An Algebraic Extension of Intuitionistic Linear Logic: the $\mathcal{L}^{\mathcal{S}}_{!}$-Calculus and Its Categorical Model

Alejandro Díaz-Caro
Inria/LORIA, Nancy, France
UNQ, Bernal, Argentina
UdelaR, Colonia, Uruguay

Malena Ivnisky
UBA, Buenos Aires, Argentina
UdelaR, Montevideo, Uruguay

Octavio Malherbe
UdelaR, Montevideo, Uruguay

Abstract

We introduce the $\mathcal{L}^{\mathcal{S}}_{!}$-calculus, a linear lambda-calculus extended with scalar multiplication and term addition, that acts as a proof language for intuitionistic linear logic (ILL). These algebraic operations enable the direct expression of linearity at the syntactic level, a property not typically available in standard proof-term calculi. Building upon previous work, we develop the $\mathcal{L}^{\mathcal{S}}_{!}$-calculus as an extension of the $\mathcal{L}^{\mathcal{S}}$-calculus with the ! modality. We prove key meta-theoretical properties—subject reduction, confluence, strong normalisation, and an introduction property—as well as preserve the expressiveness of the original $\mathcal{L}^{\mathcal{S}}$-calculus, including the encoding of vectors and matrices, and the correspondence between proof-terms and linear functions. A denotational semantics is provided in the framework of linear categories with biproducts, ensuring a sound and adequate interpretation of the calculus. This work is part of a broader programme aiming to build a quantum programming language grounded in linear logic. This is an extended abstract of [17] (also available at arXiv:2504.12128).

1 Introduction

The design of proof languages plays a central role in both logic and programming languages, particularly when reasoning about resource usage and algebraic computation. Linear Logic [22] provides a framework for such reasoning, but its standard proof-term calculi do not make linear properties explicit at the syntactic level, such as distribution over addition or scalar multiplication. This lack of syntactic expressiveness limits its applicability in settings like quantum computing, where linearity is not just a property but a fundamental constraint.

Linear Logic is named as such because it is modelled by vector spaces and linear maps, and more generally by monoidal categories [5, 6, 21, 24]. These types of categories also include the so-called Cartesian categories, generating a formal place of interaction between purely algebraic structures and purely logical structures, i.e. between algebraic operations and the exponential connective "!". Functions between two propositions are linear functions. However, expressing this linearity within the proof-term language itself is challenging. Properties such as $f(u + v) = f(u) + f(v)$ and $f(a \cdot u) = a \cdot f(u)$, for some scalar a, require operations like addition and scalar multiplication, which are typically absent in the proof language.

In [13] this challenge has been addressed by introducing the $\mathcal{L}^{\mathcal{S}}$-calculus, a proof language for the strictly linear fragment of intuitionistic linear logic (IMALL) extended with syntactic constructs for addition (✚) and scalar multiplication (•). While the extension does not change provability, it enables the expression of linear properties at the term level, such as distributivity of functions over sums and scalar multiplication. For example, any proof-term t of $A \multimap B$ satisfies that $t\ (u \mathbin{✚} v)$ is observationally equivalent to $t\ u \mathbin{✚} t\ v$. Similarly, $t\ (a \bullet u)$ is equivalent to $a \bullet t\ u$.

t	$= x \mid t \mathbin{✚} u \mid a \bullet t$		
	$\mid a.\star$	$\mid \delta_{\mathbf{1}}(t, u)$	$(\mathbf{1})$
	$\mid \lambda x^A.t$	$\mid t\ u$	$(\multimap)$
	$\mid t \otimes u$	$\mid \delta_{\otimes}(t, x^A y^B.u)$	$(\otimes)$
	$\mid \langle\rangle$		$(\top)$
		$\mid \delta_{0}(t)$	(0)
	$\mid \langle t, u\rangle$	$\mid \delta^1_{\&}(t, x^A.u) \mid \delta^2_{\&}(t, x^B.u)$	$(\&)$
	$\mid inl(t) \mid inr(t)$	$\mid \delta_{\oplus}(t, x^A.u, y^B.v)$	$(\oplus)$
	$\mid !t$	$\mid \delta_{!}(t, x^A.u)$	$(!)$
	Introductions	Eliminations	Connective

Figure 1: The proof-terms of the $\mathcal{L}^{\mathcal{S}}_{!}$-calculus.

