Resource Categories from Differential Categories

Lison Blondeau-Patissier

Resource Categories from Differential Categories

Lison Blondeau-Patissier[1]

[1]Aix Marseille Univ, CNRS, I2M and LIS, Marseille, France

Resource categories were recently introduced to capture the categorical structure of pointer concurrent games, and in particular to characterize morphisms behaving "linearly". These linear morphisms correspond to (beta-normal, eta-expanded) terms of the resource lambda-calculus. Resource calculus is closely related to Erhrard and Regnier's differential lambda-calculus, which is usually interpreted in differential categories (defined by Blute, Cockett, Lemay and Seely). However, strategies of pointer concurrent games are not built from a model of linear logic, so their categorical structure is not a differential category.

Nevertheless, resource categories can be constructed from differential categories. We present such a construction in this paper, starting from an additive monoidal category and building a resource category from it.

1 Introduction

Resource categories were recently introduced in [BCVA23] as a way to capture the categorical structure of *pointer concurrent games*, a game semantics model also first presented in that paper. The aim was to obtain a categorical framework enabling the characterization of morphisms behaving "linearly", to show that these morphisms in pointer concurrent games are in bijection with terms of the *resource λ-calculus*.

Both game semantics and resource calculus have been well-studied lines of work for years, and both of them consider (multisets of) finite executions of programs to represent programs with possibly infinite behavior.

Game semantics model programs as processes, focusing on the interactions between the program and its environment. These interactions are represented as a *game* between two protagonists, one of them called Player representing the program and the other called Opponent representing the context. Information tokens exchanged between them are seen as moves, in a game whose rules depend on the type of the program. A *play* represents one possible execution of a program; programs themselves are represented by *strategies*, which are sets of plays (corresponding to every possible execution of the program). Game semantics is known for its many full-abstraction result, particularly in [AJM13] and [HO00] which introduced two standard game models, respectively called AJM games and HO games from the name of the authors. A key notion in HO games is the one of *innocence* of a strategy, corresponding to *pure* programs, or to λ-terms without references. Such programs will react in the same way no matter how many times Opponent duplicates a request to evaluate a subprogram, or the order in which they chose to evaluate subprograms. This led to Melliès' homotopy equivalence on plays introduced in [Mel06], quotienting plays by Opponent's scheduling. Those quotiented plays can alternatively be seen as *augmentations*, another representation of plays introduced in [BC21] and inspired by concurrent games

(see [CCRW17] for an introduction to concurrent games). Later [BCVA23] considered composition of augmentations, defining strategies as sums of augmentations, forming the category of *pointer concurrent games*.

Resource calculus, on the other hand, arose from linear logic (introduced in [Gir87]) and quantitative semantics ([Gir88]). Unlike usual λ-calculus, where the substitution $M[N/x]$ can duplicate the term N as many times as the variable x occurs freely in the term M (or even more, if x itself is duplicated at some point of the execution), resource λ-calculus sees terms as *resources* which can each be used exactly once. Hence, the substitution is not defined with a term N anymore, but rather with a multiset of terms $[N_1, \dots, N_n]$, which will each replace exactly one occurence of x in M (the exact bijection being chosen non-deterministically, *via* a sum of resource terms corresponding to all possible substitutions). This allows for a control of the number of copies of N, and for example ensures that the reduction terminates. Replacing arguments with multisets of terms in λ-calculi first emerged with the λ-calculus with multiplicities [Bou93], the term *resource* appearing a few years later in [BCL99]. Resource calculus is closely related to *differential λ-calculus*, developed in [ER03], which is a λ-calculus equipped with a formal differentiation. Intuitively, the derivative of a term $M : A \to B$ is the "linear" part of M, that is a function which uses its argument *exactly once* – this corresponds to the notion of differentiation in analysis, in which the derivative of a function is a linear approximation. Resource calculus is the finitary fragment of differential λ-calculus, not containing *pure* λ-terms (which are terms that can access their arguments "as needed", possibly infinitely many times – represented in linear logic by the type $!A \multimap B$).

The correspondence between those two models was studied for instance by Tsukada and Ong in [TO16], where they show that normal, η-long resource terms are in bijection with Hyland-Ong plays up to Melliès' homotopy equivalence. This correspondence was further developed in [BCVA23], establishing a denotational interpretation, invariant under reduction, of resource terms as strategies for pointer concurrent games. To prove this result, the categorical structure of pointer concurrent games was exposed in a way which enables the characterization of strategies interpreting resource terms. This led to the definition of *resource categories*, categories in which some morphisms can be identified as "linear": either "using their resources" exactly once, or being able to be "used as a resource" exactly once.

Usually, models of linear logic are interpreted using differential categories, introduced in [BCS06] as a categorical framework for differential linear logic. But in [BCVA23] resource-calculus was studied in relation with games, and strategies are *not* linear: they do no have a linear behavior in general – the identity for pointer concurrent games is not even finite. Hence resource categories are *not* a category of resource terms[1] – in fact, this interpretation of resource terms lacks an identity. Nonetheless, resource categories are built using similar constructions to some differential categories, more precisely *monoidal storage categories* as described in [BCLS20]. The intuition behind these similarities is that the exponential ! of differential categories allows us to go from linear morphisms from A to B, to morphisms from $!A$ to $!B$, which behave linearly with respect to $!A$ and $!B$, but not with respect to the original objects A and B. These intuitions will guide us in our construction of resource categories from additive monoidal storage categories – which are the categories we mostly refer to when mentionning "differential categories" in this paper, although differential categories in general are a much wider notion.

Outline. We start by giving some general categorical definitions in Section 2, before introducing more precisely resource categories in Section 3. We then focus on differential categories in Section 4, defining additive monoidal storage categories – our main focus is not to give an exhaustive presentation of differential categories in this paper, but rather to present the particular category which we will use to build a resource category. We detail this construction in Section 5, stating in Theorem 1 that we obtain a resource category.

[1] Nevertheless we call them resource categories because they contain morphisms acting like resource terms.

2 Categorical Preliminaries

This section presents some categorical notions which are used throughout this paper: symmetric monoidal categories, string diagrams, and (co)monoids. We direct the interested reader to [ML71] for an introduction to category theory.

Symmetric Monoidal Categories. A *monoidal category* is a category equipped with a *tensor product*; if it comes with a notion of commutativity of this tensor, the monoidal category is additionally *symmetric*.

