar{t}/x\} = t$  where we assume that  $\bar{t} = \langle t \rangle \cdot \bar{t}'$  for some term  $t$  and bag  $\bar{t}'$ .

If  $s = \lambda \bar{y} \bar{f}. s'$  we have that

$$s\{\theta, id\}\{\nu_x^\theta \bar{t}\{\theta'\}/x\} = \lambda y^{\nu_y^\theta}. s'\{\theta, id, id\}\{\nu_x^\theta \bar{t}\{\theta', id\}/x\}$$

and

$$s\{\bar{t}/\bar{x}\}\{\theta \otimes \theta'\} = \lambda y^{\nu_y^{\langle \theta \otimes \theta' \rangle}}. s'\{\bar{t}/\bar{x}\}\{(\theta, id) \otimes (\theta', id)\}$$

By the IH we have that

$$s'\{\theta, id, id\}\{\nu_x^\theta \bar{t}\{\theta', id\}/x\} = s'\{\bar{t}/\bar{x}\}\{(\theta, id) \otimes (\theta', id)\}$$

we can then conclude.

If  $s = p\bar{q}$  then  $\gamma = \gamma_1 \otimes \gamma_2$  and  $\bar{a} = \bar{a}_1 \oplus \bar{a}_2$  with  $\gamma_1, x : \bar{a}_1 \vdash p : \bar{b} \multimap a$  and  $\gamma_2, x : \bar{a}_2 \vdash \bar{q} : \bar{b}$ , for some contexts  $\gamma_1, \gamma_2$ , intersection types  $\bar{a}_1, \bar{a}_2, \bar{b}$ . By definition we have that

$$\begin{aligned} s\{\theta, id\}\{\nu_x^\theta \bar{t}\{\theta'\}/x\} = \\ p\{\theta_1, id\}\{\nu_x^{\theta_1} \bar{t}_1\{\theta'_1\}/x\} \bar{q}\{\theta_2\}\{\nu_x^{\theta_2} \bar{t}_2\{\theta'_2\}/x\} \end{aligned}$$

where  $\theta = \theta_1 \otimes \theta_2$  and  $\theta' = \theta'_1 \otimes \theta'_2$ , the decomposition being the unique one induced by the typing. We then conclude by applying the IH.  $\square$

**Lemma 7.** *Let  $\gamma, x : \bar{a} \vdash s : a$  and  $\delta \vdash_q \bar{q} : \bar{a}$ . The following statements hold.*

- 1) *If  $s \rightarrow_{\theta, g; f} s'$  then  $s\{\bar{q}/x\} \rightarrow_{\theta'; f}^* u \cong s'\{id; g\}\{\nu' \bar{q}/x\}$  s.t.  $\sigma_{s'\{id; g\}, [\nu'] \bar{q}}(\nu \otimes \nu'^*) = \theta' \sigma_{s, \bar{q}}$ .*
- 2) *if  $\bar{q} \rightarrow_{\theta; \langle id; \bar{f} \rangle} \bar{q}'$  then  $s\{\bar{q}/x\} \rightarrow_{\theta'; f}^* u \cong s\{\langle id; \bar{f} \rangle\}\{\bar{q}'/x\}$  s.t.  $\sigma_{s\{\langle id; \bar{f} \rangle\}, \bar{q}'}(\nu \otimes \theta) = \theta' \sigma_{s, \bar{q}}$ .*

*Proof.* 1) By induction on  $s \rightarrow_{\theta, g; f} s'$ . Let  $s = (\lambda \bar{x}. p) \bar{q}$  and  $s' = p\{\bar{q}/\bar{x}\}$ . We conclude by applying Lemma 5.

Let  $s = \lambda \bar{y} \bar{f}. p$  and  $s' = \lambda \bar{y}. p\{id, \bar{f}\}$ . Then  $s\{\bar{t}/x\} = \lambda \bar{y}^{\sigma \bar{f}}. p\{\bar{t}/x\}$  and  $s'\{id; g\}\{\nu' \bar{q}/x\} = \lambda \bar{y}^{\sigma'}. p\{id, \bar{f}\}\{\nu' \bar{q}/x\}$ . we then have that  $s\{\bar{t}/x\} \rightarrow \lambda \bar{y}. p\{\bar{t}/x\}\{id, \sigma' \bar{f}\}$ . We then conclude by Lemma 6.

