From linear logic to quantum control

Alejandro Díaz-Caro
UNQ, UBA, and CONICET
Buenos Aires, Argentina

Gilles Dowek
Inria and ENS Paris-Saclay
Saclay, France

Octavio Malherbe
Universidad de la República
Montevideo, Uruguay

1 THE LOGIC OF QUANTUM PHYSICS

The logic of quantum physics is a topic that has been puzzling researchers for a long time. Starting from the seminal work by Birkhoff and von Neumann in the 30s [7], there have been many developments, with more or less success, to depict a logic that suits some odd behaviour seen in these theories. With the rise of quantum computing as a way to study and understand physics, many techniques from computer science have been used to solve physical problems. In particular, it is well-known that there is a correspondence between different type of logics, programming languages, and category theory: the Curry-Howard-Lambek correspondence (see, for example, [32, 41]). Therefore, in the quest for a quantum logic, this well-known technique can play a central role.

In the 2000s and early 2010s, the mainstream research on quantum programming languages centred from a practical perspective: the goal was to define programming languages to be able to program a quantum computer. With this aim, theoretical developments such as the "quantum lambda calculus" [40] have been pioneers. Many practical languages followed: Quipper [28] and QWIRE [35] are high-level functional languages based on the quantum lambda calculus, but also languages such as IBM's Qiskit [29] or Microsoft's Q# [42] are based on the same ideas. What these languages have in common is the paradigm so-called "quantum data, classical control" [39]. This paradigm is based on the original idea of architecture of Knill's QRAM [30]. In this scenario, a quantum computer is a device attached to a classical computer, which instructs the first one on what operations to do, over which registers, etc. The control flux of the program is purely classical, running on a classical computer. However, its connection to logic is an interesting one. Since quantum data cannot be copied, a classical computer cannot instruct a quantum computer to do whatever a classical computer can, such as copying data. This leads the aforementioned developments to the necessity of using Linear Logic-based type systems [26]. Although it gives a glance of what a quantum logic may look like, these developments are not attempting at defining a logic, but a programming language for the practical quantum computer.

In the quest for a quantum logic founded by computer science and the Curry-Howard-Lambek correspondence, there has been a long path in the paradigm of "quantum data and control" [2, 4, 5, 14, 17, 18, 21, 43]. This paradigm differs from the classical control in that the control flow of programs can follow a quantum particle, so, programs can be superposed, measured, etc. This idea, which seems radical, is basic in quantum computing: the Controlled-NOT (or CNOT) operation, for example, controls whether to apply a NOT operation based on a control qubit. There are more evolved examples such as the Quantum Switch [36, 37], which based on a control qubit can apply two operations in one order or another. Any connection between cut-elimination in Natural Deduction and quantum programs would require the use of quantum control if we wish to see some of these properties reflected. Indeed, in classical control, the qubits are part of the data in the quantum computer and not part of the control in the classical one. Thus, we claim that to define a quantum Curry-Howard-Lambek isomorphism, we need to consider control to be quantum.

If we want to start with an extension of the lambda calculus, we have to ensure that only linear functions are considered. One way to do so is to enforce linearity by definition. In this approach, $f(u+v)$ is, by definition, $f(u)+f(v)$ and $f(a.u)$ is, by definition, $a.f(u)$. Such has been the approach of the linear-algebraic lambda calculus [5], where a "call-by-base" strategy is defined, meaning that $f(u+v)$ reduces to $f(u)+f(v)$ and $f(a.u)$ reduces to $a.f(u)$. Another option, which we follow here since we are most interested in logic, is to enforce linearity with Linear Logic.

2 LINEAR LOGIC AND ITS LINEARITY

Linear Logic [26] is called "linear" since its models are linear in the algebraic sense: the mappings between two propositions are modelled by linear maps. However, within the proof languages of linear logic, this linearity is usually not expressible in its syntax. Indeed, the properties $f(u+v)=f(u)+f(v)$ and $f(a.u)=a.f(u)$ would require a syntactic sum and scalar multiplication.