Definition 1 (Monoidal Category)**.** A *monoidal* category is a category $\mathcal{C}$ equipped with:

- a functor $\otimes\colon \mathcal{C} \times \mathcal{C} \to \mathcal{C}$ called the *tensor*;
- an object $I \in \mathcal{C}$ called the *unit*;
- the following isomorphisms natural in A, B, C:

$$\begin{aligned} \alpha_{A,B,C}\colon (A \otimes B) \otimes C &\to A \otimes (B \otimes C) && (\textit{associator}) \\ \lambda_A\colon I \otimes A &\to A && (\textit{left-unitor}) \\ \rho_A\colon A \otimes I &\to A && (\textit{right-unitor}) \end{aligned}$$

such that for any objects A, B, C, D,

$$(\mathsf{id}_A \otimes \lambda_B) \circ \alpha_{A,I,B} = \rho_A \otimes \mathsf{id}_B \qquad \text{(triangle identity)}$$

and the diagram of Figure 1 commutes.

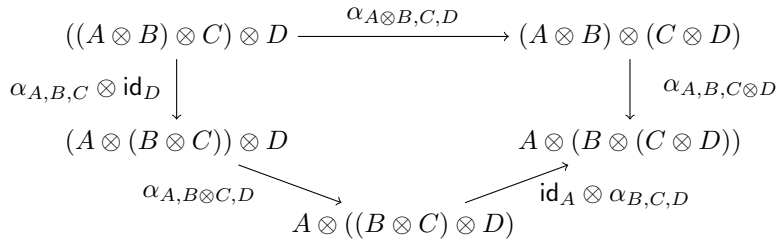


Figure 1. Pentagon identity

Definition 2 (Symmetric Monoidal Category)**.** A *symmetric monoidal category* (smc for short) is a monoidal category $(\mathcal{C}, \otimes, I)$ equipped with a natural isomorphism

$$\sigma_{A,B}\colon A \otimes B \to B \otimes A \qquad (\textit{symmetry})$$

such that $\sigma_{B,A} \circ \sigma_{A,B} = \mathsf{id}_{A \otimes B}$ and the diagram of Figure 2 commutes.

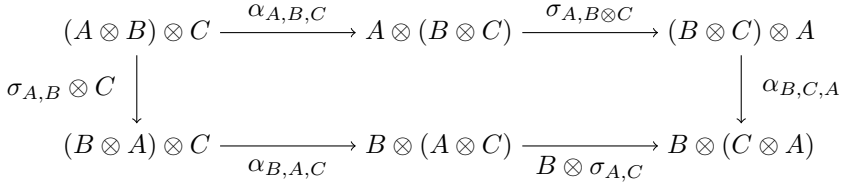
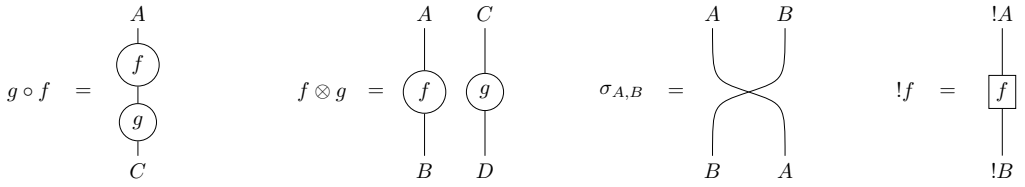


Figure 2. Hexagon identity

In this paper, all categories will be equipped with a symmetric monoidal structure (using $\otimes$ for the tensor and I for the unit), unless stated otherwise. For the sake of readability, we mostly treat associator and unitors as identities, as justified by Mac Lane's coherence theorem (see [ML63, Theorem 5.2] for the historical statement and [ML71, Chapter 7] for the more standard, textbook version).

String diagrams. As in [BCLS20], we use string diagrams, read from top to bottom, for a graphical reprentation of some categorical equations (see [JS91] for a historical introduction and [Sel10] for a survey of graphical languages). Given two morphisms $f\colon A \to B$ and $g\colon B \to C$, the composition $g \circ f\colon A \to C$ is presented in Figure 3a. The tensor of $f\colon A \to B$ and $g\colon C \to D$ is represented using two wires side by side as in Figure 3b, and the symmetry by crossing the wires as in Figure 3c. We will often omit the labels on wires if they are clear from the context; we also omit I wires because we treat unitors as identities.

Moreover, differential categories (introduced in Section 4) involve an endofunctor !. Following [BCS06], we represent $!(f)\colon !A \to !B$ with a squared box (Figure 3d).



3a. Composition **3b.** Tensor **3c.** Symmetry **3d.** !

Figure 3. String diagrams

(Co)Monoids. Finally, most categories we present are equipped with (co)monoids.

Definition 3 (Monoids)**.** A monoid in a smc $\mathcal{C}$ is an object A equipped with:

$$\mu_A\colon A \otimes A \to A \qquad (\textit{multiplication})$$
$$\eta_A\colon I \to A \qquad (\textit{unit})$$

satisfying the following equations:

$$\mu_A \circ (\mu_A \otimes \mathsf{id}_A) = \mu_A \circ (\mathsf{id}_A \otimes \mu_A) \qquad (\textit{associativity of } \mu)$$
$$\mu_A \circ (\eta_A \otimes \mathsf{id}_A) = \mathsf{id}_A = \mu_A \circ (\mathsf{id}_A \otimes \eta_A) \qquad (\textit{neutrality of } \eta)$$

which are presented in the string diagrams of Figure 4.

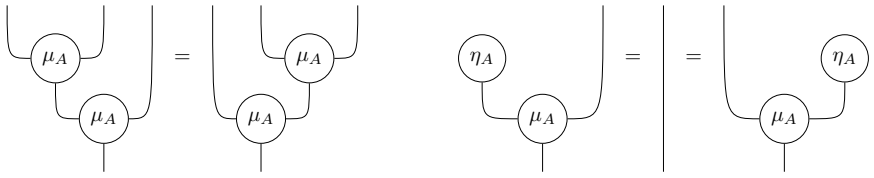


Figure 4. Monoid laws

Additionally, (A, μ_A, η_A) is *commutative* if $\mu_A \circ \sigma_{A,A} = \mu_A$.

