Generalised Species of Rigid Resource Terms

Takeshi Tsukada
University of Tokyo

Kazuyuki Asada
University of Tokyo

C.-H. Luke Ong
University of Oxford

***Abstract*—This paper introduces a variant of the resource calculus, the *rigid resource calculus*, in which a permutation of elements in a bag is distinct from but isomorphic to the original bag. It is designed so that the Taylor expansion within it coincides with the interpretation by generalised species of Fiore et al., which generalises both Joyal's combinatorial species and Girard's normal functors, and which can be seen as a proof-relevant extension of the relational model. As an application, we prove the commutation between computing Böhm trees and (standard) Taylor expansions for a particular nondeterministic calculus.**

I. Introduction

Around the same time as the birth of linear logic [20], Girard [21] proposed the *normal functor semantics* of the λ-calculus. In this semantics, a term is interpreted as a formal power series with set-valued coefficients, i.e., as a functor $X \to \mathbf{Set}$, where the set X is the denotation of a type. This idea has subsequently been studied extensively, leading to new models of linear logic such as *Köthe sequence spaces* [11] and *finiteness spaces* by Ehrhard [12]; and *differential λ-calculus*, a differential calculus for higher-order functions by Ehrhard and Regnier [14]. In a follow-up paper [16], Ehrhard and Regnier introduced the *Taylor expansion* of a λ-term as a formal sum (with rational coefficients) of terms of the *resource calculus*, which may be viewed as a calculus of linear approximants of the λ-terms.

Meanwhile, similarities were noted between Girard's normal functors and Joyal's *combinatorial species* [25], which are functors $\mathbf{P} \to \mathbf{Set}$, where $\mathbf{P}$ is the category of finite cardinals and bijections; Hasegawa [23], amongst others, investigated these connections. There is however an important difference between the two: the domain of (the power series representation of) a normal functor is any set, by which one can interpret a type; whereas a combinatorial species uses $\mathbf{P}$, which has non-trivial (iso)morphisms. Unifying them, Fiore et al. [18, 19] introduced *generalised species of structures* between small categories, whose domains have enough variation to interpret types and have non-trivial (iso)morphisms. The generalised species of structures have been proved to form a cartesian closed bicategory [19], by which one can interpret the λ-calculus. It can be seen as a proof-relevant version of (the Kleisli category of) the relational model of linear logic.

This paper investigates the connections between these lines of thought. Specifically we introduce a variant of the resource calculus, which we call the *rigid resource calculus*, together with the Taylor expansion of a term that uses this calculus, and show that the Taylor expansion coincides with the interpretation of the term by generalised species.

This work, however, did not start as an abstract study of the connection. Instead we addressed other, more concrete, problems, and the generalised species of structures seemed suitable as a tool to solve the problems, as we shall see below. In those problems, isomorphisms play a central rôle.

A. Compositional enumeration of reduction sequences

The first problem is related to the semantics of programming languages with branching constructs, such as nondeterministic, probabilistic, weighted and quantum programming languages (e.g. [8, 17, 22, 34, 39]). This is an important area of current interest.

The operational semantics of these calculi seems to share the same idea. Consider, for example, a programming language with probabilistic branching. Its operational semantics is usually defined by the following steps:

1) Define the set of all reduction sequences $\pi : M \longrightarrow^* V$ where π is the name of this sequence. A program M may have many reduction sequences.
2) Associate to each reduction sequence π the *probability* (or *weight*) $w(\pi)$, a real number between 0 and 1.
3) The probability of a program M being evaluated to a value V is defined as the sum $\sum_{\pi : M \longrightarrow^* V} w(\pi)$ of the weights of reduction sequences π resulting in V.

By choosing the weights and the sum appropriately, this framework applies to languages with other branching structures. For example, for a nondeterministic program, a weight is an element of the two-valued Boolean algebra and the sum is the disjunction (i.e. existence of a reduction sequence). We think that this framework applies also to quantum programs though the structure of weights and the sum are more complicated.

Hence it is natural to develop a denotational model of such a language following the above framework, which semantically enumerates all reduction sequences and then calculates the sum of the weights. Indeed recent quantitative models of probabilistic and quantum languages [8, 17, 34] can be seen as models of this kind. Our ultimate aim is to give a unified account for these models.

The key process of the above framework is the enumeration of reduction sequences. For example, the following program

$$M \;:=\; (\lambda f.f\,(f\,z))\,\underline{((\lambda x.x) \oplus (\lambda y.y))}$$

(where $\oplus$ is nondeterminstic branching and the underlined expression will be referenced later) always reduces to z but in four different ways in call-by-name; indeed

$$M \longrightarrow ((\lambda x.x) \oplus (\lambda y.y))\Big(((\lambda x.x) \oplus (\lambda y.y))z\Big)$$

and a choice of branch for each occurrence of $\oplus$ in the right-hand-side determines an evaluation of the program. If the whole program is given as above, there is no difficulty in enumerating the set of all reduction sequences. However we aim to develop a denotational semantics, which is usually expected to be compositional. Hence we need a way to enumerate reduction sequences *compositionally*.

How can we enumerate individual runs compositionally? We call this problem the *Compositional Enumeration Problem*

B. Key idea: Rigid resource calculus

Let us recall the above example. Observe that every reduction sequence encounters the nondeterministic branch twice, even though there is lexically one occurrence in the original program. This is because the underlined part is duplicated during the evaluation. This suggests to us that a quantitative approach of linear logic (e.g. [11, 12, 21]) may be useful.

From this observation, it is fairly natural to use the resource calculus and the Taylor expansion [16]. The resource calculus is basically the standard λ-calculus except that an argument is not a term, but a finite multiset of terms, called a *bag*, which is linear in the sense that each element in a bag must be used exactly once. Hence the number of elements in a bag corresponds to the number of calls of the argument. The Taylor expansion is a technique representing a program as a (formal) sum of resource terms with real number (or other kinds of) coefficients. For example, the Taylor expansion of the above program M is

$$\begin{array}{rrcl} & 1 & \cdot & (\lambda f.f\,[f\,[z]])\,\underline{[(\lambda x.x)_L,(\lambda y.y)_R]} \\ + & (1/2) & \cdot & (\lambda f.f\,[f\,[z]])\,\underline{[(\lambda x.x)_L,(\lambda x.x)_L]} \\ + & (1/2) & \cdot & (\lambda f.f\,[f\,[z]])\,\underline{[(\lambda y.y)_R,(\lambda y.y)_R]} \end{array}$$

where $(\cdot)_L$ and $(\cdot)_R$ are the marks indicating the chosen branch (which may or may not be a part of the resource calculus).[1] The underlined parts correspond to the underlined part in the original program: they are bags with two elements since the underlined part of the original program is called twice in every possible evaluation.

Unfortunately a resource term cannot describe individual runs. For example, the term $(\lambda f.f\,[f\,[x]])\,[(\lambda x.x)_L,(\lambda y.y)_R]$ reduces to $(\lambda x.x)_L\,[(\lambda y.y)_R\,[z]] + (\lambda y.y)_R\,[(\lambda x.x)_L\,[z]]$, which is the sum of two different evaluations of M. This phenomenon stamps from the fact that a resource term does not specify the connection between elements in a bag and occurrences of a variable (so we should try all possibilities).

To overcome the problem, we consider a calculus in which application arguments are organised into a *list* (as opposed to bag) and abstraction takes a list of variables as in $(\lambda f_1 f_2.f_1\,\langle f_2\,\langle z\rangle\rangle)\,\langle(\lambda x.x)_L,(\lambda y.y)_R\rangle$. The reduction of this calculus is deterministic: we should substitute the first element for the first variable and so on. Now a term of this calculus represents a unique evaluation of the original program.

There is another problem, though: an evaluation of the original program has many different representations. For example,

$$\begin{array}{l} (\lambda f_1 f_2.f_1\,\langle f_2\,\langle z\rangle\rangle)\,\langle(\lambda x.x)_L,(\lambda x.x)_L\rangle \quad \text{and} \\ (\lambda f_2 f_1.f_1\,\langle f_2\,\langle z\rangle\rangle)\,\langle(\lambda x.x)_L,(\lambda x.x)_L\rangle \end{array} \tag{1}$$

both represent the evaluation of the original program in which we choose the left branch twice. This suggests to us that a naïve use of lists cannot solve the problem.

Our approach is a hybrid between bags and lists: the connection between elements in an argument and occurrences of variables are tracked by lists, but terms that describe the same connection are identified.

For example, the above terms are identified since both represent the map given by $f_1 \mapsto (\lambda x.x)_L$ and $f_2 \mapsto (\lambda x.x)_L$. By writing this identification as a relation $\sim$, we have the following more general law:

$$\begin{array}{l} (\lambda x_1 \dots x_n.t)\,\langle u_1,\dots,u_n\rangle \\ \qquad \sim\ (\lambda x_{\sigma(1)} \dots x_{\sigma(n)}.t)\,\langle u_{\sigma(1)},\dots,u_{\sigma(n)}\rangle. \end{array}$$

It is not trivial to define the relation $\sim$: for example,

$$(\lambda f.f\,\langle z_1,z_2\rangle)\,\langle\lambda y_1 y_2.t\rangle\ \sim\ (\lambda f.f\,\langle z_2,z_1\rangle)\,\langle\lambda y_2 y_1.t\rangle \tag{2}$$

since both describe the mapping $y_1 \mapsto z_1$ and $y_2 \mapsto z_2$.

In order to correctly define $\sim$, we use an intersection type system in which $a \wedge b$ is distinct from but isomorphic to $b \wedge a$. Then a permutation $\sigma \in \mathfrak{S}_n$ induces an isomorphism between $a_1 \wedge \cdots \wedge a_n$ and $a_{\sigma(1)} \wedge \cdots \wedge a_{\sigma(n)}$. Key observations are that an isomorphism acts on terms, e.g., σ maps $\langle u_1,\dots,u_n\rangle$ to $\langle u_{\sigma(1)},\dots,u_{\sigma(n)}\rangle$ and that the desired relation $\sim$ can naturally be defined by using the actions of isomorphisms, instead of permutations. Note that an isomorphism may not merely be a permutation; the identification in Equation (2) is induced by an isomorphism that is not just a permutation.

The *rigid resource calculus* is now defined as a list-based resource calculus, in which terms are considered modulo $\sim$. Rigid resource terms enable us to describe individual runs, and enumeration is given by a variant of the Taylor expansion, the *rigid Taylor expansion*, which is compositional.

A significant advantage of the rigid resource calculus is determinism of reduction. It gives us a simple way to compare the rigid Taylor expansions of a term and its Böhm tree (Theorem 17).

C. Reasoning about coefficients of the Taylor expansion

The Taylor expansion of a term is a formal sum of (standard) resource terms with real number coefficients. The properties of its support (i.e. terms with non-zero coefficients) has been well-studied, but reasoning about coefficients of the Taylor expansion is rather difficult. For example, according to [35], it has been open whether the computation of the Böhm tree commutes with the Taylor expansion in the presence of

[1] Here we ignore terms evaluated to 0 such as $(\lambda f.f\,[])\,[]$.

nondeterminism; the commutation property has been proved for deterministic (untyped) λ-calculus in [15, 16], and claimed for System F with weighted branching in [13] (although the proof does not seem to have been published).

The notion of isomorphism appears here as well. Ehrhard and Regnier [16] proved that, if a resource term t has non-zero coefficient in the Taylor expansion of a λ-term M (without nondeterministic branching), the coefficient is $1/\mathsf{m}(t)$ where $\mathsf{m}(t)$ is the order of the *isotropy group* of t.