This mismatch has been addressed by the $\mathcal{L}^{\mathcal{S}}$-calculus [16], a proof language for intuitionistic multiplicative additive linear logic (IMALL). On this calculus, two proof-terms of a proposition A can be added to generate a new proof-term of A, and a proof-term of a proposition A can be multiplied by a scalar from the semiring $\mathcal{S}$, giving a new proof-term of A. That is, the following trivially valid interstitial rules are considered

$$\frac{\Gamma\vdash A \quad \Gamma\vdash A}{\Gamma\vdash A}\ \text{sum} \qquad\qquad \frac{\Gamma\vdash A}{\Gamma\vdash A}\ \text{prod}(s)$$

with proof-terms $t \boldsymbol{+} u$ for the first, and $s \bullet t$ for the second, where t and u are proof-terms of A and s is a scalar in the given field $\mathcal{S}$.

Adding these rules permits building proofs that cannot be reduced because the introduction rule of some connective and its elimination rule are separated by an interstitial rule. For example,

$$\dfrac{\dfrac{\dfrac{\dfrac{\pi_1}{\Gamma\vdash A}}{\Gamma\vdash A\oplus B}\oplus\text{-i}_1 \quad \dfrac{\dfrac{\pi_2}{\Gamma\vdash A}}{\Gamma\vdash A\oplus B}\oplus\text{-i}_1}{\Gamma\vdash A\oplus B}\text{sum} \quad \dfrac{\pi_3}{\Gamma,A\vdash C} \quad \dfrac{\pi_4}{\Gamma,B\vdash C}}{\Gamma\vdash C}\oplus\text{-e}$$

Reducing such a proof, sometimes called a commuting cut, requires reduction rules to commute the rule sum either with the elimination rule below or with the introduction rules above.

The commutation with the introduction rules above is not always possible, for example in the proof

$$\dfrac{\dfrac{\dfrac{\pi_1}{\Delta_1\vdash A} \quad \dfrac{\pi_2}{\Delta_2\vdash B}}{\Gamma\vdash A\otimes B}\otimes\text{-i} \quad \dfrac{\dfrac{\pi_3}{\Delta_3\vdash A} \quad \dfrac{\pi_4}{\Delta_4\vdash B}}{\Gamma\vdash A\otimes B}\otimes\text{-i}}{\Gamma\vdash A\otimes B}\text{sum}$$

where $\Delta_1, \Delta_2 = \Gamma = \Delta_3, \Delta_4$, it is not. Thus, in these cases the commutation with the elimination rule below is preferred. In the $\mathcal{L}^{\mathcal{S}}$-calculus, the commutation of the interstitial rules with the introduction rules is chosen, rather than with the elimination rules, whenever it is possible, that is for all connectives except the disjunction and the multiplicative conjunction. For example, the proof

$$\dfrac{\dfrac{\dfrac{\pi_1}{\Gamma \vdash A} \quad \dfrac{\pi_2}{\Gamma \vdash B}}{\Gamma \vdash A \,\&\, B}\,\&\text{-i} \quad \dfrac{\dfrac{\pi_3}{\Gamma \vdash A} \quad \dfrac{\pi_4}{\Gamma \vdash B}}{\Gamma \vdash A \,\&\, B}\,\&\text{-i}}{\Gamma \vdash A \,\&\, B}\,\text{sum}$$

reduces to

$$\dfrac{\dfrac{\dfrac{\pi_1}{\Gamma \vdash A} \quad \dfrac{\pi_3}{\Gamma \vdash A}}{\Gamma \vdash A}\,\text{sum} \quad \dfrac{\dfrac{\pi_2}{\Gamma \vdash B} \quad \dfrac{\pi_4}{\Gamma \vdash B}}{\Gamma \vdash B}\,\text{sum}}{\Gamma \vdash A \,\&\, B}\,\&\text{-i}$$

Such a choice of commutation yields a stronger introduction property for the considered connective.