Definition 4 (Comonoids)**.** A comonoid in a smc $\mathcal{C}$ is an object A equipped with:

$$\delta_A\colon A \to A \otimes A \qquad (\textit{co-multiplication})$$
$$\varepsilon_A\colon A \to I \qquad (\textit{co-unit})$$

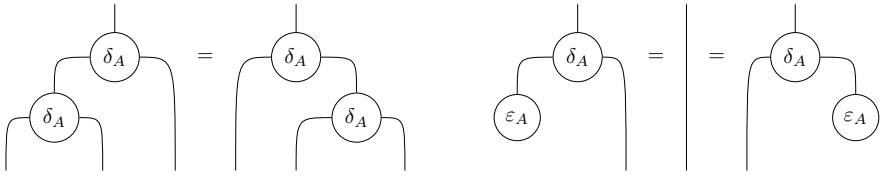


Figure 5. Comonoid laws

satisfying the equations of Figure 5.

Additionally, $(A, \delta_A, \varepsilon_A)$ is *commutative* if $\sigma_{A,A} \circ \delta_A = \delta_A$.

3 Resource Categories

Resource categories both capture the categorical structure of pointer concurrent games and allow us to identify morphisms behaving *linearly*, such as terms of resource calculus. First, composition in games generates sums of morphisms (which are also morphisms); likewise, substitution in the resource calculus generates sums of terms. Hence, resource categories have an *additive* structure. Moreover, resource terms are built using multisets of terms; we would like a way to "flatten" multisets of morphisms into one morphism. This operation is constructed with *bialgebra* morphisms. Finally, resource categories are *not* linear, because strategies, the morphisms in pointer concurrent games, do not have a linear behavior in general. However, we want to be able to characterize the morphisms that do behave linearly; which is achieved using the *pointed identities* morphisms.

Additivity. We call *additive* categories that are enriched over commutative monoids[2].

Definition 5 (Additive SMC (ASMC))**.** An *additive symmetric monoidal category* (asmc) is a symmetric monoidal category (see Definition 2) where each hom-set is a commutative monoid, with an addition $+$ and a zero 0, such that composition and tensor distribute over the additive structure:

$$h \circ (f + g) \circ k = h \circ f \circ k + h \circ g \circ k \qquad h \circ 0 \circ k = 0$$
$$h \otimes (f + g) \otimes k = h \otimes f \otimes k + h \otimes g \otimes k \qquad h \otimes 0 \otimes k = 0$$

for any morphisms k, f, g, h.

Bialgebras. Resource categories are equipped with *bialgebras*, which are a monoid and a comonoid with coherence laws between the morphisms.

Definition 6 (Bialgebra)**.** Consider C an additive symmetric monoidal category.

A *bialgebra* on C is $(A, \delta_A, \varepsilon_A, \mu_A, \eta_A)$ with

- (A, μ_A, η_A) a commutative monoid (see Definition 3),
- $(A, \delta_A, \varepsilon_A)$ a commutative comonoid (see Definition 4),
- and additional bialgebra laws presented in Figure 6.

In resource categories, every object has a bialgebra structure. Intuitively, comonoids (A, δ_A, η_A) are a way to represent *duplications* and duplicable objects: if a request is made on the output of δ_A on either side of the tensor, the request is forwarded to its input. Monoids

[2]We follow the definition of [BCS06, Section 2], which differs from the one given in [ML71].

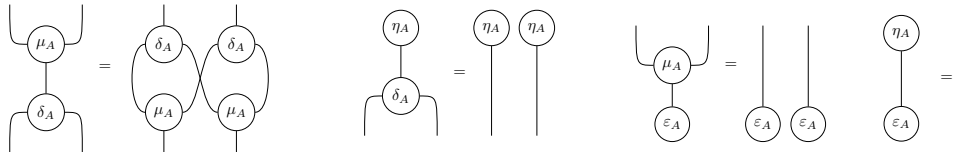


Figure 6. Bialgebra laws

$(A, \mu_A, \varepsilon_A)$ reflect the sums coming from compositions of strategies: requests made on the output of μ_A are forwarded non-deterministically (*via* a sum) to its input on either side of the tensor.

Strategies in pointer concurrent games are sums of augmentations, and augmentations have a forestial structure: they are, in a way, finite multisets of tree-like sub-augmentations. This matches the fact that in resource calculus, terms are applied to multiset of terms instead of terms. Bialgebra morphisms allow us to formalize this intuition and to flatten any multiset of morphisms into a single morphism. Indeed, for any objects A, B, we define the empty multiset of morphisms from A to B as:

$$1_{A,B} = \eta_B \circ \varepsilon_A \in \mathcal{C}(A, B)\,,$$

and for any morphisms $f, g \colon A \to B$, the "union":

$$f * g = \mu_B \circ (f \otimes g) \circ \delta_A \in \mathcal{C}(A, B)\,,$$

capturing the idea of the union of the multisets f and g.

With these definitions, $(\mathcal{C}(A, B), *, 1_{A,B})$ is a commutative monoid (and $\mathcal{C}(A, B)$ is a commutative semiring, where the composition and the tensor only preserve the additive monoid). We define the n-ary union (unambiguously thanks to the associativity of $*$): given a multiset of morphisms $\bar{f} = [f_1, \ldots, f_n]$ in $\mathcal{M}_f(\mathcal{C}(A, B))$, we set

$$\Pi \bar{f} = f_1 * \ldots * f_n \in \mathcal{C}(A, B)\,.$$

Hence, we send multisets of morphisms to single morphisms *via* Π.

Pointed identities. Finally, we wish to characterize morphisms that "behave linearly" (in pointer concurrent game, they correspond to singleton multisets of tree-like augmentations, using their argument exactly once). To do so, we define a morphism called *pointed identity*, which acts as an identity *only for "linear morphisms"*.[3]

Definition 7 (Pointed identity [BCVA23, Definition 22])**.** Consider $\mathcal{C}$ an asmc where each object has a bialgebra structure. For any A, a *pointed identity* is $\mathsf{id}^\bullet_A \in \mathcal{C}(A, A)$ satisfying:

$$\mathsf{id}^\bullet_A \circ \mathsf{id}^\bullet_A = \mathsf{id}^\bullet_A \qquad (\textit{idempotent})$$
$$\varepsilon_A \circ \mathsf{id}^\bullet_A = 0 \qquad (\textit{non-erasable})$$
$$\mathsf{id}^\bullet_A \circ \eta_A = 0 \qquad (\textit{non-erasing})$$

and the equations of Figure 7.