This extension involves changing the proof-term $\star$ of proposition $\mathbf{1}$ into a family of proof-terms $a.\star$, one for each scalar a in a given fixed commutative semiring $\mathcal{S}$:

$$\frac{}{\vdash a.\star : \mathbf{1}}\ \mathbf{1}_i(a)$$

The following two deduction rules have also been added:

$$\frac{\Gamma \vdash t : A \quad \Gamma \vdash u : A}{\Gamma \vdash t \mathbin{✚} u : A}\ \text{sum} \qquad \frac{\Gamma \vdash t : A}{\Gamma \vdash a \bullet t : A}\ \text{prod}(a)$$

Incorporating these rules requires adding commuting rules to preserve cut-elimination. Indeed, the new rules may appear between an introduction and an elimination of some connective. For example, consider the following derivation.

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \,\&\, B}\ \&_i}{\Gamma \vdash A \,\&\, B}\ \text{prod}(a) \qquad \Gamma, A \vdash C}{\Gamma \vdash C}\ \&_{e1}$$

To achieve cut-elimination, we must commute the rule prod(a) either with the introduction or with the elimination, as follows:

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash A}{\Gamma \vdash A}\ \text{prod}(a) \quad \dfrac{\Gamma \vdash B}{\Gamma \vdash B}\ \text{prod}(a)}{\Gamma \vdash A \,\&\, B}\ \&_i \qquad \Gamma, A \vdash C}{\Gamma \vdash C}\ \&_{e1}$$

$$\dfrac{\dfrac{\dfrac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \,\&\, B}\ \&_i \qquad \Gamma, A \vdash C}{\Gamma \vdash C}\ \&_{e1}}{\Gamma \vdash C}\ \text{prod}(a)$$

$$\frac{}{\Upsilon;x^A \vdash x : A}\ \text{lin-ax} \qquad \frac{}{\Upsilon, x^A; \varnothing \vdash x : A}\ \text{ax} \qquad \frac{\Upsilon;\Gamma \vdash t : A \quad \Upsilon;\Gamma \vdash u : A}{\Upsilon;\Gamma \vdash t + u : A}\ \text{sum} \qquad \frac{\Upsilon;\Gamma \vdash t : A}{\Upsilon;\Gamma \vdash a \bullet t : A}\ \text{prod}(a)$$

$$\frac{}{\Upsilon;\varnothing \vdash a.\star : 1}\ 1_i(a) \qquad \frac{\Upsilon;\Gamma \vdash t : 1 \quad \Upsilon;\Delta \vdash u : A}{\Upsilon;\Gamma,\Delta \vdash \delta_1(t,u) : A}\ 1_e \qquad \frac{\Upsilon;\Gamma, x^A \vdash t : B}{\Upsilon;\Gamma \vdash \lambda x^A.t : A \multimap B}\ \multimap_i \qquad \frac{\Upsilon;\Gamma \vdash t : A \multimap B \quad \Upsilon;\Delta \vdash u : A}{\Upsilon;\Gamma,\Delta \vdash t\ u : B}\ \multimap_e$$

$$\frac{\Upsilon;\Gamma \vdash t : A \quad \Upsilon;\Delta \vdash u : B}{\Upsilon;\Gamma,\Delta \vdash t \otimes u : A \otimes B}\ \otimes_i \qquad \frac{\Upsilon;\Gamma \vdash t : A \otimes B \quad \Upsilon;\Delta, x : A, y : B \vdash u : C}{\Upsilon;\Gamma,\Delta \vdash \delta_\otimes(t, x^A y^B.u) : C}\ \otimes_e \qquad \frac{}{\Upsilon;\Gamma \vdash \langle\rangle : \top}\ \top_i \qquad \frac{\Upsilon;\Gamma \vdash t : 0}{\Upsilon;\Gamma,\Delta \vdash \delta_0(t) : C}\ 0_e$$