The proof-terms considering sums and scalar multiplication are reminiscent of other calculi used in similar ways for quantum computing and algebraic lambda-calculi [2–6, 17–22, 40, 44, 45].

In the same way as the rule prod(s) expresses a family of rules (one for each $s \in \mathcal{S}$), there are also as many proofs of $\mathbf{1}$ as elements in $\mathcal{S}$. So, we write $s.\star$, instead of just $\star$, the valid proofs of $\mathbf{1}$. Hence, $\mathbf{1}$ can be naturally interpreted as $\mathcal{S}$, and the proofs $\underline{v}$ of $\mathbf{1}^n = \bigwedge_{i=1}^n \mathbf{1}$ (for any parentheses) are in one-to-one correspondence with the elements v of $\mathcal{S}^n$.

In the $\mathcal{L}^{\mathcal{S}}$-calculus, any closed proof t of the proposition $\mathbf{1}^n \multimap \mathbf{1}^m$ can be proved to be linear in the syntactic sense. That is, if u and v are proofs of $\mathbf{1}^n$, then $t(u \mathbin{\boldsymbol{+}} v)$ is computationally equivalent to $tu \mathbin{\boldsymbol{+}} tv$ and $t(s \bullet u)$ is computationally equivalent to $s \bullet tu$. Moreover, any linear map $f : \mathcal{S}^n \longrightarrow \mathcal{S}^m$ has a representation in a proof-term $\vdash \overline{f} : \mathbf{1}^n \multimap \mathbf{1}^m$ such that for all $v \in \mathcal{S}^n$ we have that the proof-term $\underline{f(v)}$ is equivalent to the proof-term $\overline{f}\underline{v}$ (that is, the application of $\overline{f}$ to $\underline{v}$). This makes the calculus suitable to express matrices and vectors naturally, and thus, measurement-free quantum programs.

In [23] we give a categorical semantics for the $\mathcal{L}^{\mathcal{S}}$-calculus in a symmetric monoidal closed category with a monomorphism from the field of scalars (in fact, a semiring is enough for the calculus) to the semiring $\mathsf{Hom}(I, I)$. While linear logic has been pointed out as the logic of quantum computing, due to the no-cloning theorem, it is not enough to express all the possible quantum operations. For example, the quantum measurement is not a linear operation.

3 NON-DETERMINISM AS A LOGICAL CONNECTIVE

In [15] it is introduced a new connective to Natural Deduction, $\odot$ (read "sup" for superposition), to express the superposition of data and the measurement operation. This new connective arises from the following observation. A superposition behaves as a conjunction, where both propositions are true (and so, its proof is the pair of proofs), but also, when measured, it behaves as a disjunction, where only one proposition will be recovered in a non-deterministic process. The $\odot$-calculus contains, other than the sup connective, sums and scalar product, which allows encoding a basic quantum lambda calculus. This calculus shows that superposition and measurement can be represented as this new connective. In that paper, the question of no-cloning (already solved using linear logic), or unitarity of maps (already solved using some definition of orthogonality well-formedness rules [2, 18, 21]) have been left apart. In [16] we also show how to combine the $\odot$ connective with linear logic.

In [23] we focus on modelling this non-deterministic connective $\odot$ in the linear logic setting, however, we transformed into a probabilistic connective instead: We defined the $\mathcal{L}\odot^{\mathcal{S}\mathsf{p}}$-calculus, a probabilistic variant of the $\mathcal{L}^{\mathcal{S}}$-calculus, extended with the $\odot$ connective.

The $\mathcal{L}\odot^{\mathcal{S}\mathsf{p}}$-calculus differentiates itself from other approaches to non-deterministic and probabilistic calculus [8, 9, 12, 13], where the probabilities arise from terms like $t \oplus_p u$ with t and u of the same type. In contrast, our approach introduces probabilities through a probabilistic pair destructor: fst and snd serve as deterministic pair destructors, while $\delta_\odot^{\mathsf{pq}}$ is probabilistic. This way, $\delta_\odot^{\mathsf{pq}}(\langle t_1, t_2\rangle, x.s_1, y.s_2)$ can reduce to either $(t_1/x)s_1$ or $(t_2/x)s_2$ with probabilities p and q respectively. As a consequence, the probabilistic behaviour is an explicit elimination and is not triggered from an introduction term. It also allows for a probabilistic choice among elements of different types. In [15, 16] it is also shown how to transform this connective into a measurement-like operation.