These equations express the following properties of $\mathsf{id}^\bullet_A$:

> *non-duplicable*: the post-composition with the co-multiplication δ_A is the sum of "$\mathsf{id}^\bullet_A$ takes a request from the left-hand side of the tensor" and "$\mathsf{id}^\bullet_A$ takes a request from the right-hand side", but no situation in which $\mathsf{id}^\bullet_A$ takes requests from both sides simultaneously;

[3]The name *pointed* identity comes from the particular case of pointed identities in the resource categories of pointer concurrent games: tree-like augmentations corresponding to linear morphisms in games are called *pointed*, because their forestial structure has a unique root.

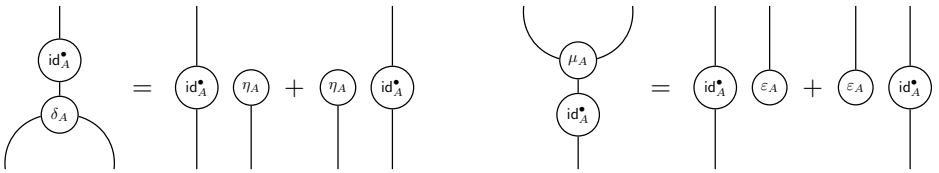


Figure 7. Laws for (co)multiplication and pointed identity

non-duplicative: the pre-composition with the multiplication μ_A is the sum of "$\mathsf{id}^\bullet_A$ forwards a request to the left-hand side of the tensor" and "$\mathsf{id}^\bullet_A$ forwards a request to the right hand side" but no "$\mathsf{id}^\bullet_A$ forwards the request to both sides".

This "strong linear" behavior of $\mathsf{id}^\bullet$ will allow us to characterize linear morphisms: those which are invariant by composition with the pointed identity.

Definition 8 ((Co-)Pointed Morphisms [BCVA23])**.** Consider A, B in an asmc $\mathcal{C}$ equipped with bialgebras, and the pointed identities $\mathsf{id}^\bullet_A$ and $\mathsf{id}^\bullet_B$.

Then $f \in \mathcal{C}(A,B)$ is *pointed* if $\mathsf{id}^\bullet_B \circ f = f$. We write $f \in \mathcal{C}_\bullet(A,B)$.

Dually, f is *co-pointed* if $f \circ \mathsf{id}^\bullet_A = f$. We write $f \in \mathcal{C}^\bullet(A,B)$.

Intuitively, pointed morphisms are morphisms behaving linearly for the substitution: they can only be used exactly once. Dually, co-pointed morphisms are morphisms behaving linearly with their arguments: they require exactly one resource.

Resource Categories We can now define resource categories.

Definition 9 (Resource Category [BCVA23, Definition 23])**.** Consider an asmc $\mathcal{C}$. It is a *resource category* if each object A has a bialgebra structure $(A, \delta_A, \varepsilon_A, \mu_A, \eta_A)$ with a pointed identity $\mathsf{id}^\bullet_A$, and bialgebras are compatible with the monoidal structure of $\mathcal{C}$ in the sense that the morphisms satisfy:

$$\varepsilon_{A\otimes B} = \lambda_I \circ (\varepsilon_A \otimes \varepsilon_B) \qquad \eta_{A\otimes B} = (\eta_A \otimes \eta_B) \circ \lambda_I \qquad \varepsilon_I = \eta_I = \mathsf{id}_I$$

and the equations of Figure 8.

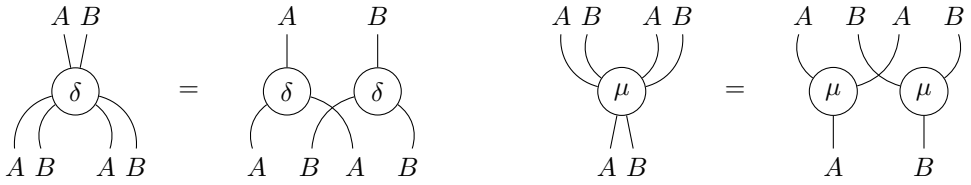


Figure 8. Compatibility of (co)monoids with the monoidal structure

Resource categories offer an interpretation of resource calculus, in which (singleton multisets of) terms are pointed morphisms. Linearity here is characterized using pointed identities; but linearity can also be linked to differential categories. Pointed identity laws are very similar to the dereliction and codereliction morphisms laws which occur in differential categories, which will guide us in our construction of a resource category in Section 5.

4 Differential Categories

Differential categories in general were introduced as a categorical framework for differential linear logic. In this paper, we focus on the *storage categories* of [BCLS20] – more precisely

on additive monoidal storage categories, which are the categories from which we construct resource categories in Section 5. In the current section we introduce all the components needed to define these storage categories (which we then define).

Coalgebra modality. *Coalgebra modalities* are similar to comonoid seen in Section 2, but they build over a comonad.

Definition 10 (Comonad)**.** Consider a category $\mathcal{C}$. A *comonad* on $\mathcal{C}$ is $(!, \mathsf{dig}, \mathsf{der})$ with

$$\begin{aligned} !\colon \mathcal{C} &\to \mathcal{C} && \text{an endofunctor,} \\ \mathsf{dig}_A\colon !A &\to !!A && \text{a natural transformation,} \\ \mathsf{der}_A\colon !A &\to A && \text{a natural transformation,} \end{aligned}$$

satisfying the equations of Figure 9

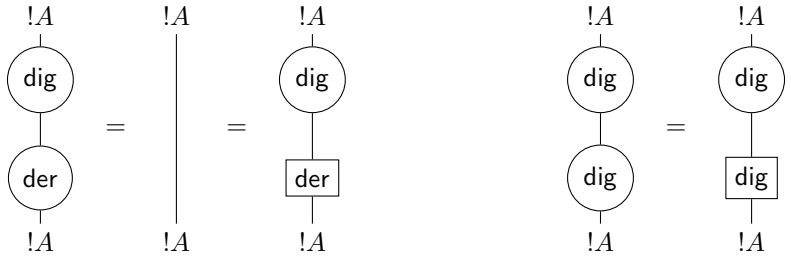


Figure 9. Comonad Laws

We write dig and der for the natural transformations because they match the *digging* and *dereliction* rules of linear logic (introduced in [Gir87]).

