Species, Profunctors and Taylor Expansion Weighted by SMCC

A Unified Framework for Modelling Nondeterministic, Probabilistic and Quantum Programs

Takeshi Tsukada
The University of Tokyo

Kazuyuki Asada
The University of Tokyo

C.-H. Luke Ong
University of Oxford

Abstract

Motivated by a tight connection between Joyal's combinatorial species and quantitative models of linear logic, this paper introduces *weighted generalised species* (or *weighted profunctors*), where weights are morphisms of a given symmetric monoidal closed category (SMCC). For each SMCC $\mathcal{W}$, we show that the category of $\mathcal{W}$-weighted profunctors is a Lafont category, a categorical model of linear logic with exponential. As a model of programming languages, the construction of this paper gives a unified framework that induces adequate models of nondeterministic, probabilistic, algebraic and quantum programming languages by an appropriate choice of the weight SMCC.

CCS Concepts •**Theory of computation** →*Linear logic; Denotational semantics;*

Keywords quantitative model, quantum computation, generalised species, weighted species, rigid resource calculus

ACM Reference format:
Takeshi Tsukada, Kazuyuki Asada, and C.-H. Luke Ong. 2018. Species, Profunctors and Taylor Expansion Weighted by SMCC. In *Proceedings of LICS '18: 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, Oxford, United Kingdom, July 9–12, 2018 (LICS '18),* 10 pages.
DOI: 10.1145/3209108.3209157

1 Introduction

Semantics of programming languages with branching constructs such as nondeterministic, probabilistic, algebraic and quantum programming languages (e.g. [5, 7, 13, 28, 33]) is an important area of current interest. The aim of this paper is to give a unified framework for modelling these languages.

This paper is, of cause, not the first work that addresses this problem. Among others, several models (e.g. [5, 7, 21, 28]) have been constructed using the techniques of quantitative models of linear logic. For example, the probabilistic coherence space model [5, 7] is a fully abstract model for probabilistic PCF; the weighted relational model [21] gives a unified account of nondeterministic, probabilistic and algebraic programs; and Pagani et al. [28] give a model of higher-order quantum programs.

This paper proposes a general model construction of which [21] and [28] are instances in a certain sense. A notable conceptual difference is that, building on [8, 9, 32], our construction is a cross-fertilization between combinatorial species and (quantitative) models of λ-calculus.

LICS '18, Oxford, United Kingdom

DOI: 10.1145/3209108.3209157

1.1 Why combinatorics matters?

Let us first explain why combinatorics ideas would be useful at the intuitive level. We shall see that a combinatorics problem naturally arises in the operational semantics of programs.

Consider, for example, a programming language with probabilistic branching.

Given a closed term P of the unit type, the probability of convergence is usually defined as follows. First we define the set $\mathit{Eval}(P)$ of reduction sequences $\pi : P \longrightarrow^* ()$, where $()$ is the unique value of the unit type and π is the name of this reduction sequence. Because of the branching construct, P may have many reduction sequences. Each reduction sequence $\pi \in \mathit{Eval}(P)$ is associated with a real number $\varpi(\pi)$ between 0 and 1, called its *probability* or *weight*. Hence $\mathit{Eval}(P)$ is not merely a set but a *weighted set*. Then the probability of convergence is defined as the sum $\sum_{\pi \in \mathit{Eval}(P)} \varpi(\pi)$. We aim to apply combinatorics techniques to enumerate the elements of $\mathit{Eval}(P)$ and to compute their weights.

The difference of the branching constructs (i.e. differences of nondetermisitic, probabilistic, algebraic or quantum programs) is understood as the difference of the domains of weights. For example, for a nondeterministic program, a weight is an element of the two-valued Boolean algebra; the weight function is defined by $\varpi(\pi) = \mathtt{true}$ for every reduction sequence π and the sum is the disjunction; then $\sum_{\pi : M \longrightarrow^* V} \varpi(\pi) = \mathtt{true}$ if and only if $\exists \pi . \pi : M \longrightarrow^* V$. This framework applies also to quantum programs as we shall see.

1.2 Two extensions of Joyal's combinatorial species

The combinatorics tool that we employ for computing $\mathit{Eval}(P)$ is based on Joyal's *combinatorial species* [17] (see also a textbook [4]), which is a functor $F : \mathbf{P} \to \mathbf{Set}$ from the category $\mathbf{P}$ of finite cardinals and bijections. This notion is, indeed, closely related to Girard's *normal functor semantics* [10], pioneering work on quantitative models (see, e.g., [14] for the relationship). To the purpose of this paper, we need its weighted and higher-order extension: the weight is used to handle weights $\varpi(\pi)$ of $\pi \in \mathit{Eval}(P)$ and higher-order feature is used to deal with higher-order constructs of programs.

There have been extensions in each direction.

Given a set W of weights, a *W-weighted species* (see, e.g., a textbook [4]) is a species $F : \mathbf{P} \to \mathbf{Set}$ together with a family of functions $\varpi_n : F(n) \to W$ that respect the action of permutation. Many ideas and operations for species can be naturally extended to weighted species. Usually W is assumed to have an algebraic structure such as ring; we shall discuss below an appropriate algebraic structure for W in our setting.

Generalised species [8, 9] is a higher-order extension: Joyal's species can be seen as generalised species of type $I \to I$ (where I is the unit type). Formally it is a profunctor $F : !\mathcal{A} \nrightarrow \mathcal{B}$ (i.e. a functor $F : \mathcal{B}^{\mathrm{op}} \times !\mathcal{A} \to \mathbf{Set}$), where $!$ is a linear exponential comonad on the bicategory **Prof** of profunctors. This can be seen as a "proof-relevant version" of the relational model of linear logic.

Our previous work [32] shows that the interpretation of a program P in **Prof** is the set $Eval(P)$ (without weights).

This paper develops a common extension of the two, which we call *weighted generalised species* or *weighted profunctor*.

1.3 Key notion: weighted generalised species

The naïve combination of the above ideas leads us to consider a profunctor $F : \mathcal{A} \nrightarrow \mathcal{B}$ with a family $\varpi_{b,a} : F(b,a) \to W$ of functions parameterised by objects $a \in \mathrm{ob}(\mathcal{A}), b \in \mathrm{ob}(\mathcal{B})$, where W is a fixed set of weights. However this simple notion does not seem to suffice for modelling quantum programs.

This paper considers the situation in which the weight W varies with a and b. The weight is not a set but a category $\mathcal{W}$; an object is not a category but a functor $A : \mathcal{A} \to \mathcal{W}^{\mathrm{op}}$, and a morphism from $A : \mathcal{A} \to \mathcal{W}^{\mathrm{op}}$ to $B : \mathcal{B} \to \mathcal{W}^{\mathrm{op}}$ is what we call a *weighted profunctor* from A to B, which consists of a pair of a profunctor $F : \mathcal{A} \nrightarrow \mathcal{B}$ and a family $\varpi_{b,a} : F(b,a) \to \mathcal{W}^{\mathrm{op}}(B(b), A(a))$. To understand this construction, we note that an element $x \in F(b,a)$ of a profunctor can be seen as a "morphism" from b to a (see, e.g., [3]); then the above construction associates a "morphism" from b to a with a real morphism $\varpi_{b,a}(x) : B(b) \to A(a)$ in $\mathcal{W}^{\mathrm{op}}$ (i.e. a real morphism $A(a) \to B(b)$ in $\mathcal{W}$). Another syntactic exposition based on a *rigid* variant [32] of the *Taylor expansion* [6] will be given in Sections 3 and 4.

The relevance of this construction is justified by the following facts: (1) The resulting category has a good structure, namely, Lafont category with biproducts if the weight category $\mathcal{W}$ is an SMCC. (2) The construction has a concise categorical definition, as (the classifying 1-category of) a full sub-bicategory of the lax slice bicategory $\mathbf{Prof}/\!\!/\mathcal{W}^{\mathrm{op}}$. (3) The construction gives us an adequate model of a programming language, in which the interpretation of a program has a syntactic counterpart, the *rigid Taylor expansion*, by which a program is interpreted as a collection of its linear approximations.

1.4 Generating series and matrices

Calculation of species and profunctors is often cumbersome. To ease the computation, in the context of weighted species, one can use the *generating series*. Let R be a ring and assume a weighted species $F : \mathbf{P} \to \mathbf{Set}$ with $\varpi_n : F(n) \to R$ such that $F(n)$ is finite for every n. Its *(exponential) generating series* is defined as $\|(F,\varpi)\| = \sum_{n=0}^{\infty} \|(F,\varpi)\|_n \, z^n$, where z is the indeterminant and the coefficient is defined by $\|(F,\varpi)\|_n := (1/n!) \sum_{x \in F(n)} \varpi_n(x)$. Many operations can be carried out in this generating series representation.

Motivated by this idea, this paper develops a concise representation for (a subclass of) weighted profunctors. It is a matrix indexed by objects of $\mathcal{A}$ and $\mathcal{B}$ whose (a,b)-entry is defined by $\|(F,\varpi)\|_{b,a} := (1/\#G) \sum_{x \in F(b,a)} \varpi_{b,a}(x)$ where G is a group describing symmetries of a and b.

This construction is applicable only if the weight category $\mathcal{W}$ has sufficient structure. For example, the sum $\sum_{x \in F(b,a)} \varpi_{b,a}(x)$ of morphisms in $\mathcal{W}(A(a), B(b))$ must be defined in order for the above definition to make sense. We characterise sufficiency of structure in terms of *enriched category theory*, namely, enrichment by Σ-*monoids* (a class of algebras with countable sum): if the SMCC structure of $\mathcal{W}$ is enriched by Σ-monoids and satisfies an additional requirement, then all computation of the Lafont category can be carried out in the matrix representation.

1.5 Contributions

This paper introduces *weighted generalised species* and *weighted profunctors* parametrised by the weight SMCC $\mathcal{W}$. The category $\mathbf{Pr}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}}$ of $\mathcal{W}$-weighted profunctors is a model of linear logic (namely a Lafont category with biproducts) and an adequate model of a calculus $\lambda_{\mathcal{W}}$, into which nondeterministic, probabilistic, algebraic and quantum programs can be embedded when $\mathcal{W}$ is appropriately chosen. Assuming additional structures for $\mathcal{W}$, this paper defines a *category of matrices* $\mathbf{Mat}(\mathcal{W})$ over $\mathcal{W}$, which is also a model of linear logic and an adequate model of $\lambda_{\mathcal{W}}$. This construction generalises that of a weighted relational model [21] and of a model of quantum programs by Pagani et al. [28] (see Remark 5.12).

1.6 Related work

The relational model **MRel** [10] is perhaps the prototypical quantitative model of the lambda calculus. In an effort to generalise Girard's *quantitative domains* [10], Lamarche introduced an important extension of the relational model, namely, the category of *weighted relations* over a complete commutative semiring [22]. Characterised as the free biproduct completion of the weight semiring, the weighted relational model was further developed by Laird et al. in a series of papers [19–21]. By an appropriate choice of the weight semiring, these weighted relational models give an adequate semantics of nondeterministic and probabilistic PCF, with scalar weights from the semiring.

For modelling probabilistic PCF, a related semantics, based on *probabilistic coherence spaces* [5], was shown to be fully abstract by Ehrhard et al. [7]. In the latter paper (§5.1), the authors drew a comparison between the probabilistic coherence spaces interpretation and the sum of weights of intersection type derivations. Since linear approximations (of our present paper) can be viewed as derivations in an intersection type system, summation of the weights of all derivations can be related to the generating series (or the matrix representation) of a weighted profunctor (or the rigid Taylor expansion). In this sense, our paper confirms the observation of Ehrhard et al. in [7] from a somewhat more general perspective. A connection [7, Footnote 7, p. 313] between the probabilistic coherence spaces and the combinatorial species interpretation [14, 17, 18] is similarly clarified by our work.