The model is also suitable for a (generalised) probabilistic calculus, in the sense that instead of considering positive real scalars adding to one, we consider the elements of a set of weights, which are pairs $(p, q) \in \mathsf{Hom}(I, I) \times \mathsf{Hom}(I, I)$ such that $p + q = \mathsf{id}_I$. In the particular case of the category being that of semirings, with $\mathsf{Hom}(I, I) = \mathbb{R}^{\geq 0}$, it is instantiated in a proper probabilistic calculus.

A summary of the different calculi mentioned in this extended abstract can be found in Table 1.

4 MODELLING PROBABILITIES

Introducing a non-deterministic (or a generalised probabilistic) operator to a linear language is not straightforward, since adapting the Powerset Monad, typically used to express non-deterministic effects [33], is not easily applicable in every scenario. Our aim was to use the a monoidal category, as it is common in Linear Logic. To this introduction more intuitive, consider the concrete category $\mathbf{SM}_{\mathcal{S}}$ of semimodules over the commutative semiring $\mathcal{S}$, which is one of the concrete examples in [23]. The arrows in $\mathbf{SM}_{\mathcal{S}}$ are defined as the $\mathcal{S}$-homomorphisms, which is a challenge. The challenge arises from the fact that the mapping, which takes the two non-deterministic outputs of a computation and returns the set containing both, is not linear. Indeed, the mapping can be represented as $\nu : A \times A \longrightarrow \mathcal{P}A$, where $\nu(a_1, a_2) = \{a_1, a_2\}$. It can be easily verified that $\nu((a_1, a_2) \mathbin{\boldsymbol{+}_A} (a_1', a_2')) = \{a_1 \mathbin{\boldsymbol{+}_A} a_1', a_2 \mathbin{\boldsymbol{+}_A} a_2'\}$ while $\nu(a_1, a_2) \mathbin{\boldsymbol{+}_A} \nu(a_1', a_2') = \{a_1 \mathbin{\boldsymbol{+}_A} a_1', a_1 \mathbin{\boldsymbol{+}_A} a_2', a_2 \mathbin{\boldsymbol{+}_A} a_1', a_2 \mathbin{\boldsymbol{+}_A} a_2'\}$.

Another option to consider would be using lists instead of subsets, because the sum of lists is pointwise, much like the sum of pairs, which would address the issue. However, the pointwise sum of lists does not capture our conception of what a non-deterministic process should entail. To illustrate, suppose we have a program t that non-deterministically produces the numerical results n_1 and n_2, and another program u that non-deterministically yields m_1 and m_2. If sum represents a program that takes two arguments and

Calculus	Reference	Propositional logic	Linear Logic	Sup connective	Non-determinism vs weights
$\odot^{\mathcal{S}}$-calculus	[15]	✓		✓	Non-determinism
$\mathcal{L}^{\mathcal{S}}$-calculus	[16, 23]		✓		None
$\mathcal{L}\odot^{\mathcal{S}}$-calculus	[16]		✓	✓	Non-determinism
$\mathcal{L}\odot^{\mathcal{S}\mathrm{p}}$-calculus	[23]		✓	✓	Probabilities

Table 1: Calculi.

performs addition, we would anticipate that the potential outcomes of $\mathsf{sum}(t,u)$ encompass n_1+m_1, n_1+m_2, n_2+m_1, and n_2+m_2, rather than solely n_1+m_1 and n_2+m_2.