$$\frac{\Upsilon;\Gamma \vdash t : A \quad \Upsilon;\Gamma \vdash u : B}{\Upsilon;\Gamma \vdash \langle t,u\rangle : A \,\&\, B}\ \&_i \qquad \frac{\Upsilon;\Gamma \vdash t : A \,\&\, B \quad \Upsilon;\Delta, x^A \vdash u : C}{\Upsilon;\Gamma,\Delta \vdash \delta^1_\&(t, x^A.u) : C}\ \&_{e1} \qquad \frac{\Upsilon;\Gamma \vdash t : A \,\&\, B \quad \Upsilon;\Delta, x^B \vdash u : C}{\Upsilon;\Gamma,\Delta \vdash \delta^2_\&(t, x^B.u) : C}\ \&_{e2}$$

$$\frac{\Upsilon;\Gamma \vdash t : A}{\Upsilon;\Gamma \vdash inl(t) : A \oplus B}\ \oplus_{i1} \qquad \frac{\Upsilon;\Gamma \vdash t : B}{\Upsilon;\Gamma \vdash inr(t) : A \oplus B}\ \oplus_{i2} \qquad \frac{\Upsilon;\Gamma \vdash t : A \oplus B \quad \Upsilon;\Delta, x^A \vdash u : C \quad \Upsilon;\Delta, y^B \vdash v : C}{\Upsilon;\Gamma,\Delta \vdash \delta_\oplus(t, x^A.u, y^B.v) : C}\ \oplus_e$$

$$\frac{\Upsilon;\varnothing \vdash t : A}{\Upsilon;\varnothing \vdash\ !t : !A}\ !_i \qquad \frac{\Upsilon;\Gamma \vdash t : !A \quad \Upsilon, x^A;\Delta \vdash u : B}{\Upsilon;\Gamma,\Delta \vdash \delta_!(t, x^A.u) : B}\ !_e$$

Figure 2: The deduction rules of the $\mathcal{L}^S_!$-calculus.

Both of these are reducible. We refer to the sum and prod(a) rules as *interstitial rules*, as they can appear in the interstice between an introduction and an elimination. We choose to commute these rules with the introductions as much as possible. This means we introduce the following commutation rule $a \bullet \langle t,u\rangle \longrightarrow \langle a \bullet t, a \bullet u\rangle$ instead of the alternative rule $\delta^1_\&(a \bullet t, x^A.u) \longrightarrow a \bullet \delta^1_\&(t, x^A.u)$. This choice provides a better introduction property, for example, a closed irreducible proof-term of a proposition $A \,\&\, B$ is a pair.

One of the objectives of our line of research is to extend a proof language of Intuitionistic Linear Logic with the necessary constructs to represent quantum algorithms. The $\mathcal{L}^S$-calculus [13] was the first step in this direction.

The $\mathcal{L}^S$-calculus has recently been extended [12] with a modality $\mathcal{B}$, which allows for a quantum denotational semantics that includes measurement. This modality is interpreted as a functor that maps a Hilbert space denoting pure states into a mixed state representation of completely positive maps, allowing the model to have the necessary projections that encode quantum measurement. This is a proof-language for the extended logic $\mathcal{B}$-IMALL, of which its theorems preserve IMALL provability when occurrences of $\mathcal{B}$ are erased.

In an early conference version of the current paper [14] we extended the $\mathcal{L}^S$-calculus to second-order intuitionistic linear logic, adding the exponential connective and a universal quantifier (but without defining a model for it), where we can encode natural numbers, iterators, and in general, classical computing, apart from vectors and matrices for quantum computing. We proved that the linearity result still holds. An early version of this proof-system was also presented as a QPL24 poster.

In the journal version [17] presented in this extended abstract, polymorphism has been removed to allow for a thorough study of the ILL fragment. We provide a denotational semantics for this fragment of the language, which we refer to as the $\mathcal{L}^S_!$-calculus.

Other related work. Our work draws inspiration from various domains, particularly quantum programming languages. These languages were trailblazers in merging programming constructs with algebraic operations, such as addition and scalar multiplication.