2) By induction on the structure of  $s$ .  $\square$

**Remark 6.** We observe that given a linear step  $s \Rightarrow_{\theta; f} s'$  the morphism  $\theta$  is a ground permutation and  $f = id$ . Hence, linear reduction almost satisfy subject reduction, up to a permutation of the typing context.

**Lemma 8.** *Let  $(\gamma \vdash s : a) \Rightarrow_{\sigma; id} (\gamma' \vdash s' : a)$  and  $\theta' : \delta \rightarrow \gamma, f' : a \rightarrow b$ . The following statements hold.*

- 1) *There exists a morphism  $\zeta$  s.t.  $(\gamma^{\mu_s^f} \vdash [f]s : b) \Rightarrow_{\zeta; id} (\gamma'^{\mu_{s'}} \vdash [f]s' : b)$  and  $\zeta \mu_s^f = \mu_{s'}^f \theta$ .*
- 2) *There exists a morphism  $\zeta$  s.t.  $(\delta^{\nu_s^\theta} \vdash s\{\theta\} : a) \Rightarrow_{\zeta; id} (\delta^{\nu_{s'}^\theta} \vdash s'\{\sigma \theta\} : a)$  and  $\zeta \nu_s^\theta = \nu_{s'}^{\sigma \theta}$ .*

**C. Reducibility**

**Lemma 9.** *SN is saturated.*

*Proof.* We have to prove that given  $s\{id; \bar{f}g\} \in \text{SN}$  with  $\bar{q} \in \text{SN}$  and  $\bar{q} \rightarrow_{\theta; \bar{f}} \bar{q}'$  then  $(\lambda \bar{x} \bar{f}. s) \bar{q} \in \text{SN}$ . First, we observe that if  $s\{id; \bar{f}g\} \in \text{SN}$ , then, in particular  $s \in \text{SN}$ . Otherwise, by Lemma 4, from an infinite chain for  $s$  we could build one for  $s\{id; \bar{f}g\}$ . From this we can infer that  $\lambda \bar{x} \bar{f}. s \in \text{SN}$ . We

observe that the only other possible reductions for  $(\lambda\bar{x}^{\bar{g}}.s)\bar{q}$  must come from  $\bar{q}$ . By hypothesis, we can then conclude.  $\square$

**Lemma 10.** *We have that  $\text{Red}(a) \subseteq \text{SN}$ .*

*Proof.* By induction on the structure of  $a$ . If  $a = o$  then  $\text{Red}(a) = \text{SN}$

If  $a = \bar{a} \multimap o$  then

$$\text{Red}(a) = \{s \mid \text{for all } \bar{q} \in \text{Red}(\bar{a}), s\bar{q} \in \text{Red}(o)\}$$

Hence, since  $s\bar{q} \in \text{Red}(o) = \text{SN}$ , in particular  $s \in \text{SN}$ . The other cases are direct corollary of the IH.  $\square$

**Lemma 11.** *Let  $\gamma \vdash s : a$  and  $\theta : \delta \rightarrow \gamma, f : a \rightarrow b$ . The following statements hold.*

- 1)  $[f]s \in \text{Red}(b)$ .
- 2)  $s\{\theta\} \in \text{Red}(a)$ .

*Proof.* We give the proof for the interesting cases.

- 1) By induction on  $s$ . If  $s = x\bar{q}$ , with  $x : \langle \bar{a} \multimap o \rangle \vdash_v x : \bar{a} \multimap o$  and  $\delta \vdash \bar{q} : \bar{a}$  then  $f = id_o$  and we shall prove that  $x\bar{q} \in \text{Red}(o)$ . By IH we have that  $[\bar{f}]\bar{q} \in \text{Red}(\bar{b})$  for any  $\bar{f} : \bar{a} \rightarrow \bar{b}$ . Hence, in particular,  $[\bar{a}]\bar{q} \in \text{Red}(\bar{a})$  and then by Lemma 10,  $\bar{q} \in \text{SN}$ . Then we can conclude, since if  $\bar{q}$  is strongly normalizing, then also  $x\bar{q}$  is.