Our approach was instead inspired by the density matrix formalism (see, for example, [34, Section 2.4]), wherein we consider the linear combination of results as a representation of a probability distribution. In the particular case of a non-deterministic context instead of the probabilistic one, the temptation might be to use $t \boldsymbol{+}_A u$ to represent the non-deterministic outcomes t and u. Given that $\mathbf{SM}_{\mathcal{S}}$ has a biproduct, we can consider the following diagram, involving the diagonal $\Delta = \langle \mathrm{id}, \mathrm{id}\rangle$ and the codiagonal $\nabla = [\mathrm{id}, \mathrm{id}]$.

$$\begin{tikzcd}
& A \arrow[dl, "\mathrm{id}"'] \arrow[d, "\Delta"] \arrow[dr, "\mathrm{id}"] & \\
A \arrow[r, shift right, "i_1"'] \arrow[dr, "\mathrm{id}"'] & A \oplus A \arrow[l, shift right, "\pi_1"'] \arrow[r, shift left, "\pi_2"] \arrow[d, "\nabla"] & A \arrow[l, shift left, "i_2"] \arrow[dl, "\mathrm{id}"] \\
& A &
\end{tikzcd}$$

This way, instead of using $v : A \times A \longrightarrow \mathcal{P}A$ to aggregate the two results of a non-deterministic process, we can employ $\nabla : A \times A \longrightarrow A$, defined as $\nabla(a_1, a_2) = a_1 \boldsymbol{+}_A a_2$. However, this alternative gives rise to another problem. Consider a scenario where we have a program t returning t_1 or t_2 of type A non-deterministically, and another program u returning deterministically v of type B. We would expect that (t,u) returns both (t_1, v) and (t_2, v). Using ∇, it is $(t_1, v) \boldsymbol{+}_{A\times B} (t_2, v) = (t_1 \boldsymbol{+}_A t_2, v \boldsymbol{+}_B v)$. Nonetheless, if we first reduce t, we would obtain $(t_1 \boldsymbol{+}_A t_2, u)$ and then $(t_1 \boldsymbol{+}_A t_2, v)$, which is not equivalent to $(t_1 \boldsymbol{+}_A t_2, v \boldsymbol{+}_B v)$.

In fact, the approach inspired by density matrices would only work in the presence of probabilities. If, instead of reducing non-deterministically to t_1 and t_2, we have a probability p of yielding t_1 and a probability of q for t_2, with $p + q = 1$, considering $\nabla_{pq}(a_1, a_2) = p \bullet_A a_1 \boldsymbol{+}_A q \bullet_A a_2$, then the process (t,u) would return, employing ∇_{pq}, $p \bullet_{A\times B} (t_1, v) \boldsymbol{+}_{A\times B} q \bullet_{A\times B} (t_2, v) = (p \bullet_A t_1 \boldsymbol{+}_A q \bullet_A t_2, p \bullet_B v \boldsymbol{+}_B q \bullet_B v) = (p \bullet_A t_1 \boldsymbol{+}_A q \bullet_A t_2, v)$ thus solving the issue.

Therefore, let $\hat{p}$ be the mapping that multiplies its argument by p. We can consider ∇_{pq} to be defined as $[\hat{p}, \hat{q}]$. Similarly, we could define $\Delta_{pq} = \langle \hat{p}, \hat{q} \rangle$, resulting in the following diagram.

$$\begin{tikzcd}
& A \arrow[dl, "\hat{p}"'] \arrow[d, "\Delta_{pq}"] \arrow[dr, "\hat{q}"] & \\
A \arrow[r, shift right, "i_1"'] \arrow[dr, "\hat{p}"'] & A \oplus A \arrow[l, shift right, "\pi_1"'] \arrow[r, shift left, "\pi_2"] \arrow[d, "\nabla_{pq}"] & A \arrow[l, shift left, "i_2"] \arrow[dl, "\hat{q}"] \\
& A &
\end{tikzcd}$$

This is the approach we used. Each probabilities process in the $\mathcal{L}\odot^{\mathcal{S}\mathrm{p}}$-calculus have a pair (p,q) associated, from the set adding to 1, or, more generically, the set $\{(p,q) \mid (p,q) \in \mathsf{Hom}(I,I) \times \mathsf{Hom}(I,I), p + q = \mathrm{id}_I\}$. Thus, the category used, for any fixed semiring $\mathcal{S}$ used by the language, is a symmetric monoidal closed category with biproduct where there exists a monomorphism from the $\mathcal{S}$ to the semiring $\mathsf{Hom}(I,I)$. In the particular case of $\mathcal{S} = \mathbb{R}^{\geq 0}$, the calculus is a probabilistic calculus.