In [28], Pagani et al. applied the free biproduct construction to the known model of completely positive maps to obtain an adequate semantics of an expressive quantum lambda calculus. A notable advance in the denotational semantics of higher-order quantum computation, their model can interpret not just infinitary computation (both infinite data types and recursion), but also general entanglement, a defining feature of quantum computation. In the Conclusion section of the paper [28], the authors observed that their model "demonstrates that the quantum and the classical 'universes' work well together, but also—surprisingly—that they do not mix too much, even at the higher-order types." Our work clarifies this phenomenon mathematically by organising the modelling process into two phases, namely, enumeration and summation. The reason why the model supports a certain clean separation of the two worlds (always yielding "an *infinite* list of *finite*-dimensional CPMs") can be traced to the fact that the category $\mathbf{CPM}_s$ is $\Sigma\mathbf{Mon}$-enriched, and, in particular, to the presence of the element "$1/n$" in the monoid, for every natural number n (see Section 5 for the

precise formulation). In fact, in the semantics, different control flows (that we do not need to distinguish) are merged.

The relational model may be generalised in quite a different way, namely, to a 2-dimensional level categorically. As set out by Fiore [8], the conceptual basis for this class of 2-categorical models of higher-order computation lies in combinatorics and its methods. In a follow-up paper [9], Fiore et al. introduced the cartesian closed bicategory of *generalised species of structures*, which generalises both Joyal's combinatorial species [17, 18] and Girard's normal functors semantics [10], and may be viewed as a proof-relevant extension of the relational model. In recent work [32], we introduced *rigid resource calculus*, and showed that the Taylor expansion semantics (within the rigid calculus) of the nondeterministic $\lambda\mathbf{Y}$-calculus coincides with the generalised species interpretation.

Building on the correspondence between linear approximations and non-idempotent intersection types, Mazza et al. [24, 29] have recently developed a general 2-operadic framework for deriving systems of intersection types that characterise normalisation properties, based on a **Rel**-valued profunctorial semantics of programs. It would be interesting to clarify how their semantics relates to the generalised species interpretation [9] (or equivalently the rigid Taylor expansion semantics [32]), and to generalise their main result [24, Theorem 4.7] to programs with such branching constructs as nondeterministic and probabilistic choice.

Melliès [25] has analysed the group-theoretic nature of the PER construction in the AJM game model [1]: his *orbital game* is a reformulation of HO-style arena games [16] with justification pointers replaced by thread indexing, modulo certain left and right group actions. A similar idea appears in our Section 5. Symmetry in a similar spirit can also be found in the model of quantum computation by Pagani et al. [28], whose construction requires invariance under certain group actions.

2 A Lambda Calculus with SMCC Data

Assume a symmetric monoidal closed category $(\mathcal{W}, \otimes, \multimap, I)$, which we call the *weight category*. Based on the typed calculus in Pagani et al. [28], this section introduces a lambda calculus $\lambda_{\mathcal{W}}$ parameterised by SMCC $\mathcal{W}$, which has the objects of $\mathcal{W}$ as base types and the morphisms as constants. The standard constructs of the lambda calculus describes "classical" control, whereas constants from $\mathcal{W}$ manipulates "non-classical" data. A goal of Sections 3, 4 and 5 is to give an adequate model of $\lambda_{\mathcal{W}}$.

The calculus $\lambda_{\mathcal{W}}$ is used as a metalanguage, which is not necessarily of practical interest but fits our model well. Its usefulness is demonstrated by embedding calculi of interest into $\lambda_{\mathcal{W}}$ with appropriate $\mathcal{W}$, adequately though not necessarily fully. A key example of $\mathcal{W}$ is the category $\mathbf{CPM}_s$, which is a model of a linear and finite quantum programming language (see [30, 31] for an account of this category as a model of quantum programs). The higher-order quantum calculus of [28] can be embedded into $\lambda_{\mathbf{CPM}_s}$.

For brevity, we often treat $\mathcal{W}$ as if it were a strict SMCC.

2.1 Syntax

Figure 1 shows the syntax of the calculus. The type constructors of the calculus are those of intuitionistic linear logic with base types a, which are objects of $\mathcal{W}$. The term constructors are the standard ones of a λ-calculus with coproduct types, constructors and a destructor of lists, nondeterministic branching $M \diamond N$, sequential execution $(M; N)$, recursion $\mathbf{Y}\, V$ and constants c^S from $\mathcal{W}$; here either $S = \mathsf{a}_1 \otimes \cdots \otimes \mathsf{a}_n \multimap \mathsf{b}_1 \otimes \cdots \otimes \mathsf{b}_m$ and $c \in \mathcal{W}(\mathsf{a}_1 \otimes \cdots \otimes \mathsf{a}_n, \mathsf{b}_1 \otimes \cdots \otimes \mathsf{b}_m)$, or $S = \mathsf{b}_1 \otimes \cdots \otimes \mathsf{b}_m$ and $c \in \mathcal{W}(I, \mathsf{b}_1 \otimes \cdots \otimes \mathsf{b}_m)$. For technical convenience, the arguments of many constructs are restricted to values. This does not lose generality; for example, the term $\mathtt{inl}(M)$ can be written as $\mathtt{let}\, x = M \,\mathtt{in}\, \mathtt{inl}(x)$ for fresh x. We use $M\,V$ as the syntactic sugar of $\mathtt{let}\, x = M \,\mathtt{in}\, x\, V$ for fresh x. We shall often omit type annotations.

The calculus has a type system based on the dual context linear logic . A judgement has the form $\Delta \mid \Gamma \vdash M : S$, where Δ and Γ are finite sequences of type bindings of the form $x : T$ called *type environments*. The variables in Δ and in Γ are non-linear and linear ones, respectively. The typing rules are standard, some of which are listed in Fig. 2.

2.2 Operational semantics

A *configuration* (typically C, C' etc.) is a triple of sequences $\vec{x} = x_1, \ldots, x_n$ of variables and $\vec{\mathsf{a}} = \mathsf{a}_1, \ldots, \mathsf{a}_n$ of atomic types, a morphism $e : I \to \mathsf{a}_1 \otimes \cdots \otimes \mathsf{a}_n$ in $\mathcal{W}$ and a term $\mid x_1 : \mathsf{a}_1, \ldots, x_n : \mathsf{a}_n \vdash M : I$ (note that x_i is a linear variable). We write such a triple as $[\vec{x} = e, M]$, which intuitively means $\mathtt{let}\, \vec{x} = e \,\mathtt{in}\, M$.

The set of *evaluation contexts* is defined by the following grammar: $E ::= [\,] \mid E; M \mid \mathtt{let}\, x = E \,\mathtt{in}\, M$. The *one-step evaluation relation* on configurations is given by the rules in Fig. 3. For a sequence $\pi \in \{0, 1, 2\}^*$, we write $C \xrightarrow{\pi} C'$ if there is a sequence of the form $C = C_0 \xrightarrow{d_1} C_1 \xrightarrow{d_2} \cdots \xrightarrow{d_n} C_n = C'$ and $\pi = d_1\, d_2 \ldots d_n$ where $n \geq 0$ (ϵ is the empty sequence and hence the length of π may be less than n).

A *program* P is a closed term of the unit I. We define

$$Eval(P) := \{\, \pi \mid [\epsilon = \mathrm{id}_I, P] \xrightarrow{\pi} [\epsilon = e, ()] \,\}.$$

For $\pi \in Eval(P)$, its *weight* $w(\pi)$ is (necessarily unique) $e \in \mathcal{W}(I, I)$ such that $[\epsilon = \mathrm{id}_I, P] \xrightarrow{\pi} [\epsilon = e, ()]$. Let us call a set X equipped with a function $w : X \to W$ a W-*weighted set* (or simply a *weighted set*). In this terminology $Eval(P)$ is a $\mathcal{W}(I, I)$-weighted set.

In a typical situation, we are not interested in the weighted set $Eval(P)$ itself but its summary. For example, if $\mathcal{W}(I, I)$ has sums, it may be more appropriate to consider the sum $\sum_{\pi \in Eval(P)} w(\pi)$; see the examples in the next subsections.

2.3 Examples

Example 2.1 (Nondeterministic calculus)**.** Let $\mathcal{W}$ be the terminal category 1 consisting of one object I and one morphism (i.e. the identity on I), which has the trivial SMCC structure. The calculus λ_1 has nothing special except for the nondeterministic branching. A closely related variant is given by a category $\mathbf{B}$ consisting of one object I and two morphisms $0, 1 \in \mathbf{B}(I, I)$, regarded as the two-value boolean algebra, with composition given by the meet. *May-convergence of* P is defined as $\bigvee_{\pi \in Eval(P)} w(\pi)$. The calculus λ_1 can be embedded into $\lambda_{\mathbf{B}}$.

Example 2.2 (Probabilistic calculus)**.** If the calculus has a probabilistic branching, each reduction sequence is associated with its *probability*, i.e. a real number p with $0 \leq p \leq 1$. This observation motivates us to consider the weight category $\mathcal{W}_{[0,1]}$ consisting of one object I and $\mathcal{W}(I, I) = [0, 1]$, where $[0, 1] = \{x \in \mathbf{R} \mid 0 < x \leq 1\}$ is the interval of real numbers, with composition defined by the multiplication. In this calculus one can express, for example, the probabilistic choice of M and N as $(\frac{1}{2}; M) \diamond (\frac{1}{2}; N)$, where $\frac{1}{2} : I$ is

$$S,T ::= \mathsf{a} \mid S \multimap T \mid I \mid S \otimes T \mid !S \mid S \oplus T$$
$$M,N,L ::= V \mid V\,W \mid M \diamond N \mid \mathbf{Y}\,V \mid M;N \mid \mathtt{let}\,x = M\,\mathtt{in}\,N \mid \mathtt{let}\,!x = V\,\mathtt{in}\,M \mid \mathtt{let}\,x \otimes y = V\,\mathtt{in}\,M \mid \mathtt{case}\,V\,\mathtt{of}\,(\mathtt{inl}(x) : N \mid \mathtt{inr}(y) : L)$$
$$V,W ::= x \mid c \mid \lambda x^A.M \mid () \mid V \otimes W \mid !V \mid \mathtt{inl}^{S,T}(V) \mid \mathtt{inr}^{S,T}(V)$$

Figure 1. Syntax of types, terms and values (syntactic sugar: $M\,V$ means $\mathtt{let}\,x = M\,\mathtt{in}\,x\,V$ for fresh x)

$$\frac{}{\Delta \mid \vdash c^S : S} \qquad \frac{\Delta \mid \Gamma, x : S \vdash M : T}{\Delta \mid \Gamma \vdash \lambda x.M : S \multimap T} \qquad \frac{\Delta \mid \Gamma \vdash M : T \qquad \Delta \mid \Gamma \vdash N : T}{\Delta \mid \Gamma \vdash M \diamond N : T}$$
$$\frac{}{\Delta \mid \vdash V : !(S \multimap T) \multimap S \multimap T}{\Delta \mid \vdash \mathbf{Y}\,V : S \multimap T} \qquad \frac{\Delta \mid \Gamma_1 \vdash M : I \qquad \Delta \mid \Gamma_2 \vdash N : T}{\Delta \mid \Gamma_1, \Gamma_2 \vdash M;N : T}$$

Figure 2. Simple typing rules (excerpt)

(a) Classical control flow
$$[\vec{x} = e, E[(\lambda y.M)\,V]] \xrightarrow{0} [\vec{x} = e, E[M\{V/y\}]] \qquad [\vec{x} = e, E[M_1 \diamond M_2]] \xrightarrow{i} [\vec{x} = e, E[M_i]]$$
$$[\vec{x} = e, E[\mathbf{Y}\,V]] \xrightarrow{0} [\vec{x} = e, E[V\,!(\lambda x.\mathbf{Y}\,V\,x)]] \qquad [\vec{x} = e, E[\mathtt{case}\,\mathtt{inl}(V)\,\mathtt{of}\,(x : M \mid y : N)]] \xrightarrow{0} [\vec{x} = e, E[M\{V/x\}]]$$

(b) "Non-classical" data
$$[\vec{x}^{\vec{\mathsf{a}}}\vec{y}^{\vec{\mathsf{b}}} = e, E[c^{\vec{\mathsf{a}} \multimap \vec{\mathsf{a}}'}(\vec{x})]] \xrightarrow{0} [\vec{z}^{\vec{\mathsf{a}}'}\vec{y}^{\vec{\mathsf{b}}} = ((c \otimes \mathrm{id}_{\vec{\mathsf{b}}}) \circ e), E[\vec{z}]] \qquad [x_1 \ldots x_n = e, P] \xrightarrow{\epsilon} [x_{\sigma(1)} \ldots x_{\sigma(n)} = \sigma \circ e, P]$$

Figure 3. Operational semantics (excerpt). Here σ is a permutation $\sigma \in \mathfrak{S}_n$ of n elements, identified with the structural isomorphism $\mathsf{a}_1 \otimes \cdots \otimes \mathsf{a}_n \to \mathsf{a}_{\sigma(1)} \otimes \cdots \otimes \mathsf{a}_{\sigma(n)}$ in $\mathcal{W}$.

the constant corresponding to $1/2 \in \mathcal{W}(I,I)$. A configuration is (essentially) a pair of $p \in [0,1]$ and M, and $[1,M] \xrightarrow{\pi} [p,N]$ means that the probability of this reduction sequence is p. The *probability of convergence of* P can be defined by $\sum_{\pi \in Eval(P)} w(\pi)$. If P is really "probabilistic", i.e. it has only nondeterministic branches of the form $(p;M) \diamond (1-p;N)$, then the sum must converge. Otherwise the sum can be infinite. If we want to ensure that the above sum is always defined, we should replace $[0,1]$ with $\mathbf{R}^{\infty}_{\geq 0} := \{x \in \mathbf{R} \mid 0 \leq x\} \cup \{\infty\}$ (with $0 \times \infty = 0$).