Inspired by their observation, we express the coefficients by a groupoid of isomorphisms and its action on rigid resource terms, in a way reminiscent of the *generating series* of combinatorial species. A resource term can be seen as an orbit of the action, and the coefficient is the ratio of the number of elements in the orbit to the number of all isomorphims. From this point of view, the rigid resource calculus may be seen as a developed form of the combinatorial concepts studied in [16].

We show that the commutation property holds for the simply-typed nondeterministic $\lambda\mathbf{Y}$-calculus, although coefficients may be infinite.

D. Contribution of this paper

The contributions of the paper are as follows.

- Proposal of the *rigid resource calculus* and the *rigid Taylor expansion*. The rigid Taylor expansion is proved to be sound with respect to Böhm theory, and adequate with respect to the standard operational semantics. It is also a solution to the *Compositional Enumeration Problem*.
- Proof of the coincidence of the rigid Taylor expansion of a term and the interpretation of the term by generalised species of structures [18, 19]. By this result, we obtain two different descriptions of the same mathematical object, one is concrete and syntactic and the other is abstract and categorical.
- Study of the relationship between the rigid and standard resource calculi. This leads us to a proof of the commutation of normalisation and the standard Taylor expansion for the simply-typed nondeterministic $\lambda\mathbf{Y}$-calculus.

To accomplish the program described in Section I-A, we need to introduce the notion of weights to generalised species of structures. The study of weighted generalised species and the construction of a model based on it are left for future work (see also Section VII).

E. Related work

In the preamble, we discussed the main ideas that have motivated our paper. Here we comment on works in the literature relevant to a number of other key themes.

The computational enumeration problem introduced in the preceding is concerned with the analysis of reduction sequences of λ-calculus with branching constructs using quantitative semantics. Versions of this problem have been studied by Danos and Ehrhard [8], and Laird et al. [29], among others.

The notion of symmetry is central to our work. The idea of considering intersection types up to isomorphism was already present in Bucciarelli and Ehrhard's work on the semantics of linear logic exponentials [7], in the guise of *indexed linear logic*, where it was developed in connection with the relational model. We have alluded to the *isotropy group* in the coefficient computation of the Taylor expansion of λ-terms [16]. There are manifestations of the same idea of group action in game semantics. A key ingredient of the AJM game model [1] is a partial equivalence relation (PER) on strategies, which enables the formulation of strategies that are "blind to the Opponent's thread indexing". Melliès [31] subsequently revealed the group-theoretic nature of this PER construction: he introduced *orbital game*, which is a reformulation of HO-style arena games [24] by replacing justification pointers by thread indexing, modulo certain left and right group actions. Symmetry in a similar spirit can be found in a model of quantum computation by Pagani et al. [34]. They construct a model of non-linear quantum computation from a linear model by focusing on morphisms invariant under certain group action.

There are a number of intersection type systems in the literature that are *non-commutative* (in a variety of senses). Thanks to Kfoury and Wells [27, 28], and Neergaard and Mairson [32], and others, it is known that non-commutative, non-idempotent intersection type systems do not enjoy subject reduction. Our approach (and notion of *rigidity*) is different from the above in a crucial way: two syntactically distinct intersections that are equal modulo permutation are considered distinct but isomorphic. This contrasts with, for example, System $\mathbb{I}$ [32] which, being "completely" rigid[2], is thus unsound with respect to the reduction.

First observed by Kfoury [28], the fruitful relationship between (non-idempotent) intersection types and linearity has been a recurrent theme in recent years (see e.g. de Carvalho [10], and Bernadet and Lengrand [5]). Building on this is the idea that intersection type derivations of a λ-term are formally representable by approximants of the term. For example, Ong and Tsukada [33] presented just such a bijective correspondence in a non-linear setting. Our rigid resource calculus may be viewed as a refinement of Mazza's calculus of approximants[3] [30], the derivations of which correspond bijectively to reduction sequences. Another related system is the type system by Ehrhard et al. [17] for calculating the interpretation in the probabilistic coherence space model [8].

Following Kfoury [28], many of the above studies use subject reduction and subject expansion to establish a strong connection between reductions of a given term and intersection type derivations (or approximants) of that term. Our approach gives a refinement: we show that the witnessing type of the reduction (respectively expansion) of a term can always be chosen from among the isomorphic types of the term. This relates to the phenomenon that, in the AJM game model, the

[2]Neergaard and Mairson [32] call an intersection operator, $\wedge$, *rigid* if idempotency, commutativity nor associativity does not hold.

[3]Mazza's intersection type system is not non-commutative in the sense of Kfoury and others [28, 32]. It follows from the only constraint in his system—each assumption be used exactly once—that the order of types (and assumptions) is insignificant. Consequently one can implicitly apply the exchange rule to the assumptions.

interpretation of a term as a strategy is not preserved by β-reduction, but the equivalence class of strategies is.

Our rigid Taylor expansion semantics of the nondeterministic $\lambda\mathbf{Y}$-calculus is closely related to two recent categorical presentations of the HO/N game model. Tsukada and Asada [37] gave a profunctorial reformulation of HO/N games in which plays are graphs: they constructed a pseudofunctor from the category of HO/N games to the bicategory of profunctors. In [39] Tsukada and Ong developed a fully abstract innocent game model for the simply-typed nondeterministic λ-calculus. Mathematically a strategy is formalised as a sheaf over an appropriate site of plays. The precise relationships between the generalised species model and each of the above reformulations of the game model will be developed elsewhere. In a different operational direction, Allioux [2] has recently given an account of the Taylor expansion of algebraic λ-terms based on a kind of resource-constrained Krivine machine.

Notation: We define $[n] := \{1, 2, \ldots, n\}$ for a natural number n (so $[n]$ is a set with n elements and $[0]$ is empty).

II. Problem Setting

This section defines a problem addressed in (the first part of) this paper. The problem is compositional enumeration of all possible evaluations of a given nondeterministic term of a certain type. We choose the simplest calculus to illustrate the ideas of our approach; and we claim that our idea is applicable to PCF and the (untyped) λ-calculus, for example.

Further, the problem is a guide, which serves to justify and explain the design of the syntactic development in Sections III and IV, from which generalised species naturally arise.

A. The target language: Nondeterministic $\lambda\mathbf{Y}$-calculus

The target language is the $\lambda_{\mathrm{nd}}\mathbf{Y}$-*calculus*, the simply-typed $\lambda\mathbf{Y}$-calculus with nondeterministic branching. The syntax of *simple types* is $A, B ::= \mathsf{o} \mid A \to B$ where o is the unique atomic type. The syntax of *terms* is given by the following grammar:

$$M, N := x \mid \lambda x^A.M \mid M\,M \mid \mathbf{Y}_A\,M \mid M \oplus_A M.$$

Here $\mathbf{Y}_A$ is the fixed-point operator of type $(A \to A) \to A$ and $M \oplus_A N$ is the nondeterministic branching where M and N are terms of type A. We identify α-equivalent terms. Type annotations are often omitted.

A *simple type environment* Γ is a finite sequence of type bindings of the form $x : A$. The typing rules are fairly straightforward.

A term M is η-*long* if every application in M is fully-applied and every variable as well as $\mathbf{Y}$ occur with its arguments. Formally it is defined by the following grammar:

$$R ::= \lambda x_1.\lambda x_2.\ldots.\lambda x_n.Q \mid R \oplus R'$$
$$Q ::= x\,R_1\,\ldots\,R_n \mid R_0\,R_1\,\ldots\,R_n \mid \mathbf{Y}\,R_1\,\ldots\,R_n \mid Q \oplus Q'$$

where $n \geq 0$ and the type of Q must be o. Every term can be transformed into an η-long form in the obvious way. Hence we can assume without loss of generality that a term is in η-long form. This assumption simplifies the statements of lemmas and theorems, and will be used often (but always explicitly).

B. Evaluation

The calculus is call-by-name. The syntax of *evaluation contexts* is given by: $E ::= [] \mid E\,M$. The *one-step evaluation relation* is given by the following rules:

$$E[M_1 \oplus M_2] \overset{L}{\hookrightarrow}_1 E[M_1] \qquad E[(\lambda x.M)\,N] \overset{0}{\hookrightarrow}_1 E[\{N/x\}M]$$
$$E[M_1 \oplus M_2] \overset{R}{\hookrightarrow}_1 E[M_2] \qquad E[\mathbf{Y}\,M] \overset{\epsilon}{\hookrightarrow}_1 E[M\,(\mathbf{Y}\,M)]$$

where ϵ is the empty sequence. For a sequence $\pi \in \{L, 0, R\}^*$, we write $M \overset{\pi}{\hookrightarrow} M'$ if there exists an evaluation sequence $M = M_0 \overset{d_1}{\hookrightarrow}_1 M_1 \overset{d_2}{\hookrightarrow}_1 \cdots \overset{d_n}{\hookrightarrow}_1 M_n = M'$ and $\pi = d_1 d_2 \ldots d_n$ (where $n \geq 0$). The length of π is the number of β- and $\oplus$-rules in the sequence; evaluation of $\mathbf{Y}$ is not recorded in π. Evaluation preserves η-longness: if $M \overset{\pi}{\hookrightarrow} M'$ and M is η-long, then M' is also η-long.

C. Compositional Enumeration Problem

Sections III and IV address the following problem.

Problem. Give a compositional semantics of the $\lambda_{\mathrm{nd}}\mathbf{Y}$-calculus such that the interpretation of a term $x_1 : \mathsf{o}, \ldots, x_n : \mathsf{o} \vdash M : \mathsf{o}$ can be seen as the set of *all its evaluation sequences*.

Here the set of evaluation sequences means something like $\{(\pi, x_i) \mid M \overset{\pi}{\hookrightarrow} x_i\}$, although it is not necessarily exactly the same. A satisfactory information on the set depends on the purpose. For example, if we aim to model a language with probabilistic binary branching which chooses each branch with probability $1/2$, the number of L and R in π is important but that of 0 is not. For another example, if the language has a weighted branching $(\alpha\,M) \oplus (\beta\,N)$ where weights α and β are taken from a given ring, we need additional information of weights for each L and R symbols.

We would like to emphasise that solving the above problem is *not* the ultimate goal of our development. We are looking for a *good* way to solve the problem, which would be applicable to other settings, which would be useful in proving an interesting result, and which would provide us with new insights. The solution presented in Sections III and IV is justified by a strong semantics background studied in Section V.

D. Full reduction and Böhm tree

Directly reasoning about the set of evaluation sequences is a bit messy. This subsection introduces another operational semantics, in which the set of all evaluation sequences in the previous subsection are put into the result (i.e. its Böhm tree).

The *(full) reduction relation* has the form $M \longrightarrow M'$. It is defined by the following base cases

$$(\lambda x.M)\,N \longrightarrow \{N/x\}M \qquad \mathbf{Y}\,M \longrightarrow M\,(\mathbf{Y}\,M)$$
$$(M_1 \oplus M_2)\,N \longrightarrow (M_1\,N) \oplus (M_2\,N)$$

and the context rule: $M \longrightarrow M'$ implies $C[M] \longrightarrow C[M']$ for every context C. Notice that, unlike the evaluation relation, no branch of a nondeterministic choice is discarded.

We write $\longrightarrow^*$ for the reflexive and transitive closure of $\longrightarrow$. The standard technique shows that $\longrightarrow^*$ is confluent.

Basically the "normal form" of the reduction relation contains information on all evaluations. Since both the length of the reduction sequence as well as the size of the resulting "term" can be infinite, we need an auxiliary definition to formalise this idea.