Let  $s = \lambda\bar{x}^{\bar{g}}.p$ , with  $\bar{g} : \bar{a} \rightarrow \bar{c}$  and  $\gamma, \bar{x} : \bar{c} \vdash p : o$ . Then  $f = \bar{f} \multimap o$  for some sequence of list morphisms  $\bar{f}$  and we need to prove that  $\lambda\bar{x}^{\bar{g}\bar{f}}.p \in \text{Red}(\bar{b} \multimap o)$ , i.e., for any  $\bar{q} \in \text{Red}(\bar{b})$ , we have that  $(\lambda\bar{x}^{\bar{g}\bar{f}}.p)\bar{q} \in \text{Red}(o)$ . By hypothesis we have that  $s\{id; \theta\} \in \text{Red}(o)$  for any morphism  $\theta$  and by Lemma 10 we have that  $\bar{q} \in \text{SN}$ . Hence, by saturation, we can conclude.

Let  $s = (\lambda\bar{x}^{\bar{g}}.p)\bar{q}$  with  $\gamma, x : \bar{b} \vdash p : o, \delta \vdash \bar{q} : \bar{a}$  and  $\bar{g} : \bar{a} \rightarrow \bar{b}$ . Then  $f = id_o$  and we need to prove that  $(\lambda\bar{x}^{\bar{g}}.p)\bar{q} \in \text{SN}$ . By IH we have that  $[\bar{a} \multimap o]\lambda\bar{x}^{\bar{g}}.p = \lambda\bar{x}^{\bar{g}}.p \in \text{Red}(\bar{a} \multimap o)$  and  $[\bar{a}]\bar{q} = \bar{q} \in \text{Red}(\bar{a})$ . Then, by definition,  $s \in \text{Red}(o)$ .

- 2) By induction on  $s$ , exploiting the former point of this lemma. If  $s = x\bar{q}$  with  $x : \langle \bar{a} \multimap o \rangle \vdash_v x : \bar{a} \multimap o$  and  $\delta \vdash \bar{q} : \bar{a}$  we have that  $\theta = \langle \rangle, \dots, \langle \bar{f} \multimap o \rangle, \dots, \langle \rangle$  with  $\bar{f} : \bar{a} \rightarrow \bar{b}$ . By hypothesis we have that  $[\bar{f}]\bar{q} \in \text{Red}(\bar{b})$ . By Lemma 10, we have that  $[\bar{f}]\bar{q} \in \text{SN}$ . Then we can conclude, since if  $\bar{q}$  is strongly normalizing, then also  $x\bar{q}$  is.

$\square$

#### D. Isomorphism and Commutation

**Lemma 12.** *Let  $\gamma \vdash s : a$  and  $\theta : \delta \rightarrow \gamma, f : a \rightarrow b$ . The following statements hold.*

- 1) If  $s \cong s'$  then  $[f]s \cong [f]s'$ .
- 2) If  $s \cong_{\sigma} s'$  then  $s\{\theta\} \cong s'\{\sigma\theta\}$ .

*Proof.* By induction on  $s$ .  $\square$

**Lemma 13.** *Let  $s \cong s'$  and  $s \Rightarrow p$  (resp.  $s \Rightarrow p$ ). Then there exists a term  $p'$  s.t.  $s' \Rightarrow p'$  (resp.  $s' \Rightarrow p'$ ) and  $p \cong p'$ .*

*Proof.* By induction on  $s \Rightarrow p$  (resp.  $s \Rightarrow t$ ). We prove the interesting cases. Let  $s = \lambda\bar{x}^{\bar{f}}.t$  and  $p = \lambda\bar{x}.t\{id, \bar{f}\}$ . We have

that  $s \cong s'$  then  $s' = \lambda\bar{x}^{\sigma\bar{f}}.t'$  with  $t \cong t'$  and  $\sigma$  permutation. Then we apply Lemma 12 and conclude. Let  $s = (\lambda\bar{x}^{\bar{f}}.t)\bar{q}$  and  $p = (\lambda\bar{x}^{\bar{f}\bar{g}}.t)\bar{q}'$  with  $\bar{q} \Rightarrow_{\theta; \bar{g}} \bar{q}'$ . Since  $s \cong s'$  then either  $s' = (\lambda\bar{x}^{\sigma\bar{f}}.t')\bar{q}$  with  $t \cong t'$  or  $s' = (\lambda\bar{x}^{\bar{f}}.t)\bar{q}''$  with  $\bar{q} \cong \bar{q}''$ . In both cases we conclude by applying the IH and then contextuality of the reduction.  $\square$

**Lemma 14.** *Let  $\gamma, x : \bar{a} \vdash s : b$  and  $\delta \vdash \bar{t} : \bar{a}$  with  $s\{\bar{t}/x\} \Rightarrow u$ . Then there exist  $p, \bar{q}$  and morphism  $g$  s.t.  $(\lambda x.s)\bar{t} \Rightarrow^* (\lambda x^g.p)\bar{q}$  and  $p\{g\}\{[\nu^g]\bar{q}/x\} \cong u$ .*

*Proof.* We prove the interesting cases. By induction on  $s$ .