Example 2.3 (Algebraic calculus)**.** The commutative monoid $([0,1],\times)$ in the previous example can be replaced with any other commutative monoid. Indeed a category $\mathcal{W}$ with one object I is an SMCC if and only if $\mathcal{W}(I,I)$ is a commutative monoid. Let R be a commutative monoid and $\mathcal{W}_R$ be a category with one object I and $\mathcal{W}_R(I,I) = R$. In $\lambda_{\mathcal{W}_R}$, one can write a sum of terms with coefficients from R, e.g. $(r;M) \diamond (r';N)$ where $r, r' \in \mathcal{W}_R(I,I) = R$, as in the algebraic lambda calculus [34]. If R has the addition operation (i.e. R is a commutative semiring), one can define the *weight of convergence of* P by $\sum_{\pi \in Eval(P)} w(\pi)$. Here the sum may be undefined since $Eval(P)$ can be a countably infinite set. It is always defined if, for example, R is a *continuous semiring* as in [21].

Example 2.4 (Quantum calculus 1)**.** Let $\mathcal{W} = \mathbf{FdHilb}$ be the category of finite dimensional Hilbert spaces, whose object is a natural number and whose morphism $f : n \to m$ is a complex linear function $f : \mathbf{C}^n \to \mathbf{C}^m$. This is a compact closed category with tensor product $n \otimes m := n \times m$. The quantum lambda calculus of [28] can be embedded into $\lambda_{\mathbf{FdHilb}}$. The calculus $\lambda_{\mathbf{FdHilb}}$ has the atomic type $\mathtt{qubit} := 2$ and every unitary map U on $\mathtt{qubit}^{\otimes n}$ as constants. The creation $\mathtt{new} : I \oplus I \to \mathtt{qubit}$ of new qubit is given by $\mathtt{new} := \lambda x.\mathtt{case}\,x\,\mathtt{of}\,(\mathtt{inl}(y) : (y;|0\rangle) \mid \mathtt{inr}(z) : (z;|1\rangle))$ where $|0\rangle$ and $|1\rangle$ be the standard basis vectors $\begin{pmatrix}1\\0\end{pmatrix}$ and $\begin{pmatrix}0\\1\end{pmatrix}$ of $\mathtt{qubit}$ regarded as morphisms $I \to \mathtt{qubit}$. The measurement $\mathtt{meas} : \mathtt{qubit} \to I \oplus I$ can be defined as the nondeterministic branching followed by projections: $\mathtt{meas} := \lambda x.((\langle 0|\,x);\mathtt{inl}(I)) \diamond ((\langle 1|\,x);\mathtt{inr}(I))$ where $\langle 0| = (1\ 0)$ and $\langle 1| = (0\ 1)$ are projections. A (typical) configuration is $[x_1, \ldots, x_n = e, M]$ where e is a vector in the Hilbert space of dimension 2^n (i.e. the Hilbert space $\mathtt{qubit}^{\otimes n}$); note that e is not normalised but the length indicates the probability of the reduction, that means, the probability of $[\epsilon = 1, P] \xrightarrow{\pi} [\vec{x} = e, Q]$ is $\|e\|^2$. Hence the *probability of convergence of* P is defined as $\sum_{\pi \in Eval(P)} w(\pi) w(\pi)^*$ where $(-)^*$ is the complex conjugate.

This definition of the probability of convergence gives the same value as in [28]. Actually there exists a bijection between the reduction sequences in [28] and those of our calculus, which maps $[e, |\vec{x}\rangle, M] \xrightarrow{p} [e', |\vec{y}\rangle, M']$ in [28] (where e and e' are normalised vectors and p is the probability of this reduction sequence) to $[\vec{x} = e, M] \longrightarrow [\vec{y} = \sqrt{p}e', M']$ in $\lambda_{\mathbf{FdHilb}}$.

Example 2.5 (Quantum calculus 2)**.** Let $\mathcal{W}$ be the category $\mathbf{CPM}_s$ of completely positive maps, whose object is a natural number and whose morphism $g : n \to m$ is a special kind of linear function from $(n \times n)$-matrices to $(m \times m)$-matrices called *completely positive maps* (see, e.g., [30] and [31]). Here we use only the following fact: given a linear function $f : n \to m$, the mapping $A \mapsto fAf^*$ (A an $n \times n$-matrix) is completely positive (where $(-)^*$ is the adjoint operator) and thus a morphism $n \to m$ in $\mathbf{CPM}_s$. This induces a functor $\mathbf{FdHilb} \to \mathbf{CPM}_s$ preserving the compact closed structure , as well as a translation from $\lambda_{\mathbf{FdHilb}}$ to $\lambda_{\mathbf{CPM}_s}$. The quantum calculus of [28] can be embedded into $\lambda_{\mathbf{CPM}_s}$ via this translation. A configuration $[\vec{x} = e, P]$ of $\lambda_{\mathbf{FdHilb}}$ corresponds to $[\vec{x} = ee^*, P]$ of $\lambda_{\mathbf{CPM}_s}$. An advantage of this is that now the *probability of convergence of* P is defined as the standard sum $\sum_{\pi \in Eval(P)} w(\pi)$ in $\mathbf{CPM}_s(I,I) \cong \mathbf{R}_{\geq 0}$. This advantage is significant; see Remark 5.17.

2.4 Categorical interpretation

A *$\lambda_{\mathcal{W}}$-model* is a category equipped with the following structures: (1) a linear-non-linear category [26], (2) finite coproducts $\oplus$, and (3) interpretations of base types and constants, including $\mathbf{Y}$ of each type. It is straightforward to give an interpretation of $\lambda_{\mathcal{W}}$-terms in a $\lambda_{\mathcal{W}}$-model. Note that the fixed-point combinator is treated as a constant and thus there is no guarantee that this interpretation is adequate. Adequacy shall be discussed for individual models.

There is an appropriate notion of *$\lambda_{\mathcal{W}}$-model morphisms*, which strongly preserves the above structure. An important property

is that a $\lambda_{\mathcal{W}}$-model morphism preserves the interpretation of a program (up to the structural isomorphism).

3 Rigid Taylor Expansion

This section reviews a theory of linear approximations of $\lambda_{\mathcal{W}}$-terms, a variant of the Taylor expansion [6] that we call the *rigid Taylor expansion* [32]. The aim of this section is to give a syntactic justification (or understanding) of the definition of weighted profunctors, which is introduced in the next section. Since most results of this section are an adaptation of our previous work [32], we give only a quick overview; see [32] for details.

3.1 Refinement types

We first introduce *refinement intersection types* (or *refinement types* for short), which properly describe classical control flows of a given term. The syntax of refinement types is shown in Fig. 4(a). It parallels the syntax of simple types: each simple-type constructor has one or two corresponding refinement-type constructors. The intuitive "correspondence" of type constructors is formally defined by the *refinement relation*, which is a binary relation $a \lhd S$ between refinement types and simple types. Some rules are listed in Fig. 4(b).

We comment on some notable points. A refinement of the exponential type $!S$ is a list $\langle a_1, \dots, a_n \rangle$ of refinement types a_i of S. This refinement type should be read as a (non-idempotent) intersection type $a_1 \wedge \cdots \wedge a_n$; a value of this type shall be made n copies, used in accord with $a_1, \dots, a_n$ respectively. A refinement of the sum type $S \oplus T$ is either $a \oplus \bullet$ or $\bullet \oplus b$. A value of type $a \oplus \bullet$ must be of the form $\mathtt{inl}(V)$ and a describes the usage of the value V. Note that a refinement type of the value V in a case analysis $\mathtt{case}\, V\, \mathtt{of}\, (\cdots)$ tells us the actual branch.

The refinement types $\langle a_1, a_2 \rangle$ and $\langle a_2, a_1 \rangle$ are different but closely related. They both say that the value of these types shall be duplicated, one copy is used as of type a_1 and the other is as of type a_2. This similarity is captured by the notion of *type isomorphisms*. We write $\varphi : a \cong a'$ to mean that refinements a and a' of S are isomorphic, of which φ is a *witness* (or a *proof*). It is defined by fairly straightforward rules, some of which are found in Fig. 4(c). Note that refinement types are isomorphic in more than one way. For example, consider $\langle \mathsf{a}, \mathsf{a} \rangle$, which is isomorphic to itself in two ways; one relates the left component to the left component, and the other relates the left component to the right component.

For each simple type S, the collection of refinement types of S and isomorphisms between them forms a *groupoid*, which means the existence of the following: (1) identity $\mathrm{id}_a : a \cong a$ for every $a \lhd S$, (2) composite $(\psi \circ \varphi) : a \cong c$ for every $\varphi : a \cong b$ and $\psi : b \cong c$ and (3) inverse $\varphi^{-1} : b \cong a$ for every $\varphi : a \cong b$. We write this groupoid as $[\![S]\!]$.

We say that an isomorphism φ is *positive* if, for each negative (i.e. contravariant) occurrence of $\langle \sigma; \psi_1, \dots, \psi_n \rangle$ in φ, we have $\sigma = \mathrm{id}$. It is *negative* if every permutation in a covariant position is the identity. The groupoid $[\![S]\!]$ has a *strict factorisation system*: positive (resp. negative) isomorphisms form a subcategory and each isomorphism φ can be uniquely decomposed as $\varphi = \varphi^+ \circ \varphi^-$ where φ^+ is a positive isomorphism and φ^- is negative. We write $[\![S]\!]^+$ (resp. $[\![S]\!]^-$) as the positive (resp. negative) subcategory, which is a groupoid.

3.2 Refinement typing rules and its term representation

The syntax of *rigid resource raw-terms* is given in Fig. 5. They are used to represent refinement type derivations. It has basically the same term constructors as $\lambda_{\mathcal{W}}$ but three crucial differences: (1) A rigid resource raw-term has only one branch of nondeterministic choice $M \diamond N$ and case analysis $\mathtt{case}\, V\, \mathtt{of}\, (\mathtt{inl}(x) : M \mid \mathtt{inr}(y) : N)$; (2) A rigid resource raw-term has a list $\langle v_1, \dots, v_n \rangle$ instead of the exponential $!V$; and (3) A rigid resource raw-term has no recursion. Thanks to these changes, rigid resource raw-terms have desirable properties: *a rigid resource raw-term has a unique reduction sequence, which must terminate; and it is linear,* i.e. each variable in a resource term is used exactly once.