QML [1] introduced the concept of superposition of terms through the if$^\circ$ constructor, allowing the representation of linear combinations $a.u + b.v$ by the expression if$^\circ$ $a.|0\rangle + b.|1\rangle$ then u else v. The linearity (and even unitarity) properties of QML were established through a translation to quantum circuits.

The ZX calculus [10], a graphical language based on a categorical model, lacks direct syntax for addition or scalar multiplication but defines a framework where such constructs can be interpreted. This language is extended by the Many-Worlds Calculus [9] which allows for linear combinations of diagrams.

The algebraic lambda-calculus [25] and Lineal [4] exhibit syntax similarities with $\mathcal{L}^S$-calculus. However, the algebraic lambda-calculus lacks a proof of linearity in its simple intuitionistic type system. In contrast, Lineal is untyped and axiomatizes the linearity, relying on explicit definitions like $f(u+v) = f(u) + f(v)$ and $f(a.u) = a.f(u)$. Several type systems have been proposed for Lineal [2, 3, 15, 16, 19]. None of these systems are directly related to linear logic, and their purpose is not to prove linearity, as we do, but rather to axiomatize it. Nevertheless, there is an indirect relation for one of them: Lambda-S [15], a simply typed version of Lineal where non-duplicable terms are marked by a type modality S. A categorical model for Lambda-S was given in [18]. The model relies on an adjunction between a monoidal closed category and a Cartesian closed category. The language is interpreted in the Cartesian category, and the modality S is interpreted using the monad induced by the adjunction. This is the same adjunction used to model the ! modality in Linear Logic. However, Linear Logic is instead interpreted in the monoidal category, while the exponential modality is interpreted by the induced comonad.

2 The $\mathcal{L}^S_!$-calculus

The $\mathcal{L}^S_!$-logic is intuitionistic linear logic (we follow the presentation of DILL [8], extended with the additive linear connectives).

$$A = 1 \mid A \multimap A \mid A \otimes A \mid \top \mid 0 \mid A \,\&\, A \mid A \oplus A \mid\ !A$$

Notice that there is no $\bot$ nor $\parr$, as we are working with the intuitionistic fragment of linear logic.

$$\begin{aligned} [\![\mathbb{1}]\!] &= I & [\![\mathbb{0}]\!] &= \mathbf{0} \\ [\![A \multimap B]\!] &= [[\![A]\!] \to [\![B]\!]] & [\![A \,\&\, B]\!] &= [\![A]\!] \oplus [\![B]\!] \\ [\![A \otimes B]\!] &= [\![A]\!] \otimes [\![B]\!] & [\![A \oplus B]\!] &= [\![A]\!] \oplus [\![B]\!] \\ [\![\top]\!] &= \mathbf{0} & [\![!A]\!] &= ![\![A]\!] \end{aligned}$$

Figure 3: Interpretation of propositions. The functor ! is the monoidal comonad in the Linear category, and 0 represents the zero object.

Let $\mathcal{S}$ be a commutative semiring of *scalars*, for instance $\{\star\}$, $\mathbb{N}$, $\mathbb{Q}$, $\mathbb{R}$, or $\mathbb{C}$. The proof-terms of the $\mathcal{L}^{\mathcal{S}}_!$-calculus are given in Figure 1, where a is a scalar in $\mathcal{S}$.

A judgement has the form $\Upsilon; \Gamma \vdash t : A$, where Υ is the intuitionistic context and Γ the linear one. The deduction rules are those of Figure 2. These rules are exactly the deduction rules of intuitionistic linear natural deduction, with two differences: the addition of the rules sum and prod(a), and the change in the rule $\mathbb{1}_i(a)$, as discussed in the introduction.

3 Encoding of vectors and matrices

The set of vector types $\mathcal{V}$ is inductively defined as follows: $\mathbb{1} \in \mathcal{V}$, and if A and B are in $\mathcal{V}$, then so is $A \,\&\, B$. To each proposition $A \in \mathcal{V}$, we associate a positive natural number $d(A)$, which is the number of occurrences of the symbol $\mathbb{1}$ in A.