Some related works

The probabilistic choice in linear logic has been studied in many settings.

Compact closed categories. In [1], the authors proposed a categorical semantics of quantum protocols using symmetric monoidal closed categories with biproducts, which are also compact. The compactness property provides a notion of dagger, which gives a natural definition of measurements in terms of the *Born rule* in quantum mechanics. Thus, the main difference between our presentation for a model of IMALL+$\odot$ and their presentation for a model of quantum protocols is their reliance on a dagger operator and their use of the compactness property for this purpose. Some properties in our presentation would be significantly easier to prove if the category were compact closed (see [23, Remark 3.15]). However, assuming compactness would limit the generality of the results.

Probabilistic coherent spaces. In [11], based on an idea from Girard [27], the authors proposed a model of linear logic using probabilistic coherence spaces, interpreting types through continuous domains. Morphisms in the associated category are Scott-continuous. Additionally, they provide a probabilistic interpretation of terms, extending PCF with a probabilistic choice construction which selects a natural number from a probability distribution. They show the denotational semantics of closed terms in their base type as sub-probability distributions.

Cones. In [38], the author employed the concept of normed cones to provide an interpretation for the probabilities inherent in quantum programming. An abstract cone is analogous to an $\mathbb{R}$-vector space, except that scalars are drawn from the set of non-negative real numbers. This idea has been further developed in [24], and then proved to be a model of intuitionistic linear logic in [25]. In addition, it is proved [10] that this model is a conservative extension of the probabilistic coherent spaces.

Weighted relational models. In [31], the authors proposed a model of PCF$^{\mathcal{R}}$—that is, PCF with a probabilistic choice operator—based on the category of weighted relations. The first main difference with our approach is that they have a probabilistic choice operator, while we have a probabilistic pair destructor, as mentioned in the previous sections. The second difference is that they use a concrete model in the category of matrices over a continuous semiring, while we use an abstract categorical model. They also consider a fixed-point operator, which is outside the scope of our work.

REFERENCES

[1] S. Abramsky and B. Coecke. 2004. A categorical semantics of quantum protocols. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science, 2004.* 415–425.

[2] Thorsten Altenkirch and Jonathan Grattage. 2005. A functional quantum programming language. In *Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science (LICS 2005).* IEEE, 249–258.

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

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

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

[6] Ali Assaf, Alejandro Díaz-Caro, Simon Perdrix, Christine Tasson, and Benoît Valiron. 2014. Call-by-value, call-by-name and the vectorial behaviour of the algebraic λ-calculus. *Logical Methods in Computer Science* 10(4:8) (2014).

[7] Garrett Birkhoff and John von Neumann. 1936. The Logic of Quantum Mechanics. *Annals of Mathematics* 37, 4 (1936), 823–843.

[8] Gérard Boudol. 1994. Lambda-calculi for (strict) parallel functions. *Information and Computation* 108, 1 (1994), 51–127.

[9] Antonio Bucciarelli, Thomas Ehrhard, and Giulio Manzonetto. 2007. Not Enough Points Is Enough. In *21th International Workshop on Computer Science Logic (CSL 2007) (Lecture Notes in Computer Science, Vol. 4646).* Springer, 268–282.

[10] Raphaëlle Crubillé. 2018. Probabilistic Stable Functions on Discrete Cones are Power Series. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2018).* ACM, 275–284.

[11] Vincent Danos and Thomas Ehrhard. 2011. Probabilistic coherence spaces as a model of higher-order probabilistic computation. *Information and Computation* 209, 6 (2011), 966–991.