The *finite canonical form* is defined by the following syntax:

$$S ::= \lambda x_1.\lambda x_2.\dots\lambda x_n.T \qquad T ::= \bot \mid T \oplus T' \mid x\, S_1\, \dots\, S_n$$

where the type of T must be o. It is basically an η-long term in β-normal form except for the additional constant $\bot$ for the diverging term of type o. We write FCF for the set of finite canonical forms. The set of finite canonical forms (of a given type) forms a poset with the least element $\lambda x_1.\dots\lambda x_n.\bot$ and the order $\sqsubseteq$ defined as the least partial order such that $C[\bot] \sqsubseteq C[T]$ for every context C and term T with $C[T] \in$ FCF.

One can compare a $\lambda_{\mathrm{nd}}\mathbf{Y}$-term with a finite canonical form in a similar way because the constructors of finite canonical forms (except for $\bot$) can be regarded as those of the calculus. We write $T \lhd M$ for the relation consisting of pairs $C[\bot,\dots,\bot] \lhd C[M_1,\dots,M_n]$ for every multi-hole context C and terms $M_1,\dots,M_n$ such that $C[\bot,\dots,\bot] \in$ FCF and $C[M_1,\dots,M_n]$ is a $\lambda_{\mathrm{nd}}\mathbf{Y}$-term.

A *Böhm tree* is an order ideal[4] of finite canonical forms with respect to $\sqsubseteq$. Given a $\lambda_{\mathrm{nd}}\mathbf{Y}$-term M in η-long form, its *Böhm tree* $BT(M)$ is defined by

$$BT(M) := \{T \in \mathrm{FCF} \mid \exists M'.\, M \longrightarrow^* M' \rhd T\}.$$

There is, of course, a tight relationship between the evaluation and reduction. Let $\bar{\cdot} : \{L,R,0\}^* \to \{L,R\}^*$ be the map that erases 0. For $\bar{\pi} \in \{L,R\}^*$, we define $C_{\bar{\pi}}$ by

$$C_\epsilon := [] \qquad C_{L\bar{\pi}} := C_{\bar{\pi}} \oplus \bot \quad \text{and} \quad C_{R\bar{\pi}} := \bot \oplus C_{\bar{\pi}}.$$

Proposition 1. *For every $x_1 : \mathsf{o},\dots,x_n : \mathsf{o} \vdash M : \mathsf{o}$ in η-long form, the map $\bar{\cdot}$ is a bijection from $\{\pi \mid M \overset{\pi}{\hookrightarrow} x_i\}$ to $\{\bar{\pi} \mid C_{\bar{\pi}}[x_i] \in BT(M)\}$ for every $i \in [n]$.*

Remark 2. Proposition 1 together with certain properties on $BT(\cdot)$ shows that the Böhm tree interpretation can be a solution to the problem in Section II-C. This is, however, unsatisfactory for us. For example, this solution is quite syntactic rather than semantic and $\oplus$ in this solution is neither commutative nor associative. Recall that to solve the problem in Section II-C is not our ultimate goal, rather we aim to use the problem as a guide of our development that provides us with a syntactic description of the generalised species of structures [18, 19].

III. Groupoid of Rigid Refinement Types

This section introduces types with non-idempotent and non-commutative intersection which we call *rigid refinement intersection types* (cf. [32]). The key difference from the existing work on non-commutative intersection types (such as [28, 30, 32]) is that we consider the types $a \wedge b$ and $b \wedge a$ distinct but isomorphic. In other words, while we do not allow implicit use of commutation, we allow explicit commutation.

Remark 3. The indexed syntax of rigid intersection types is reminiscent of the *indexed linear logic* of Bucciarelli and Ehrhard [7]. The development in Sections III and IV can be seen as a proof-relevant variant of [7], although there is a difference even in the proof-irrelevant setting. For example, indices of rigid intersection types do not affect provability of a judgement. A detailed comparison is left for future work.

A. Rigid refinement intersection types

The *rigid intersection types* are basically the same as the standard nonidempotent intersection types except that an intersection is no longer a finite multiset of types but a finite sequence. We write $\langle a_1,\dots,a_n\rangle$ for a sequence. Then the syntax is given by:

$$a, b ::= \star \mid \theta \multimap b \qquad \theta ::= \langle a_1,\dots,a_n\rangle$$

where $n \geq 0$. The sequence $\langle a_1,\dots,a_n\rangle$ is also written as $\langle a_i\rangle_{i\in[n]}$, or simply as $\langle \vec{a}\rangle$ if the length is unimportant. The sequence $\langle a_1,\dots,a_n\rangle$ means the intersection type $a_1 \wedge \cdots \wedge a_n$.

The *refinement relation* $a \lhd A$ is defined by:

$$\frac{}{\star \lhd \mathsf{o}} \qquad \frac{\theta \lhd\, !A \quad b \lhd B}{\theta \multimap b \lhd A \to B} \qquad \frac{\forall i \in [n].\ a_i \lhd A}{\langle a_i\rangle_{i\in[n]} \lhd\, !A}$$

Here $A \to B$ is identified with $!A \multimap B$, where $!$ and $\multimap$ are the exponential modality and linear implication of linear logic. We write $[\![A]\!] := \{a \mid a \lhd A\}$ for the set of all refinement types of the simple type A.

B. Type isomorphisms

Although the type $\langle a, b\rangle$ is not identical to $\langle b, a\rangle$, they are clearly closely related. Relating these types is also motivated by an observation that a non-commutative intersection type system does not enjoy subject reduction [27, Remark 2.9].[5] We consider these types isomorphic.

Let a_1 and a_2 be rigid intersection types (not necessarily refinements of simple types). We write $\varphi : a_1 \cong a_2$ to mean that a_1 and a_2 are isomorphic, of which φ is a *witness* (or a *proof*). This relation is defined by the rules below:

$$\frac{}{\mathrm{id}_\star : \star \cong \star} \qquad \frac{\psi :\ \theta' \cong \theta \qquad \varphi :\ b \cong b'}{(\psi \multimap \varphi) :\ (\theta \multimap b) \cong (\theta' \multimap b')}$$

$$\frac{\sigma \in \mathfrak{S}_n \qquad \forall i \in [n].\ \ \varphi_i :\ a_i \cong a'_{\sigma(i)}}{(\sigma, \langle \varphi_i\rangle_{i\in[n]}) :\ \langle a_i\rangle_{i\in[n]} \cong \langle a'_i\rangle_{i\in[n]}}$$

Here $\mathrm{id}_\star$ is a constant witnessing that the type $\star$ is (obviously) isomorphic to $\star$. The rules are fairly straightforward; we note contravariance of the position of θ in $\theta \multimap b$.

As expected, type isomorphism is an equivalence relation. Reflexivitiy is witnessed by id_a and id_θ defined by $\mathrm{id}_{\theta\multimap b} := \mathrm{id}_\theta \multimap \mathrm{id}_b$ and $\mathrm{id}_{\langle a_i\rangle_{i\in[n]}} := (\mathrm{id}_{[n]}, \langle \mathrm{id}_{a_i}\rangle_{i\in[n]})$. Transitivity is witnessed by the composition operation $\circ_A$ such that φ_1 :

[4]Given a poset $(P, \leq)$, an *order ideal* is a subset $X \subseteq P$ that is downward closed (i.e. $x \leq y$ and $y \in X$ implies $x \in X$) and directed (i.e., for every $x, y \in X$, there exists $z \in X$ such that $x \leq z$ and $y \leq z$).

[5]Although the remark in [27] is about system $\mathbb{I}$, in which $\wedge$ is non-commutative and non-associative, it directly applies to an intersection type system with non-commutative but associative $\wedge$.

$a_1 \cong a_2$ and $\varphi_2 : a_2 \cong a_3$ implies $(\varphi_2 \circ \varphi_1) : a_1 \cong a_3$, which can be defined by $\mathrm{id}_\star \circ_\mathrm{o} \mathrm{id}_\star := \mathrm{id}_\star$ and

$$(\psi_2 \multimap \varphi_2) \circ (\psi_1 \multimap \varphi_1) := (\psi_1 \circ \psi_2) \multimap (\varphi_2 \circ \varphi_1)$$
$$(\sigma_2, \langle \varphi^2_j \rangle_j) \circ (\sigma_1, \langle \varphi^1_i \rangle_i) := (\sigma_2\sigma_1, \langle \varphi^2_{\sigma_1(i)} \circ \varphi^1_i \rangle_i)$$

where we omit the ranges of i and j which must be the same. Symmetry is proved by the inverse operation $(\cdot)^{-1}$ defining by $\mathrm{id}_\star^{-1} := \mathrm{id}_\star$, $(\psi \multimap \varphi)^{-1} := \psi^{-1} \multimap \varphi^{-1}$ and $(\sigma, \langle \varphi_i \rangle_{i \in [n]})^{-1} := (\sigma^{-1}, \langle \varphi^{-1}_{\sigma^{-1}(j)} \rangle_{j \in [n]})$. Then $\varphi : a_1 \cong a_2$ implies $\varphi^{-1} : a_2 \cong a_1$ and $\varphi^{-1} \circ \varphi = \varphi \circ \varphi^{-1} = \mathrm{id}$.

By these data, the refinement types of a simple type A form a groupoid (i.e. a category in which every morphism is invertible), whose objects are rigid refinement types $a \lhd A$ and morphisms from a to b are type isomorphisms $\varphi : a \cong b$. We write this groupoid as $[\![A]\!]$. The groupoid $[\![!A]\!]$ whose object is $\theta \lhd !A$ is defined in the same way.

We abbreviate $(\sigma, \langle \mathrm{id}_{a_i} \rangle_i)$ to σ and $(\mathrm{id}, \langle \varphi_i \rangle_i)$ to $\langle \varphi_i \rangle_i$.

C. *Categorical definition*

The groupoid $[\![A]\!]$ of rigid refinement intersection types and isomorphisms has a simple categorical definition. This is, indeed, the interpretation of simple types by generalised species of structures [18, 19].

We start from an auxiliary definition. Let $\mathcal{C}$ be a small category. We define a category $\mathbb{P}\mathcal{C}$ as follows. An object of $\mathbb{P}\mathcal{C}$ is a (possibly empty) finite sequence $\langle C_i \rangle_{i \in [n]}$ of objects in $\mathcal{C}$. A morphism from $\langle C_i \rangle_{i \in [n]}$ to $\langle D_i \rangle_{i \in [m]}$ is a tuple $(\sigma, \langle f_i \rangle_{i \in [n]})$ where $\sigma : [n] \to [m]$ is a bijection and $f_i : C_i \to D_{\sigma(i)}$ is a morphism in $\mathcal{C}$ (for all $i \in [n]$). Note that there is no morphism if $n \neq m$. Composition is given by

$$(\sigma_2, \langle g_j \rangle_{j \in [n]}) \circ (\sigma_1, \langle f_i \rangle_{i \in [n]}) := (\sigma_2 \circ \sigma_1, \langle g_{\sigma_1(i)} \circ f_i \rangle_{i \in [n]}).$$

The groupoids $[\![A]\!]$ are given by induction on the structure of simple types A. For the base case, $[\![\mathrm{o}]\!]$ is the set $\{\star\}$, i.e. a category with a single object $\star$ and no morphism other than the identity. For the inductive steps, we have

$$[\![A \to B]\!] = [\![!A]\!]^{\mathrm{op}} \times [\![B]\!] \qquad [\![!A]\!] = \mathbb{P}\,[\![A]\!].$$

This resembles the interpretation of types in the relational model: we just replace a set with a groupoid and the finite multiset comonad with $\mathbb{P}$.