If  $s = x\bar{v}$ , we have that  $\bar{t} = \langle t \rangle \cdot \bar{t}'$  and  $s\{\bar{t}/x\} = t\bar{v}\{\bar{t}'/x\}$ . We reason by cases on the step of exponential reduction. Let  $t \Rightarrow_{\theta; \bar{f} \multimap o} t'$ . Then  $s\{\bar{t}/x\} \Rightarrow t[f]\bar{v}\{\bar{t}'/x\}$ . Then we set  $\bar{q}' = \langle t' \rangle \cdot \bar{t}', p = x$  and  $g = f$ . We can conclude since

$$(\lambda x.x\bar{v})\langle t \rangle \cdot \bar{t}' \Rightarrow (\lambda x^f.x\bar{v})\langle t' \rangle \cdot \bar{t}'.$$

Otherwise, we have that  $\bar{q}\{\bar{t}'/x\} \Rightarrow u$ . In that case, we apply the IH and conclude by contextuality.

Let  $s = \lambda\bar{y}^{\bar{f}}.s'$ , with  $\bar{y} \cap \bar{x} = \emptyset$ . Then  $s\{\bar{t}/x\} = \lambda\bar{y}^{\sigma\bar{f}}.s'\{\bar{t}/x\}$ . There are two possible cases. Let

$$s\{\bar{t}/x\} \Rightarrow \lambda\bar{y}.s'\{\bar{t}/x\}\{id; \sigma\bar{f}\}$$

Then we can conclude, since  $s \Rightarrow^* \lambda\bar{y}.s'\{id; \bar{f}\}$  and by Lemma 12 we have that  $s'\{id; \bar{f}\} \cong s'\{id; \sigma\bar{f}\}$ . Otherwise,  $s\{\bar{t}/x\} \Rightarrow \lambda\bar{y}^{\bar{g}\bar{f}}.s''$  with  $s'\{\bar{t}/x\} \Rightarrow_{\theta'; \bar{g}; o} s''$ . BY IH we have a morphism  $\bar{h}$  and terms  $p, \bar{q}$  s.t.  $s'' \cong p\{id; \bar{h}\}\{[\nu^h]\bar{q}/x\}$  and  $(\lambda x.s')\bar{t} \Rightarrow^* (\lambda x^h.p)\bar{q}$ . We can then conclude by contextuality.  $\square$

**Theorem 8 (Commutation).** *Let  $s \Rightarrow^* t \Rightarrow^* t'$  Then there exist terms  $u, u'$  s.t.  $s \Rightarrow^* u \Rightarrow^* u'$  with  $t' \cong u'$ .*

*Proof.* By induction on  $s \Rightarrow^* t$ . We prove the interesting cases.

If  $s = (\lambda\bar{x}.p)\bar{q}$  and  $t = p\{\bar{q}/\bar{x}\}$ , we apply the former lemma and conclude.

If  $s = \lambda\bar{x}^{\bar{f}}.p$  then  $t = \lambda\bar{x}^{\sigma\bar{f}}.p'$  with  $p \rightarrow_{\theta; \sigma; o} p'$  for some permutation  $\theta$  and  $\sigma$ . There are two possible cases. Let  $t' = \lambda\bar{x}.p'\{id; \sigma\bar{f}\}$ . Then  $s \Rightarrow \lambda\bar{x}.p\{id; f\}$  and we conclude by Lemma 8 since  $p\{id; f\} \Rightarrow p'\{id; \sigma f\}$ . Hence  $u' = \lambda x^{\sigma}.p'\{id; \sigma f\}$ .  $\square$

#### E. Confluence

**Theorem 9.** *The structural reduction is locally confluent.*

*Proof.* Given  $s \rightarrow^{\theta_1; f_1} t_1$  and  $s \rightarrow^{\theta_2; f_2} t_2$  we have to prove that there exists a term  $t$  and morphisms  $\theta'_1, \theta'_2, f'_1, f'_2$  s.t.  $t_1 \rightarrow^{\theta'_1; f'_1} t$  and  $t_2 \rightarrow^{\theta'_2; f'_2} t$ , with  $\theta'_2\theta_2 = \theta'_1\theta_1$  and  $f'_1f'_1 = f_2f'_2$ .