For each vector $\mathbf{u}$ in $\mathcal{S}^m$, we can define a term $\overline{\mathbf{u}}^A$, of type $A \in \mathcal{V}$ with $d(A) = m$.

Theorem 3.1 ([17, Theorem 2.20]). *Let $A, B \in \mathcal{V}$ with $d(A) = m$ and $d(B) = n$ and let M be a matrix with m columns and n lines, then there exists a closed proof-term t of $A \multimap B$ such that, for all vectors $\mathbf{u} \in \mathcal{S}^m$, $\underline{t\ \overline{\mathbf{u}}^A} = M\mathbf{u}$.* □

We also have the converse result: every closed proof-term of $A \multimap B$ induces a linear function.

Theorem 3.2 ([17, Corollary 2.33]). *Let $A, B \in \mathcal{V}$, such that $d(A) = m$ and $d(B) = n$, and f be a closed proof-term of $A \multimap B$. Then the function F from $\mathcal{S}^m$ to $\mathcal{S}^n$, defined as $F(\mathbf{u}) = \underline{f\ \overline{\mathbf{u}}^A}$, is linear.* □

4 Denotational semantics

Many of the additive properties required for our setting have been established in a recent paper [20], which develops a categorical semantics for the $\mathcal{L}\odot^{\mathcal{S}}$-calculus [13], an extension of the $\mathcal{L}^{\mathcal{S}}$-calculus incorporating the non-deterministic *sup* connective $\odot$. We adapt these constructions to incorporate the exponential connective !.

The model for the $\mathcal{L}^{\mathcal{S}}_!$-calculus is defined inside a Linear category [7] with biproduct $\oplus$ where there is a semiring monomorphism from $\mathcal{S}$ to $Hom(I, I)$, denoted by $(\!|\cdot|\!)$.

Example 4.1. The category $(\mathsf{SM}_{\mathcal{S}}, \otimes, \mathcal{S}, \oplus)$, where objects are semimodules over the semiring $\mathcal{S}$, arrows are semimodule homomorphisms, $\otimes$ is the tensor product and $\oplus$ is the biproduct. There is a monoidal adjunction $F : \mathsf{Set} \to \mathsf{SM}_{\mathcal{S}} \dashv G : \mathsf{SM}_{\mathcal{S}} \to \mathsf{Set}$ that induces a monoidal comonad FG on $\mathsf{SM}_{\mathcal{S}}$. This category is Linear, and the map $(\!|a|\!) : \mathcal{S} \to \mathcal{S}$ is defined as $b \mapsto a \cdot_{\mathcal{S}} b$.

The interpretation of propositions as objects inside this categories is given by Figure 3. In Figure 4 we give the interpretation for the interstitial rules, which make use of the semiadditive structure of the model.

Example 4.2. Let the semiring $\mathcal{S} = (\mathbb{N}, \cdot, +)$ be the natural numbers with standard multiplication and addition, and let $a, b \in \mathbb{N}$. The term $v = \langle a.\star, b.\star \rangle$ has type $\mathbb{1} \,\&\, \mathbb{1}$ in the empty context, corresponding to a vector type as defined in Section 3.

- In the category $\mathsf{SM}_{\mathbb{N}}$ (Example 4.1), the term v is interpreted by the following morphism:
$$[\![\langle a.\star, b.\star \rangle]\!] = n \mapsto (an, bn) : \mathbb{N} \to \mathbb{N} \times \mathbb{N}$$
- The scalar multiplication of v by $2 \in \mathbb{N}$ is interpreted as follows:
$$[\![2 \bullet \langle a.\star, b.\star \rangle]\!] = n \mapsto (2an, 2bn) : \mathbb{N} \to \mathbb{N} \times \mathbb{N}$$
- The interpretation of $v \mathbin{+\!\!+} v$ coincides with $[\![2 \bullet v]\!]$, as expected.
- The interpretation of v as a duplicable vector differs from that where each coordinate can be duplicated individually:
$$[\![!\langle a.\star, b.\star \rangle]\!] = n \mapsto n \cdot (a, b) : \mathbb{N} \to FG(\mathbb{N} \times \mathbb{N})$$
$$[\![\langle !a.\star, !b.\star \rangle]\!] = n \mapsto (n \cdot a, n \cdot b) : \mathbb{N} \to FG\mathbb{N} \times FG\mathbb{N}$$