D. *Isomorphisms of type environments*

A type environment is a sequence of type bindings of the form $x : a$ (such that a is a refinement of the simple type of x). Here a variable can appear many times in a sequence. The refinement relation and the isomorphisms can be naturally extended to intersection type environments:

$$\frac{\forall (x : a) \in \Theta.\ \exists (x : A) \in \Gamma.\ a \lhd A}{\Theta \lhd \Gamma}$$

$$\frac{\sigma \in \mathfrak{S}_n \qquad \forall i \in [n].\ (x_i = y_{\sigma(i)} \text{ and } \varphi_i :\ a_i \cong b_{\sigma(i)})}{(\sigma, \langle \varphi_i \rangle_{i \in [n]}) :\ (x_1 : a_1, \dots, x_n : a_n) \cong_\Gamma (y_1 : b_1, \dots, y_n : b_n)}.$$

IV. Rigid Resource Calculus

This section introduces the *rigid resource calculus*, by which one can describe individual runs of a term. It is a variant of the resource calculus in the sense of [16], a variant which uses lists instead of bags. The key notion is the equivalence relation $\sim$ on terms, which identifies terms describing the same run, where the groupoid structure of refinement types plays an important rôle.

A. *Rigid resource raw terms*

Rigid resource raw terms are defined by:

$$t, u := x \mid \lambda\vec{x}.t \mid t\,\mu \mid t \oplus \bullet \mid \bullet \oplus t \qquad \mu := \langle u_1, \dots, u_n \rangle$$

(formally each variable is annotated by a rigid intersection type, which is omitted above). Here $\bullet$ is a place holder for the unused part of branching and $\vec{x}$ is a (possibly empty) sequence of variables. Rigid resource terms appearing in the sequel are always *linear*, i.e. each variable appears exactly once.

A rigid resource term shall be used to describe a reduction sequence of a $\lambda_{\mathrm{nd}}\mathbf{Y}$-term.[6] For example, $t \oplus \bullet$ means that the left-branch should be chosen here; since the right branch is irrelevant in this case, a rigid resource raw term simply ignores it.

A *rigid type environment* Ξ is a finite sequence of type bindings of the form $x : a$. We stipulate that a rigid type environment contains each variable at most once. We write Ξ_1, Ξ_2 for the concatenation. For sequences $\vec{x} = x_1, \dots, x_n$ and $\vec{a} = a_1, \dots, a_n$, we write $\vec{x} : \vec{a}$ for $x_1 : a_1, \dots, x_n : a_n$. A *type judgement* is of the form $\Xi \vdash t : a$ or $\Xi \vdash \mu : \langle \vec{a} \rangle$. The typing rules are listed in Fig. 1. It is worth noting that the order of type bindings in a type environment does not affect the derivability of a judgement by the exchange rule. This rule can be seen as a weaker version of type-isomorphism rules; the relationship to general rules is the topic of Section IV-D.

In the sequel, we shall consider only typed rigid resource terms, i.e. rigid resource terms associated with derivable typing judgements $\Xi \vdash t : a$ (or $\Xi \vdash \mu : \theta$). The notion of *η-long forms* is defined in a similar way to the $\lambda_{\mathrm{nd}}\mathbf{Y}$-calculus . The associated judgements are often left implicit.

We can give an evaluation relation for rigid resource raw terms. *Evaluation contexts* are given by: $E ::= [] \mid E\,\mu$.

$$E[t \oplus \bullet] \overset{L}{\hookrightarrow} E[t] \qquad E[\bullet \oplus t] \overset{R}{\hookrightarrow} E[t]$$
$$E[(\lambda\vec{x}.s)\,\langle u_1, \dots, u_n \rangle] \overset{0}{\hookrightarrow} E[\{u_1/x_1, \dots, u_n/x_n\}s].$$

As before, we write $s \overset{\pi}{\hookrightarrow} s'$ if there exists a sequence $s = s_0 \overset{d_1}{\hookrightarrow} s_1 \overset{d_2}{\hookrightarrow} \cdots \overset{d_n}{\hookrightarrow} s_n = s'$ and $\pi = d_1 d_2 \dots d_n$ (where $n \geq 0$). Evaluation of a rigid resource raw term is deterministic.

We give a (full) reduction semantics for rigid resource terms. It is defined by the base cases

$$(\lambda x_1 \dots x_n.t)\,\langle u_1, \dots, u_n \rangle \longrightarrow \{u_1/x_1, \dots, u_n/x_n\}t$$
$$(t \oplus \bullet)\,\mu \longrightarrow (t\,\mu) \oplus \bullet \qquad (\bullet \oplus t)\,\mu \longrightarrow \bullet \oplus (t\,\mu).$$

[6] Here is another point of view: each term constructor corresponds to a typing rule of an intersection type system. See also Remarks 8 and 13.

$$\frac{}{x : a \vdash x : a} \qquad \frac{\Xi, x_1 : a_1, \dots, x_n : a_n \vdash t : b}{\Xi \vdash \lambda x_1 \dots x_n.t : \langle a_1, \dots, a_n \rangle \multimap b} \qquad \frac{\Xi_1 \vdash t : \theta \multimap b \qquad \Xi_2 \vdash \mu : \theta}{\Xi_1, \Xi_2 \vdash t\,\mu : b}$$

$$\frac{\Xi \vdash t : a}{\Xi \vdash t \oplus \bullet : a} \qquad \frac{\Xi \vdash t : a}{\Xi \vdash \bullet \oplus t : a} \qquad \frac{\forall i \in [n].\ \Xi_i \vdash u_i : a_i}{\Xi_1, \dots, \Xi_n \vdash \langle u_i \rangle_{i \in [n]} : \langle a_i \rangle_{i \in [n]}} \qquad \frac{x_1 : a_1, \dots, x_n : a_n \vdash t : b \qquad \sigma \in \mathfrak{S}_n}{x_{\sigma(1)} : a_{\sigma(1)}, \dots, x_{\sigma(n)} : a_{\sigma(n)} \vdash t : b}$$

Fig. 1. The typing rules for rigid resource terms (Environment permutation for lists μ is omitted.)

The reduction relation is strongly normalising because of the linearity of rigid resource raw terms. We write $\mathsf{nf}(t)$ for the normal form of t. Furthermore it is confluent since there is no critical pair. Unlike the standard resource calculus, the reduction relation is deterministic (or, reduction does not introduce formal sums).

B. Rigid term approximation of $\lambda_{\mathrm{nd}}\mathbf{Y}$-terms

A *variable refinement* X is a sequence of the form $x_1 \lhd y_1, \dots, x_n \lhd y_n$ such that $x_i \neq x_j$ if $i \neq j$. The variables $x_1, \dots, x_n$ are those of rigid resource terms and $y_1, \dots, y_n$ are those of the $\lambda_{\mathrm{nd}}\mathbf{Y}$-calculus. For sequences $\vec{x}$ and $\vec{y}$ of variables of the same length, $\vec{x} \lhd \vec{y}$ denotes the variable refinement $x_1 \lhd y_1, \dots, x_n \lhd y_n$. Let $\mathrm{rng}(X) := \{y \mid (x \lhd y) \in X\}$.

An *approximation judgement* has the form $X \vdash t \lhd M$, and valid judgements are defined by the rules in Fig. 2.

C. Representation of reduction sequences

A rigid raw term $t \lhd M$ gives an evaluation of M. The only nontrivial case is $M = E[N_1 \oplus N_2]$ as other cases are deterministic. Then $t \lhd M$ implies $t = E'[u \oplus \bullet]$ or $E'[\bullet \oplus u]$; we choose the left-branch for the former case and the right-branch for the latter.

Lemma 4. *Assume $X \vdash t \lhd M$. If $t \overset{\pi}{\hookrightarrow} t'$, then $M \overset{\pi}{\hookrightarrow} M'$ and $X \vdash t' \lhd M'$ for some M'.*

Corollary 5. *Let $x_1 : \mathsf{o}, \dots, x_n : \mathsf{o} \vdash M : \mathsf{o}$. If $y \lhd x_i \vdash t \lhd M$ and $y : \star \vdash t : \star$, then t represents a unique evaluation sequence of $M \overset{\pi}{\hookrightarrow} x_i$, i.e., $t \overset{\pi}{\hookrightarrow} y$.*

Furthermore every evaluation of a $\lambda_{\mathrm{nd}}\mathbf{Y}$-term has a representation by a rigid resource term.

Lemma 6. *Assume $X \vdash t' \lhd M'$. If $M \overset{\pi}{\hookrightarrow} M'$, then $t \overset{\pi}{\hookrightarrow} t'$ and $X \vdash t \lhd M$ for some t.*

Corollary 7. *Let $x_1 : \mathsf{o}, \dots, x_n : \mathsf{o} \vdash M : \mathsf{o}$ and $M \overset{\pi}{\hookrightarrow} x_i$. There exists t such that $y \lhd x_i \vdash t \lhd M$, $x_i : \star \vdash t : \star$ and $t \overset{\pi}{\hookrightarrow} y$.*

However there is a problem as mentioned around (1) in Section I-B: some evaluation $M \overset{\pi}{\hookrightarrow} x_i$ has more than one rigid resource raw term representation. For example, consider

$$\begin{aligned} M &= (\lambda f.f\,(f\,z))\,((\lambda x.x) \oplus (\lambda y.y)) \\ t_1 &= (\lambda f_1 f_2.f_1\,\langle f_2\,\langle z \rangle \rangle)\,\langle ((\lambda x.x) \oplus \bullet), (\bullet \oplus (\lambda y.y)) \rangle \\ t_2 &= (\lambda f_2 f_1.f_1\,\langle f_2\,\langle z \rangle \rangle)\,\langle (\bullet \oplus (\lambda y.y)), ((\lambda x.x) \oplus \bullet) \rangle. \end{aligned}$$

Then both t_1 and t_2 represent $M \xhookrightarrow{0L0R0} z$.

Remark 8. Corollaries 5 and 7 suggest existence of a sound and complete intersection type system defined as follows. A judgement of the type system is of the form $x_1 : a_1, \dots, x_n : a_n \vdash M : b$ (where x_i and x_j can be the same even when $i \neq j$). This judgement is derivable if there exists t such that $y_1 : a_1, \dots, y_n : a_n \vdash t : b$ and $y_1 \lhd x_1, \dots, y_n \lhd x_n \vdash t \lhd M$. This "type system" is sound and complete by Corollaries 5 and 7. Indeed it is the type system corresponding to the relational model of linear logic (see, e.g., [10]). From this point of view, a rigid resource term is a representation of a derivation of an intersection type system.

D. Action of type isomorphisms

The goal of this subsection is to avoid the redundancy described at the end of the previous subsection. The idea is to introduce an equivalence relation $\sim$ relating terms that represent the same evaluation.

Basically redundancy arises from application of the same permutation to both the variable list and the argument list. For example, the term

$$t \ := \ (\lambda x_1 \dots x_n.t)\,\langle u_1, \dots, u_n \rangle$$

represents the same evaluation as

$$t' \ := \ (\lambda x_{\sigma(1)} \dots x_{\sigma(n)}.t)\,\langle u_{\sigma(1)}, \dots, u_{\sigma(n)} \rangle.$$

A difficulty comes from the fact that the corresponding variable and argument lists are not necessarily syntactically adjacent. Consider, for example,

$$(\lambda f.f\,\langle z_1, z_2 \rangle)\,\langle \lambda y_1 y_2.t \rangle \qquad \text{and} \qquad (\lambda f.f\,\langle z_2, z_1 \rangle)\,\langle \lambda y_2 y_1.t \rangle$$

that are reduced to, respectively,

$$(\lambda y_1 y_2.t)\,\langle z_1, z_2 \rangle \qquad \text{and} \qquad (\lambda y_2 y_1.t)\,\langle z_2, z_1 \rangle.$$

To correctly handle such a complex case, we use type isomorphisms and their action on rigid resource terms.