We proceed by induction on  $s \rightarrow^{\theta_1; f_1} t_1$ . Let  $s = \lambda\bar{x}^{\bar{f}}.p$  and  $t_1 = \lambda\bar{x}.p\{id; \bar{f}\}$  with  $\theta_1 = \nu^{id, \bar{f}}, f_1 = \nu_x^{id, \bar{f}}$  and  $s \rightarrow_{\nu^{id, \bar{f}}; \nu_x^{id, \bar{f}} \multimap o} t_1$ . Then we have that  $t_2 = \lambda\bar{x}^{\bar{g}\bar{f}}.p'$  with  $p \rightarrow_{\theta_2; \bar{g}; o} p'$  and then  $f_2 = id$ . We perform the step



$t_2 \rightarrow^{\nu^{id, \bar{g}\bar{f}}; \nu_x^{id, \bar{g}\bar{f}} \circ o} \lambda x.p'\{id; \bar{g}\bar{f}\}$ . Now we need to close the following diagram:

$$\begin{array}{ccc} \lambda \bar{x} \bar{f}.p & \xrightarrow{\nu^{id, \bar{f}}; \nu_x^{id, \bar{f}} \circ o} & \lambda \bar{x}.p\{\nu^{id, \bar{f}}; \bar{f}\} \\ \theta_2; id \downarrow & & \\ \lambda \bar{x} \bar{g}\bar{f}.p' & \xrightarrow{\nu^{id, \bar{f}\bar{g}}; \nu_x^{id, \bar{f}\bar{g}} \circ o} & \lambda x.p'\{\nu^{\bar{f}\bar{g}}; \bar{g}\bar{f}\} \end{array}$$

We can do so by applying Lemma 4: there exist a term  $t$ , morphisms  $\zeta_1, h_1, \zeta_2$  and ground morphisms  $\mu_1, \mu_2$  s.t.

$$\begin{aligned} p\{id; \bar{f}\} &\rightarrow^{\zeta_1, \mu_1; id} t & p'\{id; \bar{g}\bar{f}\} &\rightarrow^{\zeta_2, \mu_2; id} t \\ \lambda x.p\{id; \bar{f}\} &\rightarrow^{\zeta_1; id} \lambda x^{\mu_1}.t \rightarrow_{id; \mu_1 \circ o} \lambda x.t \\ \lambda x.p'\{id; \bar{g}\bar{f}\} &\rightarrow^{\zeta_2; id} \lambda x^{\mu_2}.t \rightarrow_{id; \mu_2 \circ o} \lambda x.t \end{aligned}$$

We can then conclude since, by Lemma 4 again,  $\zeta_1 \nu_s^{id, \bar{f}} = \zeta_2 \nu_{s'}^{id, \bar{g}\bar{f}} \theta_2$  and  $\mu_1 \nu_x^{\bar{f}} = \mu_2 \nu_x^{\bar{f}\bar{g}}$ .

Let  $s = (\lambda \bar{x}.s)\bar{q}$  and  $t_1 = s\{\bar{q}/x\}$ . We remark that the step  $s \rightarrow t_2$  must be induced by either a step of  $s$  or a step of  $\bar{q}$ . Then we conclude by applying Lemma 7.

Let  $s = \lambda \bar{x} \bar{f}.p$  and  $t_1 = \lambda \bar{x} \bar{g}\bar{f}.p'$  with  $p \rightarrow^{\theta; g; o} p'$ . Let  $t_2 = \lambda \bar{x} \bar{h}\bar{f}.p''$  with  $p \rightarrow^{\theta'; h; o} p''$ . By the IH we have that there exists  $t'$  s.t.  $p' \rightarrow^{\zeta_1, g_1; o} t'$  and  $p'' \rightarrow^{\zeta_2, g_2; o} t'$  with  $\zeta \theta = \zeta' \theta'$  and  $g_1 g = g_2 h$ . We then conclude by contextuality of the reduction.

Let  $s = (\lambda \bar{x} \bar{f}.p)\bar{q}$  and  $t_1 = (\lambda \bar{x} \bar{g}\bar{f}.p')\bar{q}$  with  $p \rightarrow^{\theta; g; o} p'$ . there are three possible cases.