We define a set of rules relating resource terms and $\lambda_{\mathcal{W}}$-terms. A *refinement non-linear type environment*, ranged over by Θ, is a finite sequence of type bindings of the from $\langle x_1, \dots, x_n \rangle : \langle a_1, \dots, a_n \rangle$. We write O for refinement non-linear type environments consisting of $\langle\rangle : \langle\rangle$. A *refinement linear type environment*, ranged over by Ξ, is a finite sequence of type bindings of the from $x : a$. The *refinement relations* are defined by the following rules

$$\frac{a_i \lhd S\ (\forall i \le n) \quad \Theta \lhd \Delta}{(\langle x_1, \dots, x_n \rangle : \langle a_1, \dots, a_n \rangle, \Theta) \lhd (y : S, \Delta)} \qquad \frac{a \lhd S \quad \Xi \lhd \Gamma}{(x : a, \Xi) \lhd (y : S, \Gamma)}$$

in addition to a rule relating empty environments. Note that we only compare types but not variable names. We write $(\Theta \mid \Xi) \lhd (\Delta \mid \Gamma)$ if $\Theta \lhd \Delta$ and $\Xi \lhd \Gamma$. A *refinement type judgement* is a tuple $\Theta \lhd \Delta \mid \Xi \lhd \Gamma \vdash t : a \lhd M : S$ with $(\Theta \mid \Xi) \lhd (\Delta \mid \Gamma)$ and $a \lhd S$. We omit $\Xi \lhd \Gamma$ (resp. $\Theta \lhd \Delta \mid \Xi \lhd \Gamma$) if both Ξ and Γ (resp. the four environments) are the empty sequence. Figure 6 shows important rules. Here $\wedge$ is the component-wise concatenation, e.g.,

$$(\langle x_1, x_2 \rangle : \langle a_1, a_2 \rangle,\ \langle y_1 \rangle : \langle b_1 \rangle) \wedge (\langle\rangle : \langle\rangle,\ \langle z_1, z_2 \rangle : \langle c_1, c_2 \rangle) = (\langle x_1, x_2 \rangle : \langle a_1, a_2 \rangle,\ \langle y_1, z_1, z_2 \rangle : \langle b_1, c_1, c_2 \rangle).$$

By dropping some components, the rules can be seen as three different typing systems. First, by removing the left-hand-sides of $\lhd$, the rules are a variant of those of the simple type system of $\lambda_{\mathcal{W}}$. Second, dropping the resource calculus part and the simple type part results in a non-idempotent intersection type system: for example, an instance of the exponential rule is

$$\frac{x : \langle \vec{a}_1 \rangle \mid \vdash V : b_1 \quad \dots \quad x : \langle \vec{a}_n \rangle \mid \vdash V : b_n}{x : \langle \vec{a}_1, \dots, \vec{a}_n \rangle \mid \vdash !V : \langle b_1, \dots, b_n \rangle}.$$

Third, by ignoring the right-hand-sides of $\lhd$, the resulting system can be seen as the standard type system for the linear lambda calculus *without exponentials*; we shall discuss this point in Section 4.2.

Although we do not have the general type isomorphism rule in the type system, it is derivable in a sense. For example, assume $\Theta \lhd \Delta \mid \vdash \langle v_1, \dots, v_n \rangle : \langle a_1, \dots, a_n \rangle \lhd !V : S$ and consider the type isomorphism $\varphi = \langle \sigma; \vec{\mathrm{id}} \rangle : \langle a_1, \dots, a_n \rangle \cong \langle a_{\sigma(1)}, \dots, a_{\sigma(n)} \rangle$, determined by a permutation $\sigma \in \mathfrak{S}_n$. Although we do not have $\Theta \lhd \Delta \mid \vdash \langle v_1, \dots, v_n \rangle : \langle a_{\sigma(1)}, \dots, a_{\sigma(n)} \rangle \lhd !V : S$, by applying the permutation σ to the term as well as the refinement type, we obtain a derivable judgement $\Theta \lhd \Delta \mid \vdash \langle v_{\sigma(1)}, \dots, v_{\sigma(n)} \rangle : \langle a_{\sigma(1)}, \dots, a_{\sigma(n)} \rangle \lhd !V : S$. A generalisation of this idea is the *action of an isomorphism $\varphi : a \cong a'$ to a rigid resource raw-term t*; we write $[\varphi] \cdot t$ for the term obtained by acting φ to t. It is defined by induction on t; examples of rules are

$$[\langle \sigma; \varphi_1, \dots, \varphi_n \rangle] \cdot \langle v_1, \dots, v_n \rangle := \langle [\varphi_1] \cdot v_{\sigma(1)}, \dots, [\varphi_n] \cdot v_{\sigma(n)} \rangle$$
$$[(\varphi \multimap \psi)] \cdot (\lambda x.t) := \lambda x.([\psi] \cdot t)\{[\varphi]x/x\}$$
$$[\varphi] \cdot (v\, w) := ([(\mathrm{id} \multimap \varphi)] \cdot v)\, w.$$

(a) Syntax $\quad a, b ::= \mathsf{a} \mid a \multimap b \mid () \mid a \otimes b \mid \langle a_1, \ldots, a_n \rangle \mid a \oplus \bullet \mid \bullet \oplus a$

(b) Refinement relation
$$\frac{a \lhd S \qquad b \lhd T}{a \multimap b \lhd S \multimap T} \qquad \frac{a \lhd S \qquad b \lhd T}{a \otimes b \lhd S \otimes T} \qquad \frac{a_1 \lhd S \qquad \ldots \qquad a_n \lhd S}{\langle a_1, \ldots, a_n \rangle \lhd\, !S} \qquad \frac{a \lhd S}{a \oplus \bullet \lhd S \oplus T} \qquad \frac{b \lhd T}{\bullet \oplus b \lhd S \oplus T}$$

(c) Type isomorphisms
$$\frac{\varphi : a' \cong a \qquad \psi : b \cong b'}{\varphi \multimap \psi : a \multimap b \cong a' \multimap b'} \qquad \frac{\sigma \in \mathfrak{S}_n \qquad \varphi_1 : a_{\sigma(1)} \cong b_1 \qquad \ldots \qquad \varphi_n : a_{\sigma(n)} \cong b_n}{\langle \sigma; \varphi_1, \ldots, \varphi_n \rangle : \langle a_1, \ldots, a_n \rangle \lhd \langle b_1, \ldots, b_n \rangle} \qquad \frac{\varphi : a \cong a'}{\varphi \oplus \bullet : a \oplus \bullet \cong a' \oplus \bullet}$$

Figure 4. Syntax, refinement relation and isomorphisms of refinement intersection types (excerpt)

$$s, t, u ::= v \mid v\, w \mid t \diamond \bullet \mid \bullet \diamond t \mid s; t \mid \mathtt{let}\, x = s\, \mathtt{in}\, t \mid \mathtt{let}\, \langle x_1, \ldots, x_n \rangle = v\, \mathtt{in}\, t \mid \mathtt{let}\, x \otimes y = s\, \mathtt{in}\, t \mid \mathtt{let}\, \mathtt{inl}(x) = v\, \mathtt{in}\, t \mid \mathtt{let}\, \mathtt{inr}(x) = v\, \mathtt{in}\, t$$
$$v, w ::= [\varphi]x \mid c \mid \lambda x^a.t \mid () \mid v \otimes w \mid \langle v_1, \ldots, v_n \rangle \mid \mathtt{inl}(v) \mid \mathtt{inr}(v).$$

Figure 5. Syntax of rigid resource raw-terms

$$\frac{\varphi : a \cong a' \qquad O_1 \lhd \Delta_1 \quad a \lhd S \quad O_2 \lhd \Delta_2}{(O_1, \langle x \rangle : \langle a \rangle, O_2) \lhd (\Delta_1, y : S, \Delta_2) \mid\ \vdash [\varphi]x : a' \lhd y : S} \qquad \frac{\varphi : a \cong a' \qquad O \lhd \Delta \quad a \lhd S}{O \lhd \Delta \mid (x : a) \lhd (y : S) \vdash [\varphi]x : a' \lhd y : S} \qquad \frac{O \lhd \Delta}{O \lhd \Delta \mid\ \vdash c^S : S \lhd c^S : S}$$

$$\frac{\Theta \lhd \Delta \mid \Xi \lhd \Gamma \vdash t : a \lhd M : S}{\Theta \lhd \Delta \mid \Xi \lhd \Gamma \vdash t \diamond \bullet : a \lhd M \diamond N : S} \qquad \frac{\Theta_i \lhd \Delta \mid\ \vdash v_i : a_i \lhd V : S \quad (\forall i \le n)}{(\Theta_1 \wedge \cdots \wedge \Theta_n) \lhd \Delta \mid\ \vdash \langle v_1, \ldots, v_n \rangle : \langle a_1, \ldots, a_n \rangle \lhd\, !V :\, !S}$$

$$\frac{\Theta_0 \lhd \Delta \mid\ \vdash v : \langle b_1, \ldots, b_n \rangle \multimap a \lhd V :\, !T \multimap T \qquad \Theta_i \lhd \Delta \mid\ \vdash w_i : b_i \lhd \lambda x.\mathbf{Y}\, V\, x : T \quad (1 \le {}^{\forall} i \le n)}{(\Theta_0 \wedge \cdots \wedge \Theta_n) \lhd \Delta \mid\ \vdash ((); (v\, \langle w_1, \ldots, w_n \rangle)) : a \lhd \mathbf{Y}\, V : T}$$

$$\frac{\Theta_1 \lhd \Delta \mid \Xi_1 \lhd \Gamma_1 \vdash v : \mathtt{inl}(a) \lhd V : S \oplus T \qquad \Theta_2 \lhd \Delta \mid (\Xi_2, y : a) \lhd (\Gamma_2, z : T) \vdash t : c \lhd M : U}{(\Theta_1 \wedge \Theta_2) \lhd \Delta \mid (\Xi_1, \Xi_2) \lhd (\Gamma_1, \Gamma_2) \vdash \mathtt{let}\, \mathtt{inl}(y) = v\, \mathtt{in}\, t : c \lhd \mathtt{case}\, V\, \mathtt{of}\, (\mathtt{inl}(z) : M \mid \mathtt{inr}(z') : N) : U}$$

Figure 6. Rules relating rigid resource raw-terms and $\lambda_{\mathcal{W}}$-terms (excerpt)

The *substitution* $t\{v/x\}$ is defined as usual except for the base case where $([\varphi]x)\{v/x\} := [\varphi] \cdot v$.

Lemma 3.1. *The type isomorphism rules are derivable, e.g.,*

$$\frac{\Theta \lhd \Delta \mid \Xi \lhd \Gamma \vdash t : a \lhd M : S \qquad \varphi : a \cong a'}{\Theta \lhd \Delta \mid \Xi \lhd \Gamma \vdash [\varphi] \cdot t : a' \lhd M : S} \quad \textit{and}$$
$$\frac{\varphi : a \cong a' \qquad \Theta \lhd \Delta \mid (\Xi, x : a') \lhd \Gamma \vdash t : b \lhd M : S}{\Theta \lhd \Delta \mid (\Xi, x : a) \lhd \Gamma \vdash t\{[\varphi]x/x\} : b \lhd M : S}.$$

Let us define an isomorphism $\varphi : (\Theta \mid \Xi) \cong (\Theta' \mid \Xi')$ as a sequence of component-wise isomorphisms. The previous lemma can be generalised to

$$\frac{\varphi : (\Theta' \mid \Xi') \cong (\Theta \mid \Xi) \qquad \Theta \lhd \Delta \mid \Xi \lhd \Gamma \vdash t : b \lhd M : S}{\Theta \lhd \Delta \mid \Xi \lhd \Gamma \vdash t\{\varphi\} : b \lhd M : S}$$

where $t\{\varphi\}$ denotes the appropriate resource raw-term.

3.3 Enumeration of reduction sequences

The operational semantics of the rigid calculus is defined analogously to that of $\lambda_{\mathcal{W}}$. A *configuration* is a triple $[\vec{x} = e, t]$ that is well-typed in an appropriate sense, and the reduction is a relation on configurations. Examples of rules are

$$[\vec{x} = e, E[t \diamond \bullet]] \xrightarrow{1} [\vec{x} = e, E[t]]$$
$$[\vec{x} = e, E[\mathtt{let}\, \langle y_1, \ldots, y_n \rangle = \langle v_1, \ldots, v_n \rangle\, \mathtt{in}\, t]] \xrightarrow{0} [\vec{x} = e, E[t\{v_1/y_1, \ldots, v_n/y_n\}]]$$

where E is an *evaluation context*, defined by the grammar: $E ::= [\,] \mid E; t \mid \mathtt{let}\, x = E\, \mathtt{in}\, t$. Problematic configurations such as $[\vec{x} = e, \mathtt{let}\, \mathtt{inr}(x) = \mathtt{inl}(v)\, \mathtt{in}\, t]$ are filtered out by the type system.

The next lemma follows from the fact that a rigid resource raw-term has neither nondeterministic branching nor recursion.