We do not have the type isomorphism rule in the above type system. For example, $\Theta \vdash \langle t_1, \dots, t_n \rangle : \langle a_1, \dots, a_n \rangle$ does not imply $\Theta \vdash \langle t_1, \dots, t_n \rangle : \langle a_{\sigma(1)}, \dots, a_{\sigma(n)} \rangle$, although $\sigma : \langle a_1, \dots, a_n \rangle \cong \langle a_{\sigma(1)}, \dots, a_{\sigma(n)} \rangle$.

However, applying the same permutation σ to the list of terms gives a valid judgement. For example, in the above case, we have $\Theta \vdash \langle t_{\sigma(1)}, \dots, t_{\sigma(n)} \rangle : \langle a_{\sigma(1)}, \dots, a_{\sigma(n)} \rangle$.

By generalising this idea, it is natural to expect that, given a rigid resource term $\Xi \vdash t : a$ and a type isomorphism $\varphi : a' \cong a$, we obtain another rigid resource term t' by "applying" φ and we have $\Xi \vdash t' : a'$. Indeed it is the case provided that the term is η-long. We write such t' as $t[\varphi]$ and call the operation $(-)[\varphi]$ the *(right) action* of φ.

$$\frac{}{x \lhd y \vdash x \lhd y} \qquad \frac{X, x_1 \lhd y, \dots, x_n \lhd y \vdash t \lhd M \qquad y \notin \mathrm{rng}(X)}{X \vdash (\lambda x_1 \dots x_n.t) \lhd (\lambda y.M)} \qquad \frac{X_1 \vdash t \lhd M \qquad X_2 \vdash \mu \lhd N}{X_1, X_2 \vdash (t\,\mu) \lhd M\,N}$$

$$\frac{X_1 \vdash t \lhd M \qquad X_2 \vdash \mu \lhd (\mathbf{Y}\,M)}{X_1, X_2 \vdash (t\,\mu) \lhd (\mathbf{Y}\,M)} \qquad \frac{X \vdash t \lhd M_1}{X \vdash (t \oplus \bullet) \lhd (M_1 \oplus M_2)} \qquad \frac{X \vdash t \lhd M_2}{X \vdash (\bullet \oplus t) \lhd (M_1 \oplus M_2)}$$

$$\frac{\forall i \in [n].\ X_i \vdash t_i \lhd M}{X_1, \dots, X_n \vdash \langle t_i \rangle_{i \in [n]} \lhd M} \qquad \frac{x_1 \lhd y_1, \dots, x_n \lhd y_n \vdash t \lhd M \qquad \sigma \in \mathfrak{S}_n}{x_{\sigma(1)} \lhd y_{\sigma(1)}, \dots, x_{\sigma(n)} \lhd y_{\sigma(n)} \vdash t \lhd M}$$

Fig. 2. Rigid approximation relation (Permutation of variable refinements for lists is omitted.)

The action $(-)[\varphi]$ of an isomorphism to terms is defined by the rules in Fig. 3. It is defined by mutual induction with $\{[\varphi]/x\}(-)$, which describes the action of the isomorphism φ to environments. We have the following derived rules:

$$\frac{\Xi \vdash t : a \qquad \varphi : a' \cong a}{\Xi \vdash t[\varphi] : a'} \qquad \frac{\Xi, x : b \vdash t : a \qquad \varphi : b \cong b'}{\Xi, x : b' \vdash \{[\varphi]/x\}t : a}.$$

Example 9. Let $\sigma = (1\ 2) \in \mathfrak{S}_2$ and $a := \langle \star, \star \rangle \multimap \star$. Then $(\sigma \multimap \mathrm{id}) : a \cong a$. Let φ be this isomorphism and $\Theta \vdash \lambda y_2 y_1.u_0 : a$ be a rigid resource raw term in η-long form. Let $u := \lambda y_2 y_1.u_0$. By applying φ to u, we obtain $u[\varphi] = \lambda y_1 y_2.u_0$. Consider another isomorphism $(\langle\varphi\rangle \multimap \mathrm{id}) : (\langle a \rangle \multimap \star) \cong (\langle a \rangle \multimap \star)$. Consider $z_1 : \star, z_2 : \star \vdash \lambda f.f\,\langle z_1, z_2 \rangle : \langle a \rangle \multimap \star$, which we write as t. By applying $(\langle\varphi\rangle \multimap \mathrm{id})$ to t, we obtain $t[\langle\varphi\rangle \multimap \mathrm{id}] = \lambda f.f\,\langle z_2, z_1 \rangle$.

Lemma 10. *For every term t in η-long form, we have*

$$\begin{gathered}\{[\varphi_i]/x_i\}_i(t[\psi]) = (\{[\varphi_i]/x_i\}_i t)[\psi] \\ (t[\psi_1])[\psi_2] = t[\psi_1 \circ \psi_2] \\ \{[\varphi_i^1]/x_i\}_i(\{[\varphi_i^2]/x_i\}_i t) = \{[\varphi_i^1 \circ \varphi_i^2]/x_i\}_i t\end{gathered}$$

provided the type isomorphisms are of the appropriate types.

Remark 11. A type isomorphism $\varphi : a \cong_A a'$ induces a rigid resource term, which we write as $\vdash [\varphi] : a' \to a$ by abuse of notation, such that $\vdash [\varphi] \lhd \eta_A(\lambda y.y)$ where $\eta_A(\lambda y.y)$ is the η-long form of $\lambda y.y$. The action of a type isomorphism φ to $\Theta \vdash t : a'$ can be seen as the application of the term $[\varphi]$ to t, followed by the reduction of redex related to $[\varphi]$.

Now we define $\sim$ as the least congruence that satisfies

$$\begin{aligned}&t_0\,(\mu_1[\varphi_1])\,\dots\,(\mu_n[\varphi_n]) \\ &\quad \sim (t_0[\varphi_1 \multimap \dots \multimap \varphi_n \multimap \mathrm{id}])\,\mu_1\,\dots\,\mu_n\end{aligned}$$

for every η-long rigid resource terms $t_0, \mu_1, \dots, \mu_n$ and type isomorphisms $\varphi_1, \dots, \varphi_n$ of appropriate types. We write $\widetilde{t}$ for the equivalence class of $\sim$ to which t belongs.

Example 12. Let φ and a be those defined in Example 9. We have $(\lambda f.f\,\langle z_1, z_2 \rangle)\,\langle \lambda y_1 y_2.t \rangle \sim (\lambda f.f\,\langle z_2, z_1 \rangle)\,\langle \lambda y_2 y_1.t \rangle$ from

$$\begin{aligned}(\lambda f.f\,\langle z_1, z_2 \rangle)\,\langle \lambda y_1 y_2.t \rangle &= (\lambda f.f\,\langle z_1, z_2 \rangle)\,(\langle \lambda y_2 y_1.t \rangle[\varphi]) \\ (\lambda f.f\,\langle z_2, z_1 \rangle)\,\langle \lambda y_2 y_1.t \rangle &= ((\lambda f.f\,\langle z_1, z_2 \rangle)[\varphi \multimap \mathrm{id}])\,\langle \lambda y_2 y_1.t \rangle.\end{aligned}$$

Remark 13. Recall that a rigid resource raw-term can be seen as a derivation of an intersection type system (Remark 8). From this point of view, a rigid resource term (i.e. an equivalence class modulo $\sim$) is an equivalence class of derivations modulo natural commutations such as

$$\frac{\dfrac{\Delta, x : \vec{a} \vdash M : b}{\Delta \vdash \lambda x.M : \langle \vec{a} \rangle \multimap b} \qquad (\mathrm{id} \multimap \varphi) : (\langle \vec{a} \rangle \multimap b') \cong (\langle \vec{a} \rangle \multimap b)}{\Delta \vdash \lambda x.M : \langle \vec{a} \rangle \multimap b'}$$

$$\longleftrightarrow \qquad \frac{\dfrac{\Delta, x : \vec{a} \vdash M : b \qquad \varphi : b' \cong b}{\Delta, x : \vec{a} \vdash M : b'}}{\Delta \vdash \lambda x.M : \langle \vec{a} \rangle \multimap b'}.$$

The notions and operations defined for rigid resource terms are well-defined for the equivalence classes $\widetilde{t}$. For example,

- $\Xi \vdash t : a$ and $t \sim t'$ implies $\Xi \vdash t' : a$, and
- $t_1 \sim t_2$ and $t_1 \overset{\pi}{\hookrightarrow} t_1'$ implies $t_2 \overset{\pi}{\hookrightarrow} t_2'$ with $t_1' \sim t_2'$.

We have the following result as expected.

Theorem 14. *Let $x_1 : \mathsf{o}, \dots, x_n : \mathsf{o} \vdash M : \mathsf{o}$ and assume $M \overset{\pi}{\hookrightarrow} x_i$ for some i. Then there exists t unique up to $\sim$ such that $y \lhd x_i \vdash t \lhd M$ and $t \overset{\pi}{\hookrightarrow} y$.*

E. Rigid Taylor expansion

Let us fix an infinite sequence $z_1, z_2, \dots, z_n, \dots$ of distinct variables. We write $\vec{z}$ for a prefix of this sequence.

Definition 15. Let $\Gamma \vdash M : A$ be a $\lambda_{\mathrm{nd}}\mathbf{Y}$-term in η-long form. Define, for $\Delta = (\vec{z} : \vec{b}) \lhd \Gamma$ and $a \lhd A$,

$$[\![M]\!](\Delta, a) := \{\widetilde{t} \mid \vec{z} \lhd \vec{x} \vdash t \lhd M \text{ and } \vec{z} : \vec{b} \vdash t : a\}$$

We call $[\![M]\!]$ the *rigid Taylor expansion of* M. We write $(\Delta \vdash \widetilde{t} : a) \in [\![M]\!]$ to mean $\widetilde{t} \in [\![M]\!](\Delta, a)$.

Theorem 14 shows that the rigid Taylor expansion $[\![M]\!]$ can be seen as an enumeration of evaluation sequences, which can be defined by induction on M and is thus compositional. This aspect is further studied in the next section.

Recall the groupoid $[\![A]\!]$ of refinement types and isomorphims, defined in Section III-C. By the action of an isomorphism on a rigid resource term, the rigid Taylor expansion becomes a functor.