Definition 11 (Coalgebra modality [BCLS20, Definition 1])**.** A *coalgebra modality* on a symetric monoidal category $\mathcal{C}$ is $(!, \mathsf{dig}, \mathsf{der}, \Delta, \mathsf{e})$ with $(!, \mathsf{dig}, \mathsf{der})$ a comonad and two natural transformations

$$\Delta_A\colon !A \to !A \otimes !A \qquad\qquad \mathsf{e}_A\colon !A \to I$$

such that for any A, $(!A, \Delta_A, \mathsf{e}_A)$ is a comutative comonoid (Definition 4) and dig preserves Δ in the sense that

$$\Delta_{!A} \circ \mathsf{dig}_A = (\mathsf{dig}_A \otimes \mathsf{dig}_A) \circ \Delta_A\,.$$

Bialgebra modality. Next, we define *bialgebra modalities*, which again are reminiscent of bialgebras seen in previous sections.

Definition 12 (Bialgebra modality [BCLS20, Definition 4])**.** Consider an asmc $\mathcal{C}$. A *bialgebra modality* on $\mathcal{C}$ is $(!, \mathsf{dig}, \mathsf{der}, \Delta, \mathsf{e}, \nabla, \mathsf{u})$ with $(!, \mathsf{dig}, \mathsf{der}, \Delta, \mathsf{e})$ a coalgebra modality and for any A, $(!A, \Delta_A, \mathsf{e}_A, \nabla_A, \mathsf{u}_A)$ is a bialgebra such that:

$$\mathsf{der}_A \circ \nabla_A = (\mathsf{der}_A \otimes \mathsf{e}_A) + (\mathsf{e}_A \otimes \mathsf{der}_A)\ .$$

Definition 13 (Additive bialgebra modality [BCLS20, Definition 5])**.** An *additive bialgebra modality* in an asmc $\mathcal{C}$ is a bialgebra modality $(!, \mathsf{dig}, \mathsf{der}, \Delta, \mathsf{e}, \nabla, \mathsf{u})$ compatible with the additive structure in the sense of Figure 10.

Additive bialgebra modalities can be equipped with a *codereliction*, a natural transformation $\mathsf{cod}_A\colon A \to !A$ named codereliction because it has the inverse type to der_A, but which is *not* an inverse of der_A.

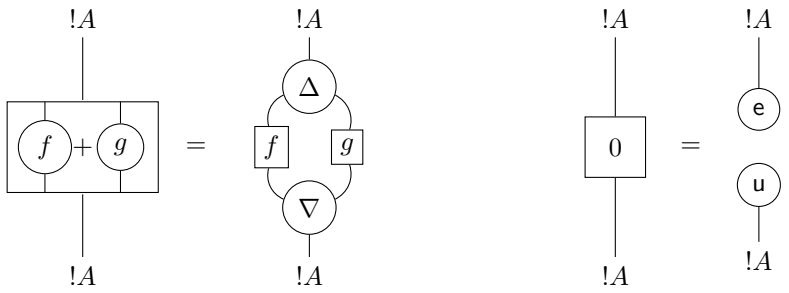


Figure 10. Addititve Bialgebra Modality Laws

Definition 14 (Codereliction [BCLS20, Definition 9][4])**.** Consider an asmc $\mathcal{C}$. A *codereliction* for an additive bialgebra modality $(!, \mathsf{dig}, \mathsf{der}, \Delta, \mathsf{e}, \nabla, \mathsf{u})$ is a natural transformation $\mathsf{cod}_A \colon A \to !A$ satisfying the following equations:

$$\mathsf{e}_A \circ \mathsf{cod}_A = 0 \qquad (\textit{constant rule})$$
$$\mathsf{der}_A \circ \mathsf{cod}_A = \mathsf{id}_A \qquad (\textit{linear rule})$$

as well as the equations of Figure 11.

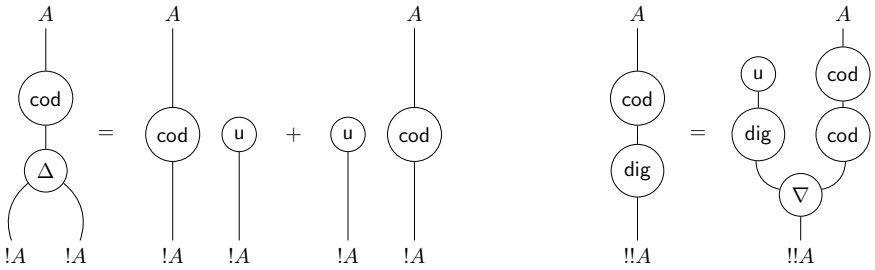


Figure 11. *Product rule* and *chain rule* of codereliction

Codereliction is a key notion of differential categories: in an asmc with a bialgebra modality, coderelictions induce deriving transformations[5] ([BCS06, Theorem 4.12]). In an asmc with an *additive* bialgebra modality, codereliction are *in bijection* with deriving transformations ([BCLS20, Theorem 4]).

Storage Categories. Now, we focus on *storage categories*, which are smc's with a coalgebra modality and a cartesian product &, with the following isomorphism:

$$!\,(A \& B) \;\cong\; !A \otimes !B$$

called *Seely isomorphism* (introduced as Δ iso in [See89]).

Recall that in a category $\mathcal{C}$, a *terminal object* is an object $\top$ such that for any object $A \in \mathcal{C}$, there exists a unique morphism in $\mathcal{C}(A, \top)$, noted $!_A \colon A \to \top$. A category $\mathcal{C}$ has *finite products* if it has a terminal object and for all objects $A, B \in \mathcal{C}$, there is a product $A \& B$ in $\mathcal{C}$ satisfying the universal property of products.

[4]The chain rule equation given here is the version presented in [Fio07] and not the (slightly longer) version of [BCS06, Definition 4.11]; however both are equivalent in monoidal storage categories ([BCLS20, Lemma 7 and Corollary 5]), which are what we are interested in in this paper.

[5]More precisely they are in bijection with deriving transformations satisfying the ∇- rule of [BCS06].

Definition 15 (Seely Isomorphism [BCLS20, Definition 10])**.** Consider an smc $\mathcal{C}$ with a binary product &, a terminal object $\top$, and a coalgebra modality $(!, \mathsf{dig}, \mathsf{der}, \Delta, \mathsf{e})$. It has *Seely isomorphisms* if the map $\chi_\top$ and the natural transformation χ, respectively defined as:

$$\chi_\top : !\top \xrightarrow{\mathsf{e}_\top} I \qquad \chi_{A,B} : !(A\&B) \xrightarrow{\Delta_{A\&B}} !(A\&B) \otimes !(A\&B) \xrightarrow{!\pi_1 \otimes !\pi_2} !A \otimes !B$$

are isomorphisms (with π_1, π_2 the projections of &).