Lemma 3.2. *For each configuration $[\vec{x} = e, t]$, there exists a unique pair (π, e') such that $[\vec{x} = e, t] \xrightarrow{\pi} [\epsilon = e', ()]$.*

For a program $\vdash P : I$, let us consider the set $\{t \mid \vdash t() \lhd P : I\}$. For each element $t \in X$ of this set, we write $\pi(t)$ and $\varpi(t)$ to mean the unique π and e such that $[\epsilon = \mathrm{id}_I, t] \xrightarrow{\pi} [\epsilon = e, ()]$. The next theorem says that the mapping $t \mapsto \pi(t)$ is a weight-preserving surjection to $\mathit{Eval}(P)$.

Theorem 3.3. *Let P be a program. If $\vdash t : () \lhd P : I$ and $[\epsilon = \mathrm{id}_I, t] \xrightarrow{\pi} [\epsilon = e, ()]$, then $[\epsilon = \mathrm{id}_I, P] \xrightarrow{\pi} [\epsilon = e, ()]$. Conversely, if $[\epsilon = \mathrm{id}_I, P] \xrightarrow{\pi} [\epsilon = e, ()]$, then there exists t such that $\vdash t : () \lhd P : I$ and $[\epsilon = \mathrm{id}_I, t] \xrightarrow{\pi} [\epsilon = e, ()]$.*

Unfortunately this is not a bijection: different approximants may induce the same computation. For example, consider refinements

$$\mathtt{let}\, \langle x_1, x_2 \rangle = \langle v_1, v_2 \rangle\, \mathtt{in}\, x_1 \otimes x_2 \qquad \mathtt{let}\, \langle x_2, x_1 \rangle = \langle v_2, v_1 \rangle\, \mathtt{in}\, x_1 \otimes x_2$$

of $\mathtt{let}\, !x =\, !V\, \mathtt{in}\, x \otimes x$ (see [32] for further discussion).

We have proposed in our previous work [32] a way to avoid this redundancy by using the action of isomorphisms. Let $\sim$ be a congruence on rigid resource raw-terms subsuming

$$v\, ([\varphi] \cdot w) \ \sim\ ([(\varphi \multimap \mathrm{id})] \cdot v)\, w$$
$$\mathtt{let}\, x = [\varphi] \cdot t\, \mathtt{in}\, u \ \sim\ \mathtt{let}\, x = t\, \mathtt{in}\, (u\{[\varphi]x/x\})$$
$$\mathtt{let}\, \langle x_1, \ldots, x_n \rangle = ([\langle \sigma; \varphi_1, \ldots, \varphi_n \rangle] \cdot v)\, \mathtt{in}\, t$$
$$\sim\ \mathtt{let}\, \langle x_{\sigma^{-1}(1)}, \ldots, x_{\sigma^{-1}(n)} \rangle = v\, \mathtt{in}\, t\{[\varphi_1]x_1/x_1, \ldots, [\varphi_n]x_n/x_n\}$$

and similar rules for other let-constructs. Note that $\sim$ is defined for terms of higher-order types as well.

Theorem 3.4 ([32])**.** *Let P be a program and assume $\vdash t_i : () \lhd P : I$ for $i = 1, 2$. Then $t_1 \sim t_2$ if and only if $\pi(t_1) = \pi(t_2)$.*

Given a $\lambda_{\mathcal{W}}$-term, its *rigid Taylor expansion* is defined as the collection of well-typed approximations of it. We write $\widetilde{t}$ for the equivalence class of $\sim$ to which t belongs.

Definition 3.5 (Rigid Taylor expansion)**.** Given $\Delta \mid \Gamma \vdash M : S$ and $(\Theta \mid \Xi) \lhd (\Delta \mid \Gamma)$, we define

$$[\![M]\!](b, (\Theta \mid \Xi)) := \{\, \widetilde{t} \mid \Theta \lhd \Delta \mid \Xi \lhd \Gamma \vdash t : b \lhd M : S \,\}$$

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

Theorems 3.3 and 3.4 give a bijective correspondence between $\mathit{Eval}(P)$ and $[\![P]\!](\epsilon, ())$, which furthermore preserves the weights. This allows us to enumerate $\mathit{Eval}(P)$ by induction on the structure of P, even though $\mathit{Eval}(P)$ is not inductively defined.

4 Weighted Generalised Species

We have seen in the previous section that the rigid Taylor expansion of a program P is a weighted set equivalent to $\mathit{Eval}(P)$ up to a weight-preserving bijection, and hence in a sense adequate. This section gives a more "semantic" description of this result, based on *weighted generalised species* (or *weighted profunctors*). The result of this section extends [32], which studies a weight-free setting.

4.1 Preliminary: profunctors

We briefly recall profunctors and introduce notations. A *profunctor* F from a category $\mathcal{A}$ to a category $\mathcal{B}$ (written $F : \mathcal{A} \nrightarrow \mathcal{B}$) is a functor $F : \mathcal{B}^{\mathrm{op}} \times \mathcal{A} \to \mathbf{Set}$. For $g \in \mathcal{B}(b', b)$, $x \in F(b, a)$ and $f \in \mathcal{A}(a, a')$, we write $x \cdot f$ for $F(b, f)(x)$ and $g \cdot x$ for $F(g, a)(x)$. Since F is a bifunctor, $(g \cdot x) \cdot f = g \cdot (x \cdot f)$, which we simply write as $g \cdot x \cdot f$.[1] The composite $G \circ F : \mathcal{A} \nrightarrow \mathcal{C}$ of $F : \mathcal{A} \nrightarrow \mathcal{B}$ and $G : \mathcal{B} \nrightarrow \mathcal{C}$ can be defined by

$$(G \circ F)(c, a) := \Big(\coprod_{b \in \mathcal{B}} G(c, b) \times F(b, a) \Big) / \sim$$

where $\coprod$ is the coproduct in $\mathbf{Set}$ and $\sim$ is the least equivalence relation containing $(y, f \cdot x) \sim (y \cdot f, x)$ for each $y \in G(c, b')$, $f \in \mathcal{B}(b', b)$ and $x \in F(b, a)$. We write $\mathbf{Prof}$ for the bicategory of categories, profunctors and natural transformations.

4.2 Properties of the rigid Taylor expansion

This subsection studies the properties of the rigid Taylor expansion, which shall be abstracted to the notion of weighted profunctors.

As pointed out in our previous work [32], the rigid Taylor expansion $[\![M]\!]$ of a term $\Delta \mid \Gamma \vdash M : S$ is a profunctor $[\![\Delta \mid \Gamma]\!] \nrightarrow [\![S]\!]$. Here $[\![\Delta \mid \Gamma]\!]$ and $[\![S]\!]$ are groupoids of refinements and isomorphisms. Lemma 3.1 shows that $\varphi : a' \cong a$, $\widetilde{t} \in [\![M]\!](a, (\Theta \mid \Xi))$ and $\psi : (\Theta \mid \Xi) \cong (\Theta' \mid \Xi')$ imply $\widetilde{[\varphi^{-1}] \cdot t\{\psi^{-1}\}} \in [\![M]\!](a', (\Theta' \mid \Xi'))$.

In the situation of this paper, one can furthermore interpret the refinement types and rigid resource (raw-)terms in $\mathcal{W}$.

The interpretation of a simple type induces a functor $S : [\![S]\!] \to \mathcal{W}^{\mathrm{op}}$, i.e. refinement types and type isomorphisms can be seen as objects and morphisms in $\mathcal{W}$, respectively. Its action on objects are defined via the following syntactic translation

$$\natural(\mathsf{a}) = \mathsf{a} \qquad \natural(a \otimes b) = \natural(a) \otimes \natural(b)$$
$$\natural(()) = I \qquad \natural(\mathtt{inl}(a)) = \natural(\mathtt{inr}(a)) = \natural(a)$$
$$\natural(a \multimap b) = \natural(a) \multimap \natural(b) \qquad \natural(\langle a_1, \ldots, a_n \rangle) = \natural(a_1) \otimes \cdots \otimes \natural(a_n)$$

[1] In this paper, the action of profunctors is written in the diagrammatic order in the sense that $g' \cdot (g \cdot x \cdot f) \cdot f' = (g'; g) \cdot x \cdot (f; f')$, where $g'; g \triangleq g \circ g'$.

of a refinement type to an IMLL formula. Its action on morphisms is defined by induction on the derivation of $\varphi : a \cong a'$, using only the structural isomorphisms in $\mathcal{W}$.

A rigid resource (raw-)term induces a term of a linear lambda calculus without exponential, by ignoring $\mathtt{inl}$ and $\mathtt{inr}$ and identifying $\langle v_1, \ldots, v_n \rangle$ with $v_1 \otimes \cdots \otimes v_n$ as well as the corresponding patterns. For example, $\mathtt{let}\, \langle x, y \rangle = v \,\mathtt{in}\, t$ is regarded as $\mathtt{let}\, x \otimes y = v \,\mathtt{in}\, t$. Thus we have an interpretation of rigid resource (raw-)terms in the SMCC $\mathcal{W}$; we write $(\!|t|\!)$ for this interpretation.

Lemma 4.1. *Let $\Delta \mid \Gamma \vdash M : S$. (1) The simple type S induces a functor $S : [\![S]\!] \to \mathcal{W}^{\mathrm{op}}$ from the groupoid of refinement types and isomorphisms. Similarly the simple type environment induces a functor $E : [\![(\Delta \mid \Gamma)]\!] \to \mathcal{W}^{\mathrm{op}}$. (2) The rigid Taylor expansion is a profunctor $[\![M]\!] : [\![\Delta \mid \Gamma]\!] \nrightarrow [\![S]\!]$ of which each element $\widetilde{t} \in [\![M]\!](a, (\Theta \mid \Xi))$ is associated with a morphism $(\!|t|\!) : E(\Theta \mid \Xi) \to S(a)$ in $\mathcal{W}$. Furthermore $(\!|\cdot|\!)$ respects the action of maps in $[\![\Delta \mid \Gamma]\!]$ and $[\![S]\!]$, i.e. $S(\varphi) \circ (\!|t|\!) \circ E(\psi) = (\!|[\varphi^{-1}] \cdot t\{\psi^{-1}\}|\!)$.*

The above syntactic translation of terms maps the reduction rules to valid equations of the standard linear lambda calculus, by regarding $[\vec{x} = e, t]$ as $\mathtt{let}\, \vec{x} = e \,\mathtt{in}\, t$. Thanks to the well-known soundness result of SMCCs for the linear lambda calculus, $(\!|\cdot|\!)$ is preserved by reduction. Hence the weight of a rigid resource (row-)term coincides with the interpretation in $\mathcal{W}$.

Theorem 4.2. *If $[\epsilon = \mathrm{id}_I, t] \xrightarrow{\pi} [\epsilon = e, ()]$, then $e = (\!|t|\!)$.*

This theorem together with Theorems 3.3 and 3.4 provides us with a compositional way for calculating the weighted set $\mathit{Eval}(P)$.

4.3 Weighted profunctors

We introduce the notion of *weighted profunctors* as an abstraction of the properties shown in Lemma 4.1.

Definition 4.3 (Weighted category, weighted profunctor)**.** A *$\mathcal{W}$-weighted category* is a pair $(\mathcal{A}, A)$ of a category $\mathcal{A}$ and a functor $A : \mathcal{A} \to \mathcal{W}^{\mathrm{op}}$. A *$\mathcal{W}$-weighted profunctor* from $(\mathcal{A}, A)$ to $(\mathcal{B}, B)$ is a pair (F, ϖ) of a profunctor $F : \mathcal{A} \nrightarrow \mathcal{B}$ (i.e. a functor $F : \mathcal{B}^{\mathrm{op}} \times \mathcal{A} \to \mathbf{Set}$) and a family of functions $\varpi_{(b,a)} : F(b, a) \to \mathcal{W}(A(a), B(b))$ $(a \in \mathcal{A}, b \in \mathcal{B})$ that respects the action of $\mathcal{A}$ and $\mathcal{B}$, i.e.,

$$B(g) \circ \varpi_{(b,a)}(e) \circ A(f) \;=\; \varpi_{(b',a')}(g \cdot e \cdot f)$$

for every $g : b' \to b$, $e \in F(b, a)$ and $f : a \to a'$. A *2-cell* $\alpha : (F, \varpi^F) \Rightarrow (G, \varpi^G)$ is a natural transformation $\alpha : F \Rightarrow G$ preserving weights, i.e. $\varpi^F_{(b,a)}(e) = \varpi^G_{(b,a)}(\alpha_{b,a}(e))$ for every $e \in F(b, a)$. We often omit "$\mathcal{W}$-" if it is clear from the context.