Theorem 16. *Let $\Gamma \vdash M : A$ be a $\lambda_{\mathrm{nd}}\mathbf{Y}$-term in η-long form. Then $[\![M]\!]$ is a functor $\mathbb{P}[\![\Gamma]\!] \times [\![A]\!]^{\mathrm{op}} \to \mathbf{Set}$.*

The rigid Taylor expansion is sound with respect to the Böhm theory. Let $\Gamma \vdash \mathcal{T} : A$ be a Böhm tree (i.e. an order

$$(\lambda x_1 \dots x_n.t)[\mathrm{id} \multimap \varphi] := \lambda x_1 \dots x_n.(t[\varphi]) \qquad \langle t_i\rangle_{i\in[n]}[\sigma] := \langle t_{\sigma(j)}\rangle_{j\in[n]}$$
$$(\lambda x_1 \dots x_n.t)[\sigma \multimap \mathrm{id}] := \lambda x_{\sigma^{-1}(1)} \dots x_{\sigma^{-1}(n)}.t \qquad \langle t_i\rangle_{i\in[n]}[\langle\varphi_i\rangle_{i\in[n]}] := \langle t_i[\varphi_i]\rangle_{i\in[n]}$$
$$(\lambda x_1 \dots x_n.t)[\langle\varphi_i\rangle_{i\in[n]} \multimap \mathrm{id}] := \lambda x_1 \dots x_n.(\{[\varphi_i]/x_i\}_{i\in[n]}t) \qquad \{[\psi]/x\}(x\,\mu_1\,\dots\,\mu_n) := x\,(\mu_1[\varphi_1])\,\dots\,(\mu_n[\varphi_n])$$
$$(t \oplus \bullet)[\varphi] := (t[\varphi]) \oplus \bullet \qquad (\bullet \oplus t)[\varphi] := \bullet \oplus (t[\varphi]) \qquad (\text{where } \psi = \varphi_1 \multimap \dots \multimap \varphi_n \multimap \mathrm{id}_\star)$$

Fig. 3. Action of isomorphisms on rigid raw terms. Other isomorphims are decomposed into the above ones. The definition of $\{[\varphi]/x\}(-)$ is the same as the standard substitution except for the above case.

ideal of the finite canonical forms $\Gamma \vdash T : A$). We define $[\![\mathcal{T}]\!]$ by, given $\Delta = (\vec{z} : \vec{b}) \lhd \Gamma$ and $a \lhd A$,

$$[\![\mathcal{T}]\!](\Delta, a) := \{\widetilde{t} \mid \vec{z} : \vec{b} \vdash t : a \text{ and } \exists T \in \mathcal{T}.\ \vec{z} \lhd \vec{x} \vdash t \lhd T\}.$$

Theorem 17. *The family of functions*

$$[\![M]\!](\Theta, a) \to [\![\mathrm{BT}(M)]\!](\Theta, a) \ : \ \widetilde{t} \mapsto \mathsf{nf}(\widetilde{t})$$

defines a natural isomorphism $[\![M]\!] \cong [\![\mathrm{BT}(M)]\!]$. Thus the rigid Taylor expansion $[\![-]\!]$ is sound with respect to the Böhm theory.

V. Generalised Species of Structures

This section shows that the rigid Taylor expansion of a $\lambda_{\mathrm{nd}}\mathbf{Y}$-term is equivalent to its interpretation in the bicategory **ESP** (for *espèces de structures* [25, 26]) of generalised species of structures by Fiore et al. [19], which is a proof-relevant extension of the relational model $\mathbf{Rel}_!$. We first review the generalised species of structures and its cartesian closed structure and then compare it with the rigid Taylor expansion.

A. Profunctor

We introduce the bicategory **Prof** of *profunctors* (or *distributors*; see e.g. [3, 4] and [6, § 7.7]), which can be regarded as a proof-relevant extension of the category **Rel** of sets and relations.

Given small categories $\mathcal{A}$ and $\mathcal{B}$, a *profunctor* from $\mathcal{A}$ to $\mathcal{B}$ is a functor $f : \mathcal{A} \times \mathcal{B}^{\mathrm{op}} \to \mathbf{Set}$. This is an extension of relations: a relation $R \subseteq Y \times Z$ can be seen as a profunctor f_R between sets (i.e. categories with no morphism except for the identities) such that $f_R(y, z)$ is either singleton or empty depending on whether $(y, z) \in R$. We write $f : \mathcal{A} \nrightarrow \mathcal{B}$ if f is a profunctor from $\mathcal{A}$ to $\mathcal{B}$.

Let $f : \mathcal{A} \nrightarrow \mathcal{B}$ and $g : \mathcal{B} \nrightarrow \mathcal{C}$ be profunctors. A morphism $\varphi : b \to b'$ in $\mathcal{B}$ acts on elements in $f(a, b')$ by $y \mapsto f(a, \varphi)(y) \in f(a, b)$ and in $g(b, c)$ by $z \mapsto g(\varphi, c)(z) \in g(b', c)$, which we write $y[\varphi]$ and $\{\varphi\}z$, respectively. The composite gf is defined by

$$(gf)(a, c) := \coprod_{b\in\mathcal{B}} (f(a, b) \times g(b, c))/\!\sim \tag{3}$$

where $\sim$ is the least equivalence relation that contains $(y, \{\varphi\}z) \sim (y[\varphi], z)$ for every morphism φ in $\mathcal{B}$.

B. Generalised species of structures

Given small categories $\mathcal{A}$ and $\mathcal{B}$, a $(\mathcal{A}, \mathcal{B})$-*species of structures* [18, 19] is defined as a functor $\mathbb{P}\mathcal{A} \times \mathcal{B}^{\mathrm{op}} \to \mathbf{Set}$, i.e. a profunctor $\mathbb{P}\mathcal{A} \nrightarrow \mathcal{B}$.

There is a bicategory whose 0-cell is a small category and whose 1-cell is an $(\mathcal{A}, \mathcal{B})$-species. The identity species $I_{\mathcal{A}} : \mathbb{P}\mathcal{A} \nrightarrow \mathcal{A}$ is defined by $I_{\mathcal{A}}(\theta, a) := \mathbb{P}\mathcal{A}(\langle a\rangle, \theta)$. The composition of $(\mathcal{A}, \mathcal{B})$-species of structures f and $(\mathcal{B}, \mathcal{C})$-species of structures g is defined as follows. Given $(\mathcal{A}, \mathcal{B})$-species of the structures $f : \mathbb{P}\mathcal{A} \nrightarrow \mathcal{B}$, its *lifting* $f^\sharp : \mathbb{P}\mathcal{A} \nrightarrow \mathbb{P}\mathcal{B}$ is defined by the following coend:

$$f^\sharp(\theta, \langle b_1, \dots, b_k\rangle) := \int^{(\theta_i)_{i\in k}\in(\mathbb{P}\mathcal{A})^k} \Big(\prod_{i\in[k]} f(\theta_i, b_i)\Big) \times \mathbb{P}\mathcal{A}(\theta_1 \cdots \theta_k, \theta)$$

where $\theta_1 \cdots \theta_k$ is the concatenation of lists. Then $g \circ f$ is defined as the composition $gf^\sharp$ as profunctors. The 2-cells are natural transformations. We write this bicategory as **ESP**.

Remark 18. Fiore et al. [19] showed that **ESP** can be seen as the coKleisli bicategory of a pseudo-comonad $\underline{\mathbb{P}}$ on the bicategory **Prof** of profunctors; the action of $\underline{\mathbb{P}}$ to 0-cells is the same as $\mathbb{P}$. There is a formal similarity to the construction of the relational model $\mathbf{Rel}_!$, which is the coKleisli category of the finite multiset comonad !.

C. Cartesian closed structure

The bicategory **ESP** is cartesian closed [19].

Given small categories $\mathcal{A}_i$ ($i \in [n]$), the cartesian product $\prod_{i\in[n]} \mathcal{A}_i$ in **ESP** is the coproduct $\coprod_{i\in[n]} \mathcal{A}_i$ as categories. For an object $a \in \mathcal{A}_i$, the corresponding object of $\prod_{j\in[n]} \mathcal{A}_j$ is written as $\iota_i(a)$. For $g_i : \mathbb{P}\mathcal{B} \nrightarrow \mathcal{A}_i$ ($i \in [n]$), the tupling $\langle g_1, \dots, g_n\rangle : \mathbb{P}\mathcal{B} \nrightarrow \prod_{i\in[n]} \mathcal{A}_i$ is defined by

$$\langle g_1, \dots, g_n\rangle(b, \iota_i(a)) := g_i(b, a).$$

The projection $\pi_i : \mathbb{P}(\prod_{j\in[n]} \mathcal{A}_j) \nrightarrow \mathcal{A}_i$ is given by

$$\pi_i(u, a) := \begin{cases} \mathcal{A}_i(a, a') & \text{if } u = \langle\iota_i(a')\rangle \\ \emptyset & \text{otherwise.} \end{cases}$$

Given small categories $\mathcal{B}$ and $\mathcal{C}$, we define $\mathcal{C}^{\mathcal{B}} := \mathbb{P}\mathcal{B}^{\mathrm{op}} \times \mathcal{C}$. Given $g : \mathbb{P}(\mathcal{A} \sqcap \mathcal{B}) \nrightarrow \mathcal{C}$, we define $\Lambda_{\mathcal{B}}(g) : \mathbb{P}\mathcal{A} \nrightarrow \mathcal{C}^{\mathcal{B}}$ by

$$\Lambda_{\mathcal{B}}(g)(\langle a_1, \dots, a_n\rangle, (\langle b_1, \dots, b_m\rangle, c)) := g(\langle\iota_1(a_1), \dots, \iota_1(a_n), \iota_2(b_1), \dots, \iota_2(b_m)\rangle, c).$$

The evaluation map $\mathrm{eval}_{\mathcal{B},\mathcal{C}} : \mathbb{P}(\mathcal{C}^{\mathcal{B}} \sqcap \mathcal{B}) \nrightarrow \mathcal{C}$ is defined by

$$\mathrm{eval}_{\mathcal{B},\mathcal{C}}(u, c) := \begin{cases} (\mathbb{P}\mathcal{B}^{\mathrm{op}} \times \mathcal{C})((u\restriction_2, c), d') & \text{if } u\restriction_1 = \langle d' \rangle \\ \emptyset & \text{otherwise} \end{cases}$$

where, for $u \in \mathbb{P}(\mathcal{A}_1 \sqcap \mathcal{A}_2)$, $u\restriction_i \in \mathbb{P}\mathcal{A}_i$ $(i = 1, 2)$ is the subsequence consisting of objects from the ith component.

D. *Interpretation*

Types are interpreted as groupoids of rigid refinement types. So $[\![o]\!]$ is the category with one object $\star$ and one morphism, $[\![!A]\!] := \mathbb{P}[\![A]\!]$ and $[\![A \to B]\!] := [\![!A]\!]^{\mathrm{op}} \times [\![B]\!]$.

The interpretation of terms is the standard one for the λ-calculus fragment (i.e. x, $\lambda x.M$ and $M\,N$). The interpretation of the nondeterministic branching is the disjoint union

$$[\![M \oplus N]\!]_{\mathbf{ESP}}(\Theta, a) := [\![M]\!]_{\mathbf{ESP}}(\Theta, a) + [\![N]\!]_{\mathbf{ESP}}(\Theta, a).$$

We write $\mathbf{Y}^{\mathbf{ESP}}_A$ for the rigid Taylor expansion of (the η-long form of) $f : A \to A \vdash \mathbf{Y}\, f : A$.[7] Now the interpretation $[\![\mathbf{Y}\, M]\!]_{\mathbf{ESP}}$ of $\Gamma \vdash \mathbf{Y}\, M : A$ is defined as the composite

$$[\![\mathbf{Y}\, M]\!]_{\mathbf{ESP}} := \mathbf{Y}^{\mathbf{ESP}}_A \circ [\![M]\!]_{\mathbf{ESP}}.$$

We show that the ESP interpretation $[\![M]\!]_{\mathbf{ESP}}$ is isomorphic to the rigid Taylor expansion $[\![M]\!]$. As we have seen in Theorem 16, $[\![M]\!]$ is a generalised species of structures. The key lemma is that the composition as ESP can be seen as substitution followed by the rigid Taylor expansion. The equivalence relation $\sim$ in the composition of profunctors (Equation (3)) coincides with that of the resource calculus.