Definition 16 (Monoidal Storage Category [BCLS20, Definition 10])**.** A *monoidal storage category* is a smc with finite products and a coalgebra modality with Seely isomorphisms.

We can consider storage categories with an additive structure.

Definition 17 (Additive Monoidal Storage Category [BCLS20, Definition 11])**.** An *additive monoidal storage category* is a category $\mathcal{C}$ that is a monoidal storage category and an additive symmetric monoidal category, with the same monoidal structure.

Additive storage categories are actually related to asmc's with a bialgebra modality.

Proposition 1 (from [BCLS20, Theorem 6])**.** *Consider an additive monoidal storage category $\mathcal{C}$. Then $(!, \mathsf{dig}, \mathsf{der}, \Delta, \mathsf{e}, \nabla, \mathsf{u})$ defined as:*

$$\begin{aligned}
\Delta_A &: !A \xrightarrow{!\langle \mathsf{id}_A, \mathsf{id}_A \rangle} !(A\&A) \xrightarrow{\chi_{A,A}} !A \otimes !A \\
\mathsf{e}_A &: !A \xrightarrow{!0} !\top \xrightarrow{\chi_\top} I \\
\nabla_A &: !A \otimes !A \xrightarrow{\chi^{-1}_{A,A}} !(A\&A) \xrightarrow{\pi_1 + \pi_2} !A \\
\mathsf{u}_A &: I \xrightarrow{\chi^{-1}_\top} !\top \xrightarrow{!0} !A
\end{aligned}$$

is a bialgebra modality.

In [BCLS20], the authors even prove that those two notions are equivalent.

5 How to build your own resource category with only these simple ingredients

Construction. We start from an additive monoidal storage category, defined in Section 4.

Definition 18 ($\mathsf{Res}(-)$)**.** Consider an additive monoidal storage category $\mathcal{C}$ with a codereliction cod. Using the notation of Proposition 1, we define the category $\mathsf{Res}(\mathcal{C})$ with the same objects as $\mathcal{C}$ and morphisms defined by:

$$\mathsf{Res}(\mathcal{C})(A, B) = \mathcal{C}(!A, !B)\,.$$

We define a bifunctor $\otimes_{\mathsf{Res}(\mathcal{C})}$ in the following way:

$$\begin{aligned}
A \otimes_{\mathsf{Res}(\mathcal{C})} B &= A \,\&\, B \\
f \otimes_{\mathsf{Res}(\mathcal{C})} g &= \chi^{-1}_{C,D} \circ (f \otimes g) \circ \chi_{A,B}
\end{aligned}$$

for any objects A, B, C, D and morphisms $f \in \mathsf{Res}(\mathcal{C})(A, C)$, $g \in \mathsf{Res}(\mathcal{C})(B, D)$.

Indeed, morphisms of a resource category do not all behave linearly, which is why we define $\mathsf{Res}(\mathcal{C})(A, B)$ as $\mathcal{C}(!A, !B)$: these are morphisms that are not necessarily linear with respect to A and B. To obtain a monoidal structure in $\mathsf{Res}(\mathcal{C})$, we prove that $\otimes_{\mathsf{Res}(\mathcal{C})}$ is a tensor, using Seely isomorphisms to see $!(A \otimes_{\mathsf{Res}(\mathcal{C})} B)$ as $!A \otimes !B$. The additive bialgebra modality structure of $\mathcal{C}$ easily induces a bialgebra structure in $\mathsf{Res}(\mathcal{C})$ (which we will define precisely in next proof). Finally, recall the parting remark of Section 3: pointed identity laws are very similar to the dereliction and codereliction laws of differential categories. We will thus construct $\mathsf{id}^\bullet$ from der and cod.

Altogether, we obtain a resource category.

Theorem 1. *Consider an additive monoidal storage category $\mathcal{C}$ with a codereliction* cod*. Then* $\mathsf{Res}(\mathcal{C})$ *is a resource category.*

Proof. We use notations of Definition 18. To make the equations less cluttered, we write $\mathcal{R}$ for $\mathsf{Res}(\mathcal{C})$ and A for id_A, and we omit indices for χ when they are clear from the context.

SMC. First, we prove that $(\mathcal{R}, \otimes_{\mathcal{R}}, \top)$ is a smc (Definition 2). We define:

$$\begin{aligned}
\alpha^{\mathcal{R}}_{A,B,C} &: !((A\&B)\&C) \xrightarrow{\chi} !(A\&B)\otimes !C \xrightarrow{\chi\otimes !C} (!A\otimes !B)\otimes !C \\
&\xrightarrow{\alpha^{\mathcal{C}}_{!A,!B,!C}} !A \otimes (!B\otimes !C) \xrightarrow{!A\otimes\chi^{-1}} !A\otimes !(B\&C) \xrightarrow{\chi^{-1}} !(A\&(B\&C)) \\
\lambda^{\mathcal{R}}_{A} &: !(\top\&A) \xrightarrow{\chi} !\top\otimes !A \xrightarrow{\chi_\top\otimes !A} I\otimes !A \xrightarrow{\lambda^{\mathcal{C}}_{!A}} !A \\
\rho^{\mathcal{R}}_{A} &: !(A\&\top) \xrightarrow{\chi} !A\otimes !\top \xrightarrow{!A\otimes\chi_\top} !A \otimes I \xrightarrow{\rho^{\mathcal{C}}_{!A}} !A \\
\sigma^{\mathcal{R}}_{A,B} &: !(A\&B) \xrightarrow{\chi} !A\otimes !B \xrightarrow{\sigma^{\mathcal{C}}_{!A,!B}} !B\otimes !A \xrightarrow{\chi^{-1}} !(B\&A)
\end{aligned}$$

and a direct diagram chasing, using smc properties of $\mathcal{C}$ and the fact that χ is an isomorphism, shows that $\mathcal{R}$ is a smc too.

Additivity. Direct from the definitions and the additive structure of $\mathcal{C}$.