Weighted categories, weighted profunctors and 2-cells in Definition 4.3 can be organised into a bicategory, which we write as $\mathbf{Prof}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}}$: The composite $(G, \varpi^G) \circ (F, \varpi^F)$ of weighted profunctors $(F, \varpi^F) : (\mathcal{A}, A) \nrightarrow (\mathcal{B}, B)$ and $(G, \varpi^G) : (\mathcal{B}, B) \nrightarrow (\mathcal{C}, C)$ consists of the composite profunctor $G \circ F$ with the weight function $\varpi_{(c,a)} : (G \circ F)(c, a) \to \mathcal{W}(A(a), C(c))$ defined by

$$\varpi_{c,a}([(y, x)]_\sim) := \varpi^G_{c,b}(y) \circ \varpi^F_{b,a}(x)$$

where $(y, x) \in G(c, b) \times F(b, a)$. This is well-defined since ϖ^G and ϖ^F respect the action of $\mathcal{B}$ morphisms.

We shall mainly use a 1-categorical version of the bicategory $\mathbf{Prof}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}}$, written $\mathbf{Pr}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}}$. This is defined as the *classifying category* $\mathit{Cl}(\mathbf{Prof}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}})$ [2, Section 7] of $\mathbf{Prof}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}}$, whose object is a

0-cell of $\mathbf{Prof}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}}$ and whose morphism is an equivalence class of 1-cells of $\mathbf{Prof}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}}$ modulo the existence of an iso-2-cell.

4.4 $\mathbf{Pr}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}}$ as a $\lambda_{\mathcal{W}}$-model

We first discuss the Lafont structure of $\mathbf{Pr}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}}$. For space reasons, we only give an overview.

The SMCC structure of $\mathbf{Pr}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}}$ follows from the SMCC structure of $\mathbf{Prof}$ and $\mathcal{W}$. Let $A : \mathcal{A} \to \mathcal{W}^{\mathrm{op}}$ and $B : \mathcal{B} \to \mathcal{W}^{\mathrm{op}}$ be weighted categories. The tensor product is defined by $(\mathcal{A}, A) \mathbin{\hat{\otimes}} (\mathcal{B}, B) \triangleq (\mathcal{A} \times \mathcal{B}, A \mathbin{\hat{\otimes}} B)$ where $A \mathbin{\hat{\otimes}} B \triangleq (\otimes^{\mathrm{op}}) \circ (A \times B)$, i.e. $(A \mathbin{\hat{\otimes}} B)(a, b) = A(a) \otimes B(b)$ and $(A \mathbin{\hat{\otimes}} B)(f, g) = A(f) \otimes B(g)$. This definition uses the tensor products $\times$ and $\otimes$ of $\mathbf{Prof}$ and $\mathcal{W}$, respectively. Its action on morphisms $(F_i, \varpi_i) : (\mathcal{A}_i, A_i) \nrightarrow (\mathcal{B}_i, B_i)$ $(i = 1, 2)$ is given by $(F_1 \hat{\otimes} F_2)((b_1, b_2), (a_1, a_2)) \triangleq F_1(b_1, a_1) \times F_2(b_2, a_2)$ with the weight function $F_1(b_1, a_1) \times F_2(b_2, a_2) \ni (x_1, x_2) \mapsto \varpi_1(x_1) \otimes \varpi_2(x_2) \in \mathcal{W}(A_1(a_1) \otimes A_2(a_2), B_1(b_1) \otimes B_2(b_2))$. The closed structure is defined similarly: $(\mathcal{A}, A) \mathbin{\hat{\multimap}} (\mathcal{B}, B) \triangleq (\mathcal{A}^{\mathrm{op}} \times \mathcal{B}, (\multimap^{\mathrm{op}}) \circ (A^{\mathrm{op}} \times B))$.

For any category $\mathcal{W}$, the category $\mathbf{Pr}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}}$ has (small) biproducts given by the biproduct of $\mathbf{Prof}$.

We can show that $\mathbf{Pr}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}}$ has free commutative comonoids, and thus a linear exponential comonad, following the recipe of [21, 27]. It suffices to show that $\mathbf{Pr}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}}$ has *symmetric tensor powers* [27], i.e. the equaliser $\mathbb{P}_n(\mathcal{A}, A) \to (\mathcal{A}, A)$ of $n!$ symmetries from $(\mathcal{A}, A)^{\hat{\otimes} n}$ to itself, and show that the equaliser is preserved by the tensor product. The underlying category of $\mathbb{P}_n(\mathcal{A}, A)$ has as an object a sequence $(a_i)_{i \le n}$ of objects of $\mathcal{A}$ and as a morphism $(a_i)_i \to (a'_i)_i$ a pair of permutation σ and $(f_i : a_i \to a'_{\sigma(i)})_{i \le n}$.

Theorem 4.4. $\mathbf{Pr}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}}$ *is a Lafont category with biproducts.*

Remark 4.5. In the proof of Theorem 4.4, we employ an equivalent but categorically simpler definition of the bicategory $\mathbf{Prof}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}}$, as a full sub-bicategory of the *lax-slice* bicategory of $\mathbf{Prof}$ over $\mathcal{W}^{\mathrm{op}}$. There we prove that $\mathbf{Prof}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}}$ has the symmetric monoidal closed structure, biproducts and symmetric tensor powers in 2-dimensional category theory; hence if we can extend the construction of Lafont categories in [21, 27] to 2-dimensional category theory, we obtain a Lafont *bi*category. □

The interpretation of base type a is a functor $\star \mapsto \mathsf{a} : 1 \to \mathcal{W}^{\mathrm{op}}$. Therefore the interpretation of type $\mathsf{a}_1 \otimes \cdots \otimes \mathsf{a}_n$ is $\star \mapsto \mathsf{a}_1 \otimes \cdots \otimes \mathsf{a}_n : 1 \to \mathcal{W}^{\mathrm{op}}$. The interpretation of constant $c^{\mathsf{a}_1 \otimes \cdots \otimes \mathsf{a}_n \multimap \mathsf{b}_1 \otimes \cdots \otimes \mathsf{b}_m}$ consists of the profunctor $F(\star, \star) := \{*\}$ with the weight function $* \mapsto c \in \mathcal{W}(\mathsf{a}_1 \otimes \cdots \otimes \mathsf{a}_n, \mathsf{b}_1 \otimes \cdots \otimes \mathsf{b}_m)$. The interpretation of Y is defined as the rigid Taylor expansion of $x : !T \multimap T \vdash \mathsf{Y}\, x : T$, in order to establish Theorem 4.6. We expect this to coincide with the fixed-point operator of Laird's theorem [19, Thm. 4.20], though a proper comparison is left for future work.

The concrete definition of the $\lambda_{\mathcal{W}}$-model structure of $\mathbf{Pr}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}}$ is tightly related to the rigid Taylor expansion. For example, for $\mid \Gamma_i \vdash V_i : S_i$ $(i = 1, 2)$, it is fairly easy to see that $[\![V_1]\!] \mathbin{\hat{\otimes}} [\![V_2]\!]$ defines the same 1-cell of $\mathbf{Pr}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}}$ as $[\![V_1 \otimes V_2]\!]$. A notable point is that the equivalence relation $\sim$ in the definition of the composition of profunctors (Section 4.1) coincides with the relation $\sim$ on rigid resource raw-terms (Section 3.3). Hence it is also easy to show that $[\![N]\!] \circ [\![M]\!] = [\![\mathtt{let}\, x = M\, \mathtt{in}\, N]\!]$ for $\mid \Gamma \vdash M : S$ and $\mid x : S \vdash N : T$.

Theorem 4.6. *The interpretation of a term M in $\mathbf{Pr}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}}$ coincides with the rigid Taylor expansion $[\![M]\!]$.*

Corollary 4.7 (Adequacy)**.** *The interpretation of a program P in $\mathbf{Pr}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}}$ coincides with the weighted set $Eval(P)$.*

5 Associated Matrix as Generating Series

This section introduces a concise representation for (a subclass of) weighted profunctors, inspired by the generating series of a weighted species (see e.g. [4]). Recall that the (exponential) generating series of a weighted species $(F : \mathbf{P} \to \mathbf{Set}, \{\varpi_n : F(n) \to W\}_n)$ is defined as $\|(F, \varpi)\| = \sum_{n=0}^{\infty} \|(F, \varpi)\|_n \, z^n$, where z is the indeterminant and the coefficient $\|(F, \varpi)\|_n$ is defined by

$$\|(F, \varpi)\|_n \triangleq \frac{1}{n!} \sum_{x \in F(n)} \varpi_n(x)$$

provided that this expression makes sense (e.g. $W \supseteq \mathbf{Q}$ is a ring and $F(n)$ is finite for every n).

Since a profunctor $F : \mathcal{A} \nrightarrow \mathcal{B}$ is a functor $\mathcal{B}^{\mathrm{op}} \times \mathcal{A} \to \mathbf{Set}$, the ordinary species is a special case of $\mathcal{A} = [\![!I]\!]$ and $\mathcal{B} = [\![I]\!]$ as observed in [9]. This motivates us to define

$$\|(F, \varpi)\|_{b,a} := \frac{1}{\#\mathcal{B}^-(b, b)\, \#\mathcal{A}^+(a, a)} \sum_{x \in F(b,a)} \varpi_{b,a}(x) \qquad (1)$$

($\#X$ is cardinality of the set X; $\mathcal{A}$ and $\mathcal{B}$ will be strict factorisation systems, and the superscripts $(-, +)$ refer resp. to the two classes (E, M) of morphisms). We call $\|(F, \varpi)\|$ the *associated matrix*, as it can be seen as a matrix indexed by $\mathrm{ob}(\mathcal{B})$ and $\mathrm{ob}(\mathcal{A})$, whose elements are morphisms of $\mathcal{W}$. A remarkable difference from the ordinary matrix is that the domains of elements vary with indexes.

The weight category $\mathcal{W}$ should have additional structures for Equation (1) to make sense. In particular, each hom-set $\mathcal{W}(A(a), \mathcal{B}(b))$, to which $\varpi_{b,a}(x)$ belongs, should have the summation operation $\sum$, as well as the multiplication with $1/(\#\mathcal{B}^-(b, b)\, \#\mathcal{A}^+(a, a))$. Section 5.1 defines the requirements of $\mathcal{W}$ in terms of enrichment.

Section 5.2 defines the category of matrices with elements from $\mathcal{W}$ and gives a formal definition of $\|\cdot\|$. Unfortunately $\|\cdot\|$ is not even functorial. Section 5.3 introduces a subclass of profunctors, called *P-visible* profunctors, on which $\|\cdot\|$ behaves well.

5.1 Σ-monoids and ΣMon-categories

Since $Eval(P)$ can be countably infinite, the sum in (1) can also be countably infinite. This subsection introduces an algebra with countable sum, known as Σ-*monoids* [11, 12, 15], and the notion of SMCCs whose hom-sets are Σ-monoids.

Let e and e' be expressions possibly having partial operations. We write $e \sqsubseteq e'$ to mean that, if e is defined, then e' is also defined and the values are the same; $e \eqsim e'$ is a shorthand for $e \sqsubseteq e' \wedge e' \sqsubseteq e$.

Let X be a set. A *countable family in X* is a pair (I, x) of a countable set I of indexes and a function $x : I \to X$. We write x_i for $x(i)$ and $\{x_i\}_{i \in I}$ for a countable family. Countable families $\{x_i\}_{i \in I}$ and $\{y_j\}_{j \in J}$ are *equivalent* if there exists a bijection $f : I \to J$ such that $x_i = y_{f(i)}$ for every $i \in I$. Given a set X, let $Fam(X)$ be the set of all countable families $\{x_i\}_{i \in I}$ in X indexed by a subset of natural numbers (i.e. $I \subseteq \mathbb{N}$).