Lemma 19. *Let $x_1 : A_1, \dots, x_k : A_k \vdash M : B$ and $\Gamma \vdash N_i : A_i$ $(i \in [k])$. Then there is a natural isomorphism*

$$[\![M]\!] \circ \langle [\![N_1]\!], \dots, [\![N_k]\!] \rangle \;\cong\; [\![\{N_1/x_1, \dots, N_k/x_k\}M]\!].$$

Theorem 20. *For every term $\Gamma \vdash M : A$, the ESP interpretation is naturally isomorphic to the rigid Taylor expansion:*

$$[\![M]\!] \cong [\![M]\!]_{ESP} \;:\; \mathbb{P}[\![\Gamma]\!]^{\mathrm{op}} \times [\![A]\!] \to \mathbf{Set}.$$

Proof. (Sketch) By induction on the structure of M using Lemma 19. For example, $[\![\mathbf{Y}\, M]\!]_{\mathbf{ESP}} = \mathbf{Y}^{\mathbf{ESP}}_A \circ [\![M]\!]_{\mathbf{ESP}} \cong [\![\mathbf{Y}\, f]\!] \circ [\![M]\!] \cong [\![\{M/f\}\,(\mathbf{Y}\, f)]\!] = [\![\mathbf{Y}\, M]\!]$. A key observation is that $\mathrm{eval}_{[\![B]\!],[\![C]\!]}$ is equivalent to the rigid Taylor expansion of (the η-long form of) $f : B \to C, x : B \vdash f\, x : C$. □

VI. Reasoning about standard resource terms

This section studies the rigid Taylor expansion *of* standard resource terms—in which application arguments are bags (i.e. finite multisets)—and compares the rigid and standard Taylor expansions. As an application, we show the commutation between computing Böhm trees and Taylor expansions for the $\lambda_{\mathrm{nd}}\mathbf{Y}$-calculus (although some coefficients may be ∞). The proof is an extension of the combinatorial arguments in [16] showing the commutation property of the deterministic case. In this subsection, *resource terms* always mean standard resource terms, and rigid terms are always called rigid.

[7]There is another definition of $\mathbf{Y}^{\mathbf{ESP}}_A$ that does not relies on the rigid Taylor expansion. We omit it because of the space limitation.

A. *Preliminary: The resource calculus and Taylor expansions*

The syntax of *simple terms* and *simple bags* is given by:

$$v, w ::= x \mid \lambda x.v \mid v\,\xi \qquad \xi ::= [v_1, \dots, v_n]$$

where $n \geq 0$ and $[\dots]$ is a finite multiset. A *term* is a (possibly infinite) formal sum of simple terms with coefficients in $\mathbf{R}_+$, where $\mathbf{R}_+ := \{x \in \mathbf{R} \mid x \geq 0\}$. A simple term can appear (possibly infinitely) many times in a term. See, e.g., [16] for a more detailed definition.

Remark 21. Given a set D and a countable set I, we identify a formal sum $\sum_{i \in I} r_i d_i$ where $r_i \in \mathbf{R}_+$ and $d_i \in D$ with a mapping $(d \in D) \mapsto (\sum_{i \in I, d_i = d} r_i) \in \mathbf{R}_+ \cup \{\infty\}$. The equality of formal sums is that of the associated mappings.

We shall consider only *simply-typed* resource terms. The typing rules are the standard ones: for example, $[v_1, \dots, v_n]$ has type A if v_i has type A for every $i \in [n]$. The notion of η-long form is defined in the straightforward way.

The reduction is defined as follows. If t has n free occurrences of x, we have

$$(\lambda x.t)\,[u_1, \dots, u_n] \;\longrightarrow\; \sum_{\sigma \in \mathfrak{S}_n} \{u_{\sigma(1)}/x_1, \dots, u_{\sigma(n)}/x_n\}t$$

where $x_1, \dots, x_n$ are the names of the free occurrences of x in t. If the number of free occurrence of x in t is not n, then the above term is reduced to the empty sum.

Intermediate representations (or simply *representations*) of simple terms and bags are the same as simple terms and bags except that (i) simple terms have new constructors $v \oplus \bullet$ and $\bullet \oplus v$, and (ii) elements of each bag are linearly ordered (i.e. a bag ξ is represented by a list). A representation v induces a simple term $|v|$ obtained by forgetting the branching information (i.e. $|v \oplus \bullet| = |\bullet \oplus v| = |v|$) and the order of elements in a bag. Representations are used as an intermediate data structure during the Taylor expansion.

Figure 4 defines the *approximation relation* $v \blacktriangleleft M$, where v ranges over representations, and M over $\lambda_{\mathrm{nd}}\mathbf{Y}$-terms. We define the function $\mathsf{w}(-)$ by

$$\mathsf{w}(x) = 1 \qquad \mathsf{w}(\lambda x.v) = \mathsf{w}(v \oplus \bullet) = \mathsf{w}(\bullet \oplus v) = \mathsf{w}(v)$$
$$\mathsf{w}(v\,\xi) = \mathsf{w}(v) \times \mathsf{w}(\xi) \quad \mathsf{w}([v_1, \dots, v_n]) = n! \times \prod_{i \in [n]} \mathsf{w}(v_i).$$

Definition 22. The *Taylor expansion* of a $\lambda_{\mathrm{nd}}\mathbf{Y}$-term M is

$$M^* \;:=\; \sum_{v \blacktriangleleft M} \frac{1}{\mathsf{w}(v)} |v|$$

where v ranges over representations.

It is not difficult to see that this coincides with the standard definition. The definition can be easily extended to Böhm trees. The above formal sum may not converge for Böhm trees.

$$\frac{}{x \blacktriangleleft x} \quad \frac{v \blacktriangleleft M}{\lambda x.v \blacktriangleleft \lambda x.M} \quad \frac{v \blacktriangleleft M \qquad \xi \blacktriangleleft N}{v\,\xi \blacktriangleleft M\,N} \quad \frac{v \blacktriangleleft M_1}{v \oplus \bullet \blacktriangleleft M_1 \oplus M_2} \quad \frac{v \blacktriangleleft M_2}{\bullet \oplus v \blacktriangleleft M_1 \oplus M_2} \quad \frac{v \blacktriangleleft M \qquad \xi \blacktriangleleft \mathbf{Y}\,M}{v\,\xi \blacktriangleleft \mathbf{Y}\,M} \quad \frac{\forall i \in [n].\ v_i \blacktriangleleft N}{[v_1, \dots, v_n] \blacktriangleleft N}$$

Fig. 4. Approximation relation

B. Marked rigid resource terms

Interestingly, in order to describe the rigid Taylor expansion (or the ESP interpretation) of a standard resource term, we need to use a variant of rigid resource calculus that we call the *marked rigid resource calculus*. This reflects a difference between the symmetries associated to the standard resource calculus and the symmetries associated to the $\lambda\mathbf{Y}$-calculus: see Remark 23.

In the new syntax, a list is annotated with a permutation: $\mu := \langle t_1, \dots, t_n \rangle^\sigma$ where $\sigma \in \mathfrak{S}_n$. This annotation σ does not affect the operational semantics, which simply ignores σ, but the action of a type isomorphism. Consider the list $\mu = \langle (\lambda x.x), (\lambda y.y) \rangle$ in the old syntax, whose elements are identical (as we identify α-equivalent terms), and $a = \langle \star \rangle \multimap \star$. Let $\varphi : \langle a, a \rangle \cong \langle a, a \rangle$ be the type isomorphism swapping the elements. Then $[\varphi]\mu = \langle \lambda y.y, \lambda x.x \rangle = \mu$ (using α-conversion), so μ is a fixed-point of $[\varphi](-)$. In other words the old syntax cannot notice if the elements were really swapped. The aim of the new syntax is to detect such swapping: $[\varphi](\langle (\lambda x.x), (\lambda y.y) \rangle^{\mathrm{id}})$ is defined as $\langle (\lambda y.y), (\lambda x.x) \rangle^\sigma$ (where σ swaps 1 and 2), which has a different annotation.

We use $(-)^\circ$ to express that it is marked. For example, a marked rigid resource raw term is written as u°.

Remark 23. The necessity of the marks can be intuitively understood as follows. The marks are needed because a resource term distinguishes different occurrences of the same term in a bag. For example, consider $(\lambda f.f\,[f\,[z]])\,[\lambda x.x, \lambda x.x]$. This is reduced to $(\lambda x.x)[(\lambda x.x)[z]] + (\lambda x.x)[(\lambda x.x)[z]]$. The one is obtained by substituting the left occurrence of $\lambda x.x$ to the left occurrence of f, and the other is by the left occurrence of $\lambda x.x$ to the right occurrence of f; here we distinguish the occurrences of $\lambda x.x$.

C. ESP interpretation of standard resource terms

Let v be a representation of a resource term in η-long form with no $\oplus$. As before, we write $\vec{z}$ for a prefix of a fixed infinite sequence $z_1, z_2, \dots$. The rigid Taylor expansion $\llbracket v \rrbracket$ of v is defined by the rules in Fig. 5 and

$$\llbracket v \rrbracket(\Delta, a) := \{\, \widetilde{t}^\circ \mid \vec{z} \lhd \vec{x} \vdash t^\circ \lhd v \text{ and } \vec{z} : \vec{b} \vdash t^\circ : a \,\}$$

where $\Delta = (\vec{z} : \vec{b})$. This is a generalised species by the action of isomorphisms to rigid resource terms defined in Section IV.

It is obvious to see that the permutation of elements induces a natural isomorphism: if v and v' represent the same resource term, then $\llbracket v \rrbracket \cong \llbracket v' \rrbracket$. So $\llbracket \cdot \rrbracket$ is well-defined for resource terms (provided that we do not refer to the annotations; it is possible to check if the two annotations are equivalent, since the equivalence of annotations is preserved by the natural isomorphism mentioned above).

This interpretation is sound with respect to the reduction of the resource term. Similarly to the case of $\lambda_{\mathrm{nd}}\mathbf{Y}$, a marked rigid resource term $\widetilde{u}^\circ \lhd v$ resolves nondeterminism by, in this case, a mapping from elements in a bag to occurrences of a variable.

Lemma 24. *Let v be a resource term and assume that $\mathsf{nf}(v) = \sum_k w_k$. Then there exists a natural isomorphism*

$$\alpha_{\Delta,a} : \llbracket v \rrbracket(\Delta, a) \xrightarrow{\cong} \coprod_k \llbracket w_k \rrbracket(\Delta, a)$$

such that $\alpha_{\Delta,a}(\widetilde{s}^\circ) = \iota_k(\widetilde{u}^\circ)$ implies $\mathsf{nf}(\widetilde{s}^\circ) = \widetilde{u}^\circ$. Here ι_k is the k-th injection.

Let $\llparenthesis - \rrparenthesis_X$ be the map from rigid resource terms to standard resource terms, which replaces a list $\langle u_1, \dots, u_n \rangle$ with a bag $[\llparenthesis u_1 \rrparenthesis_X, \dots, \llparenthesis u_n \rrparenthesis_X]$, an abstraction $\lambda x_1 \dots x_n.t$ with $\lambda y.\llparenthesis t \rrparenthesis_{X, x_1 \lhd y, \dots, x_n \lhd y}$ and a variable x with y s.t. $(x \lhd y) \in X$. We write $\#\mathsf{iso}(\Delta, a)$ for the number of isomorphisms, i.e. the number of elements in $\{(\psi, \varphi) \mid \psi : \Delta \cong \Delta' \text{ and } \varphi : a \cong a'\}$.