Bialgebra structure. For any object A, we define the morphisms:

$$\begin{aligned}
\delta^{\mathcal{R}}_A &: !A \xrightarrow{\Delta^{\mathcal{C}}_A} !A\otimes !A \xrightarrow{\chi^{-1}} !(A\&A) & \qquad \varepsilon^{\mathcal{R}}_A &: !A \xrightarrow{\mathsf{e}_A} I \xrightarrow{\chi_\top^{-1}} !\top \\
\mu^{\mathcal{R}}_A &: !(A\&A) \xrightarrow{\chi} !A\otimes !A \xrightarrow{\nabla_A} !A & \qquad \eta^{\mathcal{R}}_A &: !\top \xrightarrow{\chi_\top} I \xrightarrow{\mathsf{u}_A} !A
\end{aligned}$$

Then one can check that $(A, \delta_A, \varepsilon_A, \mu_A, \eta_A)$ is a bialgebra by diagram chasing, using χ and the properties of the bialgebra modality of $\mathcal{C}$. Likewise, we check that it is compatible with the monoidal structure of $\mathcal{R}$ (Figure 8).

Pointed Identity. Finally, for any object A, we define the pointed identity as:

$$\mathsf{id}^{\bullet}_A : !A \xrightarrow{\mathsf{der}_A} A \xrightarrow{\mathsf{cod}_A} !A$$

and we check that it matches Definition 7:

- *idempotent:*

$$\begin{aligned}
\mathsf{id}^{\bullet}_A \circ \mathsf{id}^{\bullet}_A &= \mathsf{cod}_A \circ \mathsf{der}_A \circ \mathsf{cod}_A \circ \mathsf{der}_A && \text{(definition of } \mathsf{id}^{\bullet}_A\text{)} \\
&= \mathsf{cod}_A \circ \mathsf{id}_A \circ \mathsf{der}_A && \text{(linear rule of Definition 14)} \\
&= \mathsf{id}^{\bullet}_A && \text{(definition of } \mathsf{id}^{\bullet}_A\text{)}
\end{aligned}$$

- *non-erasable:*

$$\begin{aligned}
\varepsilon^{\mathcal{R}}_A \circ \mathsf{id}^{\bullet}_A &= \chi_\top^{-1} \circ \mathsf{e}_A \circ \mathsf{cod}_A \circ \mathsf{der}_A && \text{(definition)} \\
&= \chi_\top^{-1} \circ 0 \circ \mathsf{der}_A && \text{(constant rule of Definition 14)} \\
&= 0 && \text{(additivity)}
\end{aligned}$$

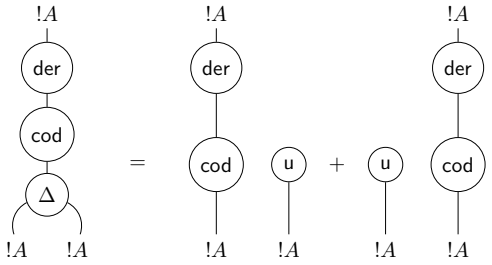
- *non-erasing:*

$$\begin{aligned}
\mathsf{id}^{\bullet}_A \circ \eta^{\mathcal{R}}_A &= \mathsf{cod}_A \circ \mathsf{der}_A \circ \mathsf{u}_A \circ \chi_\top && \text{(definition)} \\
&= \mathsf{cod}_A \circ 0 \circ \chi_\top && \text{([BCLS20, Lemma 2]}^{6}\text{)} \\
&= 0 && \text{(additivity)}
\end{aligned}$$

- *non-duplicable:*

$$\delta^{\mathcal{R}}_A \circ \mathsf{id}^{\bullet}_A = \chi^{-1}_{A,A} \circ \Delta_A \circ \mathsf{cod}_A \circ \mathsf{der}_A \qquad \text{(definition)}$$

and using string diagrams in $(\mathcal{C}, \otimes, I)$, we have:



by product rule (Definition 14, Figure 11) and additivity. Therefore,

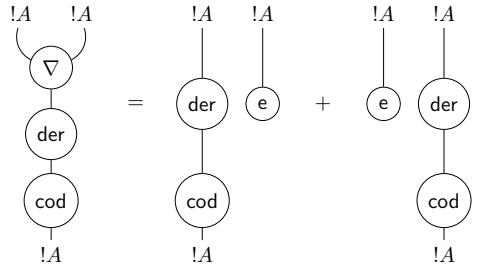
$$\delta^{\mathcal{R}}_A \circ \mathsf{id}^{\bullet}_A = (\mathsf{id}^{\bullet}_A \otimes_{\mathcal{R}} \mathsf{u}_A) + (\mathsf{u}_A \otimes_{\mathcal{R}} \mathsf{id}^{\bullet}_A)$$

again using Seely and the definition of $\mathsf{id}^{\bullet}$.

- *non-duplicative:*

$$\mathsf{id}^{\bullet}_A \circ \mu^{\mathcal{R}}_A = \mathsf{cod}_A \circ \mathsf{der}_A \circ \nabla_A \circ \chi A, A \qquad \text{(definition)}$$

which gives us, using string diagrams in $(\mathcal{C}, \otimes, I)$:



by compatibility of der and ∇ (Definition 12) and additivity; that is

$$\mathsf{id}^{\bullet}_A \circ \mu^{\mathcal{R}}_A = (\mathsf{id}^{\bullet}_A \otimes_{\mathcal{R}} \mathsf{e}_A) + (\mathsf{e}_A \otimes_{\mathcal{R}} \mathsf{id}^{\bullet}_A)$$

using Seely again and the definition of $\mathsf{id}^{\bullet}$.

□

Closed Structure. In general, a category $\mathcal{D}$ is closed if for any pair of objects A and B, $\mathcal{D}(A, B)$ can also be seen as an object of $\mathcal{D}$. In particular, for monoidal categories, $\mathcal{D}$ is *monoidal closed* if there exists $\multimap$ and an isomorphism natural in A, B, C such that:

$$\Lambda_{A,B,C} \colon \mathcal{D}(A \otimes B, C) \;\cong\; \mathcal{D}(A, B \multimap C)$$

[6] Actually $\mathsf{der}_A \circ \mathsf{u}_A = 0$ was part of the original definition of bialgebra modalities ([BCS06, Definition 4.8]), but it can be deduced from the other axioms and naturality of u and der ([BCLS20, Lemma 2]).