If  $p \rightarrow^{\theta'; g'; o} p''$  with  $t_1 = (\lambda \bar{x} \bar{f} \bar{g}'.p'')\bar{q}$  we reason by cases on the second step. If its again a step for  $\lambda \bar{x} \bar{f}.p$  we conclude either by applying the IH or by definition of base exponential step. Otherwise  $\bar{q} \rightarrow^{\theta'; \bar{h}} \bar{q}'$ . In that case, we conclude in this way:

$$\begin{array}{ccc} (\lambda \bar{x} \bar{f}.p)\bar{q} & \xrightarrow{id \otimes \theta'; id} & (\lambda \bar{x} \bar{f} \bar{h}.p)\bar{q}' \\ \theta \otimes id; id \downarrow & & \downarrow \theta \otimes id; id \\ (\lambda \bar{x} \bar{g}\bar{f}.p')\bar{q} & \xrightarrow{id \otimes \theta'; id} & (\lambda \bar{x} \bar{g}\bar{f} \bar{h}.p')\bar{q}' \end{array}$$

If  $\lambda \bar{x} \bar{f}.p \rightarrow \lambda \bar{x}.p\{id, \bar{f}\}$ . Then  $t_1 = (\lambda \bar{x}.p\{id, \bar{f}\})[\nu_x^{id, \bar{f}}]\bar{q}$ . There are two possible cases for the second step. If  $t_2 = (\lambda \bar{x} \bar{g}.t')\bar{q}$  with  $p \rightarrow^{\zeta, \bar{g}; o} t'$ , we conclude by applying Lemma 4. If  $t_2 = (\lambda \bar{x} \bar{f} \bar{g}.p)\bar{q}$  with  $\bar{q} \rightarrow_{\theta; \bar{g}} \bar{q}'$ , we apply again Lemma 4 and conclude.

### F. Embedding of simple types

**Lemma 15.** Let  $\Gamma, x : A \vdash M : B$  and  $\Gamma \vdash N : A$ . We have that

$$\text{qt}(M) \{ \langle \text{qt}(N)^{n_x^M} \rangle / x \} = \text{qt}(N\{M/x\})$$

*Proof.* By induction on  $M$ .

If  $M = x$  then  $\text{qt}(M) = x$  with  $x : \langle \text{I}(A) \rangle \vdash x : \text{I}(A)$ . Hence  $n_x^M = 1$  and we can conclude since  $M\{N/x\} = N$ .

If  $M = \lambda y.M$  with  $\Gamma, x : A, y : B \vdash M' : C$ , we have that  $\text{qt}(M) = \lambda y. \text{cart}_{\text{I}(B)}^n \text{qt}(M')$  with  $\text{I}(\Gamma)^{\bar{n}^M}, x : \text{I}(A)^{m^M}, y : \text{I}(B)^{n^M} \vdash \text{qt}(M') : \text{I}(C)$  ad  $\text{cart}_{\text{I}(B)}^n : \text{I}(B) \rightarrow \text{I}(B)^{n^M}$ . By definition  $M\{N/x\} = \lambda y.M'\{N/x\}$  and by IH we have that  $\text{qt}(M') \{ \langle \text{qt}(M)^{m^M} \rangle / x \} = \text{qt}(M\{N/x\})$ . We can then conclude by applying the IH and observing that for any simple type  $A$  and permutation  $\sigma$ , we have that  $\sigma \text{cart}_{\text{I}(A)}^n = \text{cart}_{\text{I}(A)}^n$ .

If  $M = PQ$  with  $\Gamma, x : A \vdash M : B \Rightarrow C$  and  $\Gamma \vdash N : B$ , we have that  $\text{qt}(M) = \text{qt}(\Gamma)^{\bar{n}^M}, x : \langle \text{I}(A) \rangle^{m^P + m^Q} \otimes \text{qt}(\Gamma)^{\bar{n}^N} \vdash \text{qt}(M) \langle \text{qt}(N) \rangle : \text{I}(C)$ . We have that  $M\{N/x\} = P\{N/x\}Q\{N/x\}$ . By IH we get that  $\text{qt}(P) \{ \langle \text{qt}(N)^{m^P} \rangle / x \} = \text{qt}(P\{N/x\})$  and that  $\text{qt}(Q) \{ \langle \text{qt}(N)^{m^Q} \rangle / x \} = \text{qt}(Q\{N/x\})$ . We conclude again then by applying the IH and observing that for any simple type  $A$  and permutation  $\sigma$ , we have that  $\sigma \text{cart}_{\text{I}(A)}^n = \text{cart}_{\text{I}(A)}^n$ .  $\square$