Lemma 25. *Let v be a resource term in normal form.*

$$v \;=\; \sum_{(\vec{x}:\vec{b} \vdash \widetilde{u}^\circ : a) \in \llbracket v \rrbracket} \frac{1}{\#\mathsf{iso}((\vec{x} : \vec{b}), a)} \llparenthesis \widetilde{u}^\circ \rrparenthesis_{\vec{z} \lhd \vec{x}}$$

Proof. (Sketch) This follows from the following observations: (i) if v is normal, $\llbracket v \rrbracket$ has a single orbit (i.e. any marked rigid approximations in $\llbracket v \rrbracket$ are related by an isomorphism) and (ii) isomorphisms (except for the identity) do not have a fixed-point because of the annotations on $\widetilde{u}^\circ$. □

The next result follows from Lemmas 24 and 25.

Corollary 26. *For every simple resource term v,*

$$\mathsf{nf}(v) \;=\; \sum_{(\vec{x}:\vec{b} \vdash \widetilde{u}^\circ : a) \in \llbracket v \rrbracket} \frac{1}{\#\mathsf{iso}((\vec{x} : \vec{b}), a)} \llparenthesis \mathsf{nf}(\widetilde{u}^\circ) \rrparenthesis_{\vec{z} \lhd \vec{x}}.$$

D. Reasoning about the standard Taylor expansion

The following lemma is an explicit computation of the Taylor expansion of the Böhm tree of a given term.

Lemma 27.

$$BT(M)^* \;=\; \sum_{(\vec{x}:\vec{b} \vdash \widetilde{t}:a) \in \llbracket BT(M) \rrbracket} \frac{1}{\#\mathsf{iso}((\vec{x} : \vec{b}), a)} \llparenthesis \widetilde{t} \rrparenthesis_{\vec{z} \lhd \vec{x}}$$

Proof. The claim is essentially the same as [16, Lemma 13] because the action of an isomorphism on rigid terms in *normal form* is a combination of swapping of variable occurrences and permutation of lists. □

The next theorem follows from Corollary 26, Lemma 27 and Theorem 17 with a careful calculation; the proof is omitted because of the page limit.

Theorem 28. $\mathsf{nf}(M^*) \;=\; BT(M)^*$.

$$\frac{}{y \lhd x \vdash y \lhd x} \qquad \frac{X, \vec{y} \lhd x \vdash t^\circ \lhd v \quad x \notin X}{X \vdash \lambda \vec{y}.t^\circ \lhd \lambda x.v} \qquad \frac{X_1 \vdash t^\circ \lhd v \quad X_2 \vdash \mu^\circ \lhd \xi}{X_1, X_2 \vdash t^\circ\, \mu^\circ \lhd v\, \xi} \qquad \frac{\sigma \in \mathfrak{S}_n \qquad \forall i \in [n].\ X_i \vdash t_i^\circ \lhd v_{\sigma(i)}}{X_1, \dots, X_n \vdash \langle t_1^\circ, \dots, t_n^\circ \rangle^\sigma \lhd [v_1, \dots, v_n]}$$

Fig. 5. The rules of the rigid Taylor expansion of the standard resource terms. Exchange rule of the assumptions in X is omitted.

VII. Future work

For future work, we aim to develop a theory of "weighted generalised species". For example, an $\mathbf{R}$-weighted generalised species, where $\mathbf{R}$ is the ring of real numbers, is a generalised species $f : \mathbb{P}\mathcal{A} \times \mathcal{B}^{\mathrm{op}} \to \mathbf{Set}$ together with the *weight function* $w_{a,b} : f(a,b) \to \mathbf{R}$ ($a \in \mathbb{P}\mathcal{A}, b \in \mathcal{B}$) that is preserved by the action of morphisms in $\mathbb{P}\mathcal{A}$ and $\mathcal{B}$. It is natural to expect that a probabilistic programming language can be modelled by using $\mathbf{R}$-weighted generalised species, where the weight function associates a run of a program to its probability, and this idea is a reminiscent of the existing models (e.g. [8, 9, 17, 38]). Furthermore we think that any SMCC can be used as the domain of weights. In fact, the rigid Taylor expansion can be seen as generalised species with weights from the linear λ-calculus and this observation suggests to us the possibility of replacing the rigid resource terms by morphisms of the chosen SMCC. We are interested in the SMCC $\mathbf{CPM}$ [36] and conjecture that $\mathbf{CPM}$-weighted generalised species serve as a model of a quantum programming language, which should be closely related to [34].

Acknowledgement

The authors are grateful to Marcelo Fiore for insightful comments. They would also like to thank the anonymous reviewers for their helpful and constructive comments. This work was supported by JSPS KAKENHI Grant Numbers JP15H05706 and JP16K16004, and EPSRC EP/M023974/1.

References

[1] S. Abramsky, R. Jagadeesan, and P. Malacaria, "Full abstraction for PCF," *Inf. Comput.*, vol. 163, no. 2, pp. 409–470, 2000. I-E
[2] A. Allioux, "Krivine machine and Taylor expansion in a non-uniform setting," in *LINEARITY*, 2016, pp. 24–32. I-E
[3] J. Bénabou, "Les distributeurs," Tech. Rep. 33, 1973. V-A
[4] ——, "Distributors at work," 2000, course notes, TU Darmstadt. V-A
[5] A. Bernadet and S. Lengrand, "Non-idempotent intersection types and strong normalisation," *LMCS*, vol. 9, no. 4, 2013. I-E
[6] F. Borceux, *Handbook of Categorical Algebra I*. CUP, 1994. V-A
[7] A. Bucciarelli and T. Ehrhard, "On phase semantics and denotational semantics: the exponentials," *Ann. Pure Appl. Logic*, vol. 109, no. 3, pp. 205–241, 2001. I-E, 3
[8] V. Danos and T. Ehrhard, "Probabilistic coherence spaces as a model of higher-order probabilistic computation," *Inf. Comput.*, vol. 209, no. 6, pp. 966–991, 2011. I-A, I-A, I-E, VII
[9] V. Danos and R. S. Harmer, "Probabilistic game semantics," *ACM Trans. Comp. Logic*, vol. 3, no. 3, pp. 359–382, 2002. VII
[10] D. de Carvalho, "Execution time of lambda-terms via denotational semantics and intersection types," *CoRR*, vol. abs/0905.4251, 2009. I-E, 8
[11] T. Ehrhard, "On Köthe sequence spaces and linear logic," *Mathematical Structures in Computer Science*, vol. 12, no. 5, pp. 579–623, 2002. I, I-B
[12] ——, "Finiteness spaces," *Mathematical Structures in Computer Science*, vol. 15, no. 4, pp. 615–646, 2005. I, I-B
[13] ——, "A finiteness structure on resource terms," in *LICS*, 2010, pp. 402–410. I-C
[14] T. Ehrhard and L. Regnier, "The differential lambda-calculus," *Theoretical Computer Science*, vol. 309, no. 1-3, pp. 1–41, 2003. I
[15] ——, "Böhm trees, krivine's machine and the taylor expansion of lambda-terms," in *CiE*, ser. Lecture Notes in Computer Science, A. Beckmann, U. Berger, B. Löwe, and J. V. Tucker, Eds., vol. 3988. Springer, 2006, pp. 186–197. I-C
[16] ——, "Uniformity and the Taylor expansion of ordinary lambda-terms," *Theoretical Computer Science*, vol. 403, no. 2-3, pp. 347–372, 2008. I, I-B, I-C, I-E, IV, VI, VI-A, VI-D
[17] T. Ehrhard, C. Tasson, and M. Pagani, "Probabilistic coherence spaces are fully abstract for probabilistic PCF," in *POPL*, 2014, pp. 309–320. I-A, I-A, I-E, VII
[18] M. P. Fiore, "Mathematical models of computational and combinatorial structures," in *FoSSaCS*, 2005, pp. 25–46. I, I-D, 2, III-C, V-B
[19] M. P. Fiore, N. Gambino, J. M. E. Hyland, and G. Winskel, "The cartesian closed bicategory of generalised species of structures," *J. London Math. Soc.*, vol. 77, pp. 203–220, 2008. I, I-D, 2, III-C, V, V-B, 18, V-C
[20] J. Girard, "Linear logic," *Theor. Comput. Sci.*, vol. 50, pp. 1–102, 1987. I
[21] ——, "Normal functors, power series and λ-calculus," *Ann. Pure Appl. Logic*, vol. 37, no. 2, pp. 129–177, 1988. I, I-B
[22] R. Harmer and G. McCusker, "A fully abstract game semantics for finite nondeterminism," in *LICS*. IEEE Comput. Soc, 1999, pp. 422–430. I-A
[23] R. Hasegawa, "Two applications of analytic functors," *Theoretical Computer Science*, vol. 272, no. 1-2, pp. 113–175, 2002. I
[24] J. M. E. Hyland and C.-H. L. Ong, "On full abstraction for PCF: I, II, and III," *Info. & Comp.*, vol. 163, no. 2, pp. 285–408, 2000. I-E
[25] A. Joyal, "Une théorie combinatoire des séries formelles," *Adv. Math.*, vol. 42, p. 182, 1981. I, V
[26] ——, "Foncteurs analytiques et espèces de structures," in *Combinatoire Énumérative*. Springer, 1986, pp. 126–159. V
[27] A. Kfoury and J. Wells, "Principality and type inference for intersection types using expansion variables," *TCS*, vol. 311, no. 1-3, pp. 1–70, 2004. I-E, III-B, 5
[28] A. J. Kfoury, "A linearization of the lambda-calculus and consequences," *J. Logic & Comp.*, vol. 10, no. 3, pp. 411–436, 2000. I-E, 3, III
[29] J. Laird, G. Manzonetto, G. McCusker, and M. Pagani, "Weighted relational models of typed lambda-calculi," in *LICS*, 2013, pp. 301–310. I-E
[30] D. Mazza, "Affine approximations and intersection types," 2016, contributed talk, ITRS 2016. I-E, III
[31] P.-A. Melliès, "Asynchronous games 1: Uniformity by group invariance," 2003, unpublished manuscript. I-E
[32] P. M. Neergaard and H. G. Mairson, "Types, potency, and idempotency: why nonlinearity and amnesia make a type system work," in *ICFP*, C. Okasaki and K. Fisher, Eds. ACM, 2004, pp. 138–149. I-E, 2, 3, III
[33] C.-H. L. Ong and T. Tsukada, "Two-level game semantics, intersection types, and recursion schemes," in *ICALP*, 2012, pp. 325–336. I-E
[34] M. Pagani, P. Selinger, and B. Valiron, "Applying quantitative semantics to higher-order quantum computing," in *POPL*, 2014, pp. 647–658. I-A, I-A, I-E, VII
[35] M. Pagani, C. Tasson, and L. Vaux, "Strong normalizability as a finiteness structure via the Taylor expansion of λ-terms," in *FoSSaCS*, 2016, pp. 408–423. I-C
[36] P. Selinger and B. Valiron, "On a fully abstract model for a quantum linear functional language: (extended abstract)," *Electr. Notes Theor. Comput. Sci.*, vol. 210, pp. 123–137, 2008. VII
[37] T. Tsukada and K. Asada, "Strategies in HO/N games as profunctors," 2016, contributed talk, GaLoP Workshop. I-E
[38] T. Tsukada and C.-H. L. Ong, "Innocent strategies are sheaves over plays - deterministic, non-deterministic and probabilistic innocence," *CoRR*, vol. abs/1409.2764, 2014. VII
[39] ——, "Nondeterminism in game semantics via sheaves," in *LICS*, 2015. I-A, I-E