What happens if we consider $\mathcal{C}$ as in Definition 18 a monoidal *closed* category? Does the closed structure also transport to $\mathsf{Res}(\mathcal{C})$? Let us try to prove the isomorphism above for $\mathcal{R} = \mathsf{Res}(\mathcal{C})$. Everything seems to go smoothly for the first part:

$$\begin{aligned} \mathcal{R}(A \otimes_{\mathcal{R}} B, C) &= \mathcal{C}(!\,(A \& B)\,, !C) && \text{(definition)} \\ &\cong \mathcal{C}(!A \otimes !B, !C) && \text{(Seely isomorphism)} \\ &\cong \mathcal{C}(!A, !B \multimap !C) && \text{(closed structure of } \mathcal{C}) \end{aligned}$$

All that is left to do now is to define $\multimap_{\mathcal{R}}$ such that

$$\mathcal{R}(A, B \multimap_{\mathcal{R}} C) = \mathcal{C}(!A, !B \multimap !C)\,,$$

but that is where the difficulty lies: there seems to be no obvious way to define $\multimap_{\mathcal{R}}$ such that $!(B \multimap_{\mathcal{R}} C) \cong !B \multimap !C$. In particular, it is clear that $!(B \multimap C)$ and $!B \multimap !C$ are not necessarily isomorphic. Hence, the question of whether or not we can build a *closed* resource category from a closed differential category remains open – at least with the construction presented in this paper.

6 Conclusion

We show a general construction of resource categories from differential categories; however, certain key properties such as closure do not seem to be preserved by this construction. All in all, Theorem 1 formalizes the expected link between resource categories and differential categories: resource calculus is the finitary fragment of differential λ-calculus, and resource categories are to resource calculus what differential categories are to differential λ-calculus. Thus it is not surprising that we can build resource categories from differential categories.

Yet there is still much to study on resource categories. For instance, we did not tackle the subject of cartesian structure for a resource category in this paper. However, the subcategory of comonoid morphisms is cartesian – to what strategies do they correspond in pointer concurrent games? Besides, morphisms interpreting finite resource terms do not form a subcategory, because they lack identities – how can we best describe their structure? What about finite strategies in general?

Finally, resource categories were introduced to better understand the links between resource terms and strategies: we hope to generalize this correspondence to the Taylor expansion of λ-terms (introduced in [ER03, Section 6]), an operation which translates a λ-term with a possibly infinite behavior to a multiset of resource terms, representing all its linear approximations. The first step of this correspondence was fleshed out in [BCVA23], and we wish to further investigate this question in future works.

References

[AJM13] Samson Abramsky, Radha Jagadeesan et Pasquale Malacaria : Full abstraction for PCF. *CoRR*, abs/1311.6125, 2013.

[BC21] Lison Blondeau-Patissier et Pierre Clairambault : Positional injectivity for innocent strategies. *In* Naoki Kobayashi, éditeur : *6th International Conference on Formal Structures for Computation and Deduction, FSCD 2021, July 17-24, 2021, Buenos Aires, Argentina (Virtual Conference)*, volume 195 de *LIPIcs*, pages 17:1–17:22. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[BCL99] Gérard Boudol, Pierre-Louis Curien et Carolina Lavatelli : A semantics for lambda calculi with resources. *Mathematical Structures in Computer Science*, 9:437 – 482, 1999.

[BCLS20] Richard Blute, J. Robin B. Cockett, Jean-Simon Pacaud Lemay et Robert A. G. Seely : Differential categories revisited. *Appl. Categorical Struct.*, 28(2):171–235, 2020.

[BCS06] Richard Blute, J. Robin B. Cockett et Robert A. G. Seely : Differential categories. *Mathematical Structures in Computer Science*, 16:1049 – 1083, 2006.

[BCVA23] Lison Blondeau-Patissier, Pierre Clairambault et Lionel Vaux Auclair : Strategies as Resource Terms, and Their Categorical Semantics. *In* Marco Gaboardi et Femke van Raamsdonk, éditeurs : *8th International Conference on Formal Structures for Computation and Deduction (FSCD 2023)*, volume 260 de *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 13:1–13:22, Dagstuhl, Germany, 2023. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.

[Bou93] Gérard Boudol : The lambda-calculus with multiplicities. *In* Eike Best, éditeur : *CONCUR'93*, pages 1–6, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[CCRW17] Simon Castellan, Pierre Clairambault, Silvain Rideau et Glynn Winskel : Games and strategies as event structures. *Log. Methods Comput. Sci.*, 13(3), 2017.

[ER03] Thomas Ehrhard et Laurent Regnier : The differential lambda-calculus. *Theoretical Computer Science*, 309(1):1–41, 2003.

[Fio07] Marcelo P. Fiore : Differential structure in models of multiplicative biadditive intuitionistic linear logic. *In* Simona Ronchi Della Rocca, éditeur : *Typed Lambda Calculi and Applications*, pages 163–177, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[Gir87] Jean-Yves Girard : Linear logic. *Theoretical Computer Science*, 50(1):1–101, 1987.

[Gir88] Jean-Yves Girard : Normal functors, power series and λ-calculus. *Ann. Pure Appl. Log.*, 37:129–177, 1988.

[HO00] J. M. E. Hyland et C.-H. Luke Ong : On full abstraction for PCF: I, II, and III. *Inf. Comput.*, 163(2):285–408, 2000.

[JS91] André Joyal et Ross Street : The geometry of tensor calculus, i. *Advances in Mathematics*, 88:55–112, 1991.

[Mel06] Paul-André Melliès : Asynchronous games 2: The true concurrency of innocence. *Theor. Comput. Sci.*, 358(2-3):200–228, 2006.

[ML63] Saunders Mac Lane : Natural associativity and commutativity. *Rice Institute Pamphlet - Rice University Studies*, 49:28–46, 1963.

[ML71] Saunders Mac Lane : *Categories for the Working Mathematician*. Springer-Verlag, New York, 1971. Graduate Texts in Mathematics, Vol. 5.

[See89] Robert A. G. Seely : Linear logic, $*$-autonomous categories and cofree coalgebras. 1989.

[Sel10] Peter Selinger : A survey of graphical languages for monoidal categories. *In New Structures for Physics*, pages 289–355. Springer Berlin Heidelberg, 2010.

[TO16] Takeshi Tsukada et C.-H. Luke Ong : Plays as resource terms via non-idempotent intersection types. *In* Martin Grohe, Eric Koskinen et Natarajan Shankar, éditeurs : *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pages 237–246. ACM, 2016.