Definition 5.1 (Σ-monoids)**.** A pair $(\mathfrak{M}, \sum)$ of a nonempty set $\mathfrak{M}$ and a partial function $\sum : Fam(\mathfrak{M}) \to \mathfrak{M}$ is a Σ-*monoid* if it satisfies the following conditions: (1) for every $I, J \subseteq \mathbb{N}$ and partition $\{I_j\}_{j \in J}$ of I, we have $\sum\{x_i\}_{i \in I} \eqsim \sum\{\sum_{i \in I_j} x_i\}_{j \in J}$, and (2) for a singleton $I = \{j\}$, we have $\sum\{x_i\}_{i \in I} \eqsim x_j$. We say $\{x_i\}_{i \in I}$ is *summable* if $\sum\{x_i\}_{i \in I}$ is defined. A Σ-monoid is *total* (aka *complete*)

if all countable families are summable. We often write $\sum_{i\in I} x_i$ for $\sum\{x_i\}_{i\in I}$. A total Σ-monoid is a commutative monoid in the usual sense, with binary sum $x_1 + x_2 := \sum_{i\in\{1,2\}} x_i$.

Example 5.2. Recall examples in Section 2.3. The two-valued Boolean algebra $\mathbf{B}(I,I)$ in Example 2.1 is a total Σ-monoid by disjunction. Both $[0,1]$ and $\mathbf{R}^{\infty}_{\geq 0}$ in Example 2.2 are Σ-monoids by the standard sum of reals (in $\mathbf{R}^{\infty}_{\geq 0}$, $\sum_{i\in I} x_i = \infty$ if it does not converge). The latter is total though the former is not. *Continuous semirings* used in [21] and *(countably) complete semirings* used in [19] are examples of total Σ-monoids by summation. As for Examples 2.4 and 2.5, both $\mathbf{FdHilb}(n,m)$ and $\mathbf{CPM}_s(n,m)$ are non-total Σ-monoids.

Definition 5.3 (Category $\Sigma\mathbf{Mon}$)**.** A *homomorphism of Σ-monoids* is a function $f : \mathfrak{M} \to \mathfrak{N}$ such that $f(\sum_{i\in I} x_i) \sqsubseteq \sum_{i\in I} f(x_i)$ for every $\{x_i\}_{i\in I} \in \mathit{Fam}(\mathfrak{M})$. The *category* $\Sigma\mathbf{Mon}$ has Σ-monoids as objects and homomorphisms of Σ-monoids as morphisms. We write $\Sigma\mathbf{Mon}_t$ for the full subcategory of total Σ-monoids.

We review the structure of $\Sigma\mathbf{Mon}$ and $\Sigma\mathbf{Mon}_t$ following [15].

Definition 5.4 (Bilinear map)**.** Let $\mathfrak{M}$, $\mathfrak{N}$ and $\mathfrak{L}$ be Σ-monoids. A *bilinear map* $f \in \mathrm{Bilin}(\mathfrak{M},\mathfrak{N};\mathfrak{L})$ is a function $\mathfrak{M}\times\mathfrak{N} \to \mathfrak{L}$ such that

$$f(\sum_{i\in I} x_i, y) \sqsubseteq \sum_{i\in I} f(x_i, y) \quad \text{and} \quad f(x, \sum_{i\in I} y_i) \sqsubseteq \sum_{i\in I} f(x, y_i).$$

The functor $\mathrm{Bilin}(\mathfrak{M},\mathfrak{N};-) : \Sigma\mathbf{Mon} \to \mathbf{Set}$ is representable [15, Proposition 3.5]; we write $\mathfrak{M}\otimes\mathfrak{N}$ for the representation, and identify $\Sigma\mathbf{Mon}(\mathfrak{M}\otimes\mathfrak{N},\mathfrak{L})$ with $\mathrm{Bilin}(\mathfrak{M},\mathfrak{N};\mathfrak{L})$.

The category $\Sigma\mathbf{Mon}$ is an SMCC with $\otimes$ as the monoidal product. The unit is $I = \{0,1\}$ with $1+1$ undefined. We have $\Sigma\mathbf{Mon}(I,\mathfrak{M}) \cong \mathfrak{M}$ as sets. The linear function space $\mathfrak{M} \multimap \mathfrak{N}$ is the set of homomorphisms with the sum defined by the point-wise sum.

Definition 5.5 ($\Sigma\mathbf{Mon}$-category, $\Sigma\mathbf{Mon}$-SMCC)**.** A $\Sigma\mathbf{Mon}$*-category* is a locally small category $\mathcal{W}$ such that (1) each hom-set $\mathcal{W}(a,b)$ is equipped with a Σ-monoid structure, and (2) the composition is bilinear. A $\Sigma\mathbf{Mon}$-category is a $\Sigma\mathbf{Mon}$*-SMCC* if (1) the underlying category $\mathcal{W}$ is an SMCC, (2) the action of the tensor product on morphisms, $(f,g) \mapsto (f\otimes g)$, is bilinear, and (3) the bijections $\mathcal{W}(a\otimes b, c) \cong \mathcal{W}(a, b\multimap c)$ are homomorphisms of Σ-monoid. A $\Sigma\mathbf{Mon}_t$*-SMCC* is a $\Sigma\mathbf{Mon}$-SMCC $\mathcal{W}$ all of whose hom-objects $\mathcal{W}(a,b)$ are total Σ-monoids.

Example 5.6. The category $\mathbf{B}(I,I)$ in Example 2.1 is a $\Sigma\mathbf{Mon}_t$-SMCC. The category $\mathcal{W}_{[0,1]}$ and its variant $\mathcal{W}_{\mathbf{R}^{\infty}_{\geq 0}}$ in Example 2.2 are $\Sigma\mathbf{Mon}$-SMCCs; the latter is also an example of $\Sigma\mathbf{Mon}_t$-SMCC. In general, one-object $\Sigma\mathbf{Mon}_t$-SMCCs coincide with *(countably) complete semirings* in the sense of [19, Definition 2.5]. A continuous semiring used in [21] is an example of total Σ-monoid by summation. $\mathbf{FdHilb}$ and $\mathbf{CPM}_s$ are $\Sigma\mathbf{Mon}$-SMCCs but not $\Sigma\mathbf{Mon}_t$-SMCCs.

Definition 5.7 (Reciprocal for natural numbers)**.** Let $\mathcal{W}$ be a $\Sigma\mathbf{Mon}$-category and $a \in \mathrm{ob}(\mathcal{W})$. Given a natural number n, we say $r \in \mathcal{W}(a,a)$ is a *reciprocal for* n if $\sum_{i=1}^{n} r = \mathrm{id}_a$. A reciprocal for n is unique if it exists. We write $1/n$ for the reciprocal for n.

Lemma 5.8. *Let $\mathcal{W}$ be a $\Sigma\mathbf{Mon}$-category. If $\mathcal{W}(I,I)$ has reciprocals for n, then so does $\mathcal{W}(a,a)$ for every $a \in \mathrm{ob}(\mathcal{W})$.*

5.2 Associated Matrices of Weighted Profunctors

Let $\mathcal{W}$ be a $\Sigma\mathbf{Mon}_t$-SMCC, fixed below. Assume that, for each $n \in \mathbb{N}$, the Σ-monoid $\mathcal{W}(I,I)$ has the reciprocal for n.

A category $\mathcal{A}$ is *countable* if the collection of morphisms is countable (then $\mathrm{ob}(\mathcal{A})$ is also countable). It is *locally-finite* if $\mathcal{A}(a,a')$ is finite for every $a, a' \in \mathrm{ob}(\mathcal{A})$. We write $\mathrm{ob}(\mathcal{A})/\mathrm{iso}$ for the collection of isomorphic classes of objects in $\mathcal{A}$.

Definition 5.9 (Matrix category)**.** The *matrix category* $\mathbf{Mat}(\mathcal{W})$ is defined by the following data. An object is a weighted category $A : \mathcal{A} \to \mathcal{W}^{\mathrm{op}}$ such that $\mathcal{A}$ is a countable, locally-finite groupoid with a strict factorisation system. A morphism $f : (\mathcal{A},A) \to (\mathcal{B},B)$ is a family $\{f_{a,b} : A(a) \to B(b)\}_{(a,b)\in\mathrm{ob}(\mathcal{A})\times\mathrm{ob}(\mathcal{B})}$ of morphisms in $\mathcal{W}$ that respects the action of $\mathcal{A}$- and $\mathcal{B}$-morphisms (i.e. $B(g)\circ f_{a,b}\circ A(h) = f_{a',b'}$ for every $h : a \to a'$ and $g : b' \to b$). Composition of $f = \{f_{b,a}\} : (\mathcal{A},A) \to (\mathcal{B},B)$ and $g = \{g_{c,b}\} : (\mathcal{B},B) \to (\mathcal{C},C)$ is defined by

$$(g\circ f)_{c,a} := \sum_{[b]\in\mathrm{ob}(\mathcal{B})/\mathrm{iso}} g_{c,b}\circ f_{b,a}$$

where the sum is that of $\mathcal{W}(A(a),C(c))$. The identity $\mathrm{id} : (\mathcal{A},A) \to (\mathcal{A},A)$ is defined by $\mathrm{id}_{a,a'} := 1/\#\mathcal{A}(a,a') \sum_{h\in\mathcal{A}(a,a')} A(h)$.

Now we are ready to define the associated matrix formally. A profunctor $F : \mathcal{A} \nrightarrow \mathcal{B}$ is said to be *countable* if $F(a,b)$ is countable for every $a \in \mathrm{ob}(\mathcal{A})$ and $b \in \mathrm{ob}(B)$.

Definition 5.10 (Associated matrix)**.** Let $\mathcal{A},\mathcal{B} \in \mathrm{ob}(\mathbf{Mat}(\mathcal{W}))$. Given a $\mathcal{W}$-weighted countable profunctor $F : \mathcal{A} \nrightarrow \mathcal{B}$ with weight function $\varpi_{a,b} : F(a,b) \to \mathcal{W}(A(a),B(b))$, the *associated matrix* is a morphism $\|(F,\varpi)\| : \mathcal{A} \to \mathcal{B}$ in $\mathbf{Mat}(\mathcal{W})$ given by (1).

A morphism $f = \{f_{a,b}\}_{a,b} : (\mathcal{A},A) \to (\mathcal{B},B)$ in $\mathbf{Mat}(\mathcal{W})$ bijectively corresponds to a weight function ϖ^f for the locally-terminal profunctor $F : \mathcal{A} \nrightarrow \mathcal{B}$ (i.e. $F(b,a) = \{*\}$ for every a and b): let us define $\varpi^f_{b,a}(*) = f_{a,b}$. Although this correspondence is not functorial, the SMCC structure of $\mathbf{Mat}(\mathcal{W})$ can be defined via the correspondence. The biproducts and symmetric tensor powers are obtained from $\mathbf{Pr}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}}$ by applying $\|\cdot\|$.

Theorem 5.11. $\mathbf{Mat}(\mathcal{W})$ *is a Lafont category with countable biproducts.*

Remark 5.12. The *countable biproduct completion* $\mathcal{W}^{\prod}$ (cf. [19–21]) is a full subcategory of $\mathbf{Mat}(\mathcal{W})$ consisting of objects $A : \mathcal{A} \to \mathcal{W}$ with $\mathcal{A}$ discrete. (I.e. objects of $\mathcal{W}^{\prod}$ are countable lists of objects of $\mathcal{W}$.) A notable difference is that $\mathbf{Mat}(\mathcal{W})$ is a Lafont category, whereas $\mathcal{W}^{\prod}$ is not. The objects $\mathcal{A}$ of $\mathbf{Mat}(\mathcal{W})$ with nontrivial isomorphisms (i.e. those not in $\mathcal{W}^{\prod}$) are essential for $\mathbf{Mat}(\mathcal{W})$ to be a Lafont category. In a related construction in [28], a morphism is required to be invariant under the action of chosen permutations of basis vectors. This can be seen as a special case of requirements for morphisms (i.e. $B(g)\circ f_{a,b}\circ A(h) = f_{a',b'}$) in $\mathbf{Mat}(\mathcal{W})$: if $(\mathcal{A},A)$ is an interpretation of a simple type, then $A(\varphi)$ is composed of structural isomorphisms in $\mathcal{W}$, which are permutations of basis vectors if $\mathcal{W} = \mathbf{CPM}_s$. □

5.3 P-visible Weighted Profunctors

Unfortunately, as mentioned at the beginning of this section, $\|\cdot\|$ is not functorial. This subsection introduces a subcategory of $\mathbf{Pr}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}}$, which contains the interpretations of $\lambda_{\mathcal{W}}$-terms, and to which the restriction of $\|\cdot\|$ is a functor.