Example 4.3. Let $\mathcal{S} = \mathbb{C}$, and let H be the term representing the Hadamard operator $\left(\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right)$, encoded as the term:

$$H = \lambda x^{\mathbb{1}\&\mathbb{1}}.\delta^1_{\&}(x, y^{\mathbb{1}}.\delta_{\mathbb{1}}(y, \langle 1.\star, 1.\star \rangle)) \mathbin{+\!\!+} \delta^2_{\&}(x, z^{\mathbb{1}}.\delta_{\mathbb{1}}(z, \langle 1.\star, (-1).\star \rangle))$$

This term is interpreted in the category $\mathsf{SM}_{\mathbb{C}}$ (Example 4.1) by the following morphism:

$$[\![H]\!] = x \mapsto ((y, z) \mapsto x(y + z, y - z)) : \mathbb{C} \to \mathsf{hom}(\mathbb{C} \times \mathbb{C}, \mathbb{C} \times \mathbb{C})$$

This is the linear map that represents the action of the Hadamard operator on a vector in $\mathbb{C}^2$.

5 Work in progress: polymorphism

We are currently extending this model to include both polymorphism and the exponential modality, as an interpretation for our earlier calculus [14]. We define a hyperdoctrine [11] to index the Linear categories which model the $\mathcal{L}^{\mathcal{S}}_!$-calculus, as an extension to work by Maneggia [23] on Linear hyperdoctrines.

$$\left[\!\!\left[\frac{\Upsilon;\Gamma \vdash t : A \quad \Upsilon;\Gamma \vdash u : A}{\Upsilon;\Gamma \vdash t \mathbin{+\!\!+} u : A}\ \text{sum} \right]\!\!\right] = [\![!\Upsilon]\!] \otimes [\![\Gamma]\!] \xrightarrow{\Delta} ([\![!\Upsilon]\!] \otimes [\![\Gamma]\!]) \oplus ([\![!\Upsilon]\!] \otimes [\![\Gamma]\!]) \xrightarrow{[\![t]\!] \oplus [\![u]\!]} [\![A]\!] \oplus [\![A]\!] \xrightarrow{\nabla} [\![A]\!]$$

$$\left[\!\!\left[\frac{\Upsilon;\Gamma \vdash t : A}{\Upsilon;\Gamma \vdash a \bullet t : A}\ \text{prod}(a) \right]\!\!\right] = [\![!\Upsilon]\!] \otimes [\![\Gamma]\!] \xrightarrow{[\![t]\!]} [\![A]\!] \xrightarrow{\rho^{-1}} [\![A]\!] \otimes I \xrightarrow{id \otimes (\!|a|\!)} [\![A]\!] \otimes I \xrightarrow{\rho} [\![A]\!]$$

Figure 4: Interpretation of the interstitial rules.

References

[1] T. Altenkirch and J. Grattage. 2005. A functional quantum programming language. In *Proceedings of LICS 2005*. 249–258.

[2] P. Arrighi and A. Díaz-Caro. 2012. A System F accounting for scalars. *Logical Methods in Computer Science* 8, 1:11 (2012).

[3] P. Arrighi, A. Díaz-Caro, and Benoît Valiron. 2017. The Vectorial Lambda-Calculus. *Information and Computation* 254, 1 (2017), 105–139.

[4] P. Arrighi and G. Dowek. 2017. Lineal: A linear-algebraic Lambda-calculus. *Logical Methods in Computer Science* 13, 1 (2017).

[5] Michael Barr. 1991. $*$-Autonomous categories and linear logic. *Mathematical Structures in Computer Science* 1, 2 (1991).

[6] Jean Bénabou. 1963. Catégories avec multiplication. *Comptes Rendus des Séances de l'Académie des Sciences* 256 (1963), 1887–1890.