Definition 5.13 (P-visible profunctor)**.** Let S and T be simple types. A countable profunctor $F : [\![S]\!] \nrightarrow [\![T]\!]$ is *P-visible* if, for each

$a \in [\![S]\!]$, $b \in [\![T]\!]$ and $x \in F(b,a)$, there exists a rigid resource term $x : a \vdash \widetilde{t} : b$ such that $\mathrm{fix}(x) \subseteq \mathrm{fix}(\widetilde{t})$ (here $\mathrm{fix}(x) = \{(\varphi,\psi) \mid \varphi \cdot x \cdot \psi = x\}$). A weighted profunctor is P-visible if so is the underlying profunctor. We write $(\mathbf{Pr}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}})\restriction_{\mathrm{V}}$ for the subcategory whose objects are the interpretations of simple types and whose morphisms are the P-visible ones.

By definition, the interpretation of a $\lambda_{\mathcal{W}}$-term in $\mathbf{Pr}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}}$ lives in $(\mathbf{Pr}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}})\restriction_{\mathrm{V}}$. It has the structure of a $\lambda_{\mathcal{W}}$-model induced by that of $\mathbf{Pr}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}}$. The embedding $(\mathbf{Pr}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}})\restriction_{\mathrm{V}} \to \mathbf{Pr}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}}$ strictly preserves this structure.

Lemma 5.14. *$\|\cdot\| : (\mathbf{Pr}/\!\!/^{\mathbf{Cat}}_{\mathcal{W}^{\mathrm{op}}})\restriction_{\mathrm{V}} \to \mathbf{Mat}(\mathcal{W})$ is a $\lambda_{\mathcal{W}}$-model morphism.*

Proof. (Sketch) The most nontrivial part is the functoriality of $\|\cdot\|$. Let $(F, \varpi^F) : (\mathcal{A}, A) \nrightarrow (\mathcal{B}, B)$ and $(G, \varpi^G) : (\mathcal{B}, B) \nrightarrow (\mathcal{C}, C)$ be P-visible profunctors. The key observation is that, thanks to P-visibility, for every $(y,x) \in G(c,b) \times F(b,a)$ and $f : b \to b$, we have $(y, f \cdot x) = (y \cdot f, x)$ implies $f = \mathrm{id}$. Then each equivalence class of $G(c,b) \times F(b,a)$ by $\sim$ (where $\sim$ is that appears in the composition of profunctors) has exactly $\#\mathcal{B}(b,b)$ elements. Hence

$$\sum_{[(y,x)]_\sim \in (G(c,b)\times F(b,a))/\sim} \varpi^G(y) \circ \varpi^F(x) = \frac{1}{\#\mathcal{B}(b,b)} \sum_{(y,x)\in G(c,b)\times F(b,a)} \varpi^G(y) \circ \varpi^F(x).$$

A calculation using this fact and $\#\mathcal{B}(b,b) = \#\mathcal{B}^+(b,b) \times \#\mathcal{B}^-(b,b)$ shows $\|G \circ F\|_{c,a} = (\|G\| \circ \|F\|)_{c,a}$. □

Corollary 5.15. *For every program P, the interpretation of P in $\mathbf{Mat}(\mathcal{W})$ is $\sum_{\pi\in Eval(P)} \varpi(\pi)$ where the sum is that in $\mathcal{W}(I,I)$.*

So far, we have assumed that $\mathcal{W}$ is a $\Sigma\mathbf{Mon}_t$-SMCC with reciprocals n^{-1} for every natural number n. We can also deal with $\Sigma\mathbf{Mon}$-SMCCs such as **FdHilb** and $\mathbf{CPM}_s$ as follows. First $\Sigma\mathbf{Mon}_t$ is a reflexive full subcategory of $\Sigma\mathbf{Mon}$ (see [15]) and it is an exponential ideal (i.e., for every $\mathfrak{N} \in \Sigma\mathbf{Mon}_t$ and $\mathfrak{M} \in \Sigma\mathbf{Mon}$, we have $\mathfrak{M} \multimap \mathfrak{N} \in \Sigma\mathbf{Mon}_t$). A general result shows that $\Sigma\mathbf{Mon}_t$ is an SMCC and the adjunction between $\Sigma\mathbf{Mon}$ and $\Sigma\mathbf{Mon}_t$ is symmetric monoidal [15, Corollary 3.9]. Let us write $T : \Sigma\mathbf{Mon} \to \Sigma\mathbf{Mon}_t$ for the left adjoint of the inclusion $\Sigma\mathbf{Mon}_t \to \Sigma\mathbf{Mon}$. Thanks to a result in [23], a $\Sigma\mathbf{Mon}$-SMCC $\mathcal{W}$ induces a $\Sigma\mathbf{Mon}_t$-SMCC $T\mathcal{W}$ obtained by applying T to each hom-object. Let $\eta_{\mathcal{W}(I,I)}$ be the unit $\mathcal{W}(I,I) \to T(\mathcal{W}(I,I)) = (T\mathcal{W})(I,I)$, which is injective.

Theorem 5.16 (Adequacy)**.** *Assume that $\mathcal{W}$ is a $\Sigma\mathbf{Mon}$-SMCCs with reciprocals for natural numbers. For every $\lambda_{\mathcal{W}}$ program P, we have $\sum_{\pi\in Eval(P)} \varpi(\pi) \sqsubseteq \eta_{\mathcal{W}(I,I)}([\![P]\!]_{\mathbf{Mat}(T\mathcal{W})})$.*

Remark 5.17. Taking $\mathcal{W} = T(\mathbf{CPM}_s)$, this theorem shows that $\mathbf{Mat}(\mathcal{W})$ is adequate for the calculus in [28]; indeed the model $\mathbf{Mat}(\mathcal{W})$ is essentially the same model as in [28], at least on the interpretation of types, except that [28] applies a different completion to $\mathbf{CPM}_s$. Although **FdHilb** is also $\Sigma\mathbf{Mon}$-SMCC and their calculus can be embedded into $\lambda_{\mathbf{FdHilb}}$, the category $\mathbf{Mat}(T(\mathbf{FdHilb}))$ is not an adequate model. This is because the sum in $\mathbf{FdHilb}(I,I) = \mathbb{C}$ differs from what we needed; recall that the meaning of a $\lambda_{\mathbf{FdHilb}}$ program P is $\sum_{\pi\in Eval(P)} \varpi(\pi)\varpi(\pi)^*$, not $\sum_{\pi\in Eval(P)} \varpi(\pi)$. □

Acknowledgments

The authors would like to thank Marcelo Fiore for insightful discussions. This work was supported by JSPS KAKENHI 15H05706, 16K16004 and 18K11156, and EPSRC grant EP/M023974/1.

References

[1] Samson Abramsky, Radha Jagadeesan, and Pasquale Malacaria. 2000. Full Abstraction for PCF. *Information and Computation* 163, 2 (2000), 409–470.
[2] Jean Bénabou. 1967. *Introduction to bicategories.* Springer Berlin Heidelberg, Berlin, Heidelberg, 1–77.
[3] Jean Bénabou. 2000. Distributors at work. (2000). Course notes, TU Darmstadt.
[4] François Bergeron, Gilbert Labelle, and Pierre Leroux. 1997. *Combinatorial Species and Tree-like Structures.* Cambridge Univ. Press.
[5] Vincent Danos and Thomas Ehrhard. 2011. Probabilistic coherence spaces as a model of higher-order probabilistic computation. *Inf. Comput.* 209, 6 (2011), 966–991.
[6] Thomas Ehrhard and Laurent Regnier. 2008. Uniformity and the Taylor expansion of ordinary lambda-terms. *Theor. Comput. Sci.* 403, 2-3 (2008), 347–372.
[7] Thomas Ehrhard, Christine Tasson, and Michele Pagani. 2014. Probabilistic coherence spaces are fully abstract for probabilistic PCF. In *POPL* (2014). 309–320.
[8] Marcelo P. Fiore. 2005. Mathematical Models of Computational and Combinatorial Structures. In *FoSSaCS* (2005). 25–46.
[9] Marcelo P. Fiore, Nicola Gambino, J. Martin E. Hyland, and Glynn Winskel. 2007. The cartesian closed bicategory of generalised species of structures. *J. London Maths. Soc.* 77 (2007), 203–220.
[10] Jean-Yves Girard. 1988. Normal functors, power series and λ-calculus. *Ann. Pure Appl. Logic* 37, 2 (1988), 129–177.
[11] Esfandiar Haghverdi. 2001. Partially Additive Categories and Fully Complete Models of Linear Logic. In *TLCA* (2001). 197–216.
[12] Esfandiar Haghverdi and Philip J. Scott. 2006. A categorical model for the geometry of interaction. *Theor. Comput. Sci.* 350, 2-3 (2006), 252–274.
[13] Russell Harmer and Guy McCusker. 1999. A fully abstract game semantics for finite nondeterminism. In *LICS* (1999). 422–430.
[14] Ryu Hasegawa. 2002. Two applications of analytic functors. *Theor. Comput. Sci.* 272, 1-2 (2002), 113–175.
[15] Naohiko Hoshino. 2012. A Representation Theorem for Unique Decomposition Categories. *Electr. Notes Theor. Comput. Sci.* 286 (2012), 213–227.
[16] J. M. E. Hyland and C.-H. Luke Ong. 2000. On Full Abstraction for PCF: I, II, and III. *Information and Computation* 163, 2 (2000), 285–408.
[17] Andre Joyal. 1981. Une théorie combinatoire des séries formelles. *Adv. Math.* 42 (1981), 1–82.
[18] Andre Joyal. 1986. Foncteurs analytiques et espèces de structures. In *Combinatoire Énumérative.* Lecture Notes in Mathematics, Vol. 1234. Springer, Berlin, 126–159.
[19] James Laird. 2016. Fixed points in quantitative semantics. In *LICS* (2016). 347–356.
[20] James Laird. 2017. From Qualitative to Quantitative Semantics - By Change of Base. In *FoSSaCS* (2017). 36–52.
[21] Jim Laird, Giulio Manzonetto, Guy McCusker, and Michele Pagani. 2013. Weighted Relational Models of Typed Lambda-Calculi. In *LICS* (2013). 301–310.
[22] François Lamarche. 1992. Quantitative Domains and Infinitary Algebras. *Theor. Comput. Sci.* 94, 1 (1992), 37–62.
[23] R. B. B. Lucyshyn-Wright. 2016. Relative symmetric monoidal closed categories I: Autoenrichment and change of base. *Theory and Applications of Categories* 31 (2016), 138–174.
[24] Damiano Mazza, Luc Pellissier, and Pierre Vial. 2018. Polyadic approximations, fibrations and intersection types. *PACMPL* 2, POPL (2018), 6:1–6:28.
[25] Paul-André Melliès. 2003. Asynchronous games 1: Uniformity by group invariance. (2003). unpublished manuscript.
[26] Paul-André Melliès. 2009. *Categorical Semantics of Linear Logic.* 1–196 pages.
[27] Paul-André Melliès, Nicolas Tabareau, and Christine Tasson. 2009. An Explicit Formula for the Free Exponential Modality of Linear Logic. In *ICALP* (2009). 247–260.
[28] Michele Pagani, Peter Selinger, and Benoît Valiron. 2014. Applying quantitative semantics to higher-order quantum computing. In *POPL* (2014). 647–658.
[29] Luc Pellissier. 2017. *Réduction et Approximation Linéaires.* Ph.D. Dissertation. Université Paris 13.
[30] Peter Selinger. 2004. Towards a quantum programming language. *Mathematical Structures in Computer Science* 14, 4 (2004), 56.
[31] Peter Selinger and Benoît Valiron. 2008. On a Fully Abstract Model for a Quantum Linear Functional Language. *Electronic Notes in Theoretical Computer Science* 210, (2008), 123–137.
[32] Takeshi Tsukada, Kazuyuki Asada, and C.-H. Luke Ong. 2017. Generalised species of rigid resource terms. In *LICS* (2017). 1–12.
[33] Takeshi Tsukada and C.-H. Luke Ong. 2015. Nondeterminism in Game Semantics via Sheaves. In *LICS* (2015). 220–231
[34] Lionel Vaux. 2009. The algebraic lambda calculus. *Mathematical Structures in Computer Science* 19, 5 (2009), 1029–1059.