[7] G.M. Bierman. 1994. *On intuitionistic linear logic.* Technical Report UCAM-CL-TR-346. University of Cambridge, Computer Laboratory. doi:10.48456/tr-346

[8] G.M. Bierman. 1996. *Dual Intuitionistic Linear Logic.* Technical Report ECS-LFCS-96-347. Laboratory for Foundations of Computer Science, University of Edinburgh. https://www.lfcs.inf.ed.ac.uk/reports/96/ECS-LFCS-96-347/

[9] Kostia Chardonnet, Marc de Visme, Benoît Valiron, and Renaud Vilamart. 2025. The Many-Worlds Calculus. Logical Methods in Computer Science, to appear.

[10] B. Coecke and A. Kissinger. 2017. *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning*. Cambridge University Press. doi:10.1017/9781316219317

[11] Roy L. Crole. 1993. *Categories for types.* Cambridge University Press.

[12] Kinnari Dave, Alejandro Díaz-Caro, and Vladimir Zamdzhiev. 2025. IMALL with a Mixed-State Modality: A Logical Approach to Quantum Computation. In *APLAS 2025: 23rd Asian Symposium on Programming Languages and Systems (Lecture Notes in Computer Science, Vol. 16201)*, A. Potanin (Ed.). Springer, 131–150.

[13] Alejandro Díaz-Caro and Gilles Dowek. 2024. A linear linear lambda-calculus. *Mathematical Structures in Computer Science* 34 (2024), 1103–1137. Issue 10.

[14] Alejandro Díaz-Caro, Gilles Dowek, Malena Ivnisky, and Octavio Malherbe. 2024. A linear proof language for second-order intuitionistic linear logic. In *Logic, Language, Information and Computation (Lecture Notes in Computer Science, Vol. 14672)*, George Metcalfe, Thomas Studer, and Ruy de Queiroz (Eds.). Springer, 18–35.

[15] A. Díaz-Caro, G. Dowek, and J.P. Rinaldi. 2019. Two linearities for quantum computing in the lambda calculus. *BioSystems* 186 (2019), 104012.

[16] A. Díaz-Caro, M. Guillermo, A. Miquel, and B. Valiron. 2019. Realizability in the Unitary Sphere. In *Proceedings of LICS 2019*. 1–13.

[17] Alejandro Díaz-Caro, Malena Ivnisky, and Octavio Malherbe. 2025. An Algebraic Extension of Intuitionistic Linear Logic: The L_{i}^{S}-Calculus and Its Categorical Model. *Journal of Logic and Computation* 35, 8 (2025), exaf053.

[18] Alejandro Díaz-Caro and Octavio Malherbe. 2020. A Categorical Construction for the Computational Definition of Vector Spaces. *Applied Categorical Structures* 28, 5 (2020), 807–844.

[19] Alejandro Díaz-Caro and Octavio Malherbe. 2022. Quantum control in the unitary sphere: Lambda-S_1 and its categorical model. *Logical Methods in Computer Science* 18, 3:32 (2022).

[20] Alejandro Díaz-Caro and Octavio Malherbe. 2026. The Sup Connective in IMALL: A Categorical Semantics. To appear in TCS. Draft at `arXiv:2205.02142`.

[21] Samuel Eilenberg and G. Max Kelly. 1966. Closed Categories. In *Proceedings of the Conference on Categorical Algebra*, S. Eilenberg, D. K. Harrison, S. MacLane, and H. Röhrl (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 421–562.

[22] Jean-Yves Girard. 1987. Linear Logic. *Theoretical Computer Science* 50 (1987), 1–102.

[23] P. Maneggia. 2004. *Models of linear polymorphism.* Ph. D. Dissertation. The University of Birmingham, UK.

[24] Robert Seely. 1989. Linear logic, $*$-autonomous categories and cofree coalgebras. In *Categories in Computer Science and Logic (Contemporary Mathematics)*, John W. Gray and Andre Scedrov (Eds.). American Mathematical Society.

[25] L. Vaux. 2009. The algebraic lambda calculus. *Mathematical Structures in Computer Science* 19, 5 (2009), 1029–1059.