[12] Mariangiola Dezani-Ciancaglini, Ugo de'Liguoro, and Adolfo Piperno. 1996. Filter Models for Conjunctive-Disjunctive lambda-Calculi. *Theoretical Computer Science* 170, 1-2 (1996), 83–128.

[13] Mariangiola Dezani-Ciancaglini, Ugo de'Liguoro, and Adolfo Piperno. 1998. A filter model for concurrent lambda-calculus. *SIAM Journal of Compututing* 27, 5 (1998), 1376–1419.

[14] Alejandro Díaz-Caro. 2021. A quick overview on the quantum control approach to the lambda calculus. In *16th Workshop on Logical and Semantic Frameworks with Applications (LSFA 2021) (Electronic Proceedings in Theoretical Computer Science, Vol. 357).* Open Publishing Association, 1–17.

[15] Alejandro Díaz-Caro and Gilles Dowek. 2023. A new connective in natural deduction, and its application to quantum computing. *Theoretical Computer Science* 957 (2023), 113840.

[16] Alejandro Díaz-Caro and Gilles Dowek. 2024. A linear linear lambda-calculus. *Mathematical Structures in Computer Science* (to appear) (2024).

[17] Alejandro Díaz-Caro, Gilles Dowek, and Juan Pablo Rinaldi. 2019. Two linearities for quantum computing in the lambda calculus. *BioSystems* 186 (2019), 104012. Post-proceedings of TPNC 2017.

[18] Alejandro Díaz-Caro, Mauricio Guillermo, Alexandre Miquel, and Benoît Valiron. 2019. Realizability in the Unitary Sphere. In *Proceedings of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2019).* ACM, 1–13.

[19] Alejandro Díaz-Caro and Octavio Malherbe. 2019. A concrete categorical semantics for Lambda-S. In *Proceedings of the 13th Workshop on Logical and Semantic Frameworks with Applications (LSFA 2018) (Electronic Notes in Theoretical Computer Science, Vol. 344)*, Beniamino Accattoli and Carlos Olarte (Eds.). Elsevier, 83–100.

[20] 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.

[21] 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).

[22] Alejandro Díaz-Caro and Octavio Malherbe. 2023. A concrete model for a linear algebraic lambda calculus. *Mathematical Structures in Computer Science* 34, 1 (2023), 1–44.

[23] Alejandro Díaz-Caro and Octavio Malherbe. 2024. The Sup Connective in IMALL: A Categorical Semantics. Draft at `arXiv:2205.02142`.

[24] Thomas Ehrhard, Michele Pagani, and Christine Tasson. 2017. Measurable cones and stable, measurable functions: a model for probabilistic higher-order programming. In *Proceedings of the ACM on Programming Languages (POPL 2017)*, Vol. 2. ACM, 1–28.

[25] Thomas Ehrnard. 2020. Cones as a model of intuitionistic linear logic. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2020).* ACM, 370–383.

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

[27] Jean-Yves Girard. 2004. *Between Logic and Quantic: a Tract.* Cambridge University Press, Cambridge, 346–381.

[28] Alexander S. Green, Peter LeFanu Lumsdaine, Neil J. Ross, Peter Selinger, and Benoît Valiron. 2013. Quipper: a scalable quantum programming language. *ACM SIGPLAN Notices* 48, 6 (2013), 333–342.

[29] IBM Quantum. 2021. Qiskit. `https://quantum-computing.ibm.com/`.

[30] Emanuel Knill. 1996. *Conventions for Quantum Pseudocode.* Technical Report LAUR-96-2724. Los Alamos National Lab.

[31] Jim Laird, Giulio Manzonetto, Guy McCusker, and Michele Pagani. 2013. Weighted relational models of typed lambda-calculi. In *Proceedings of the 28rd Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2013).* ACM, 301–310.

[32] Joachim Lambek and Philip Scott. 1988. *Introduction to Higher-Order Categorical Logic.* Cambridge University Press.

[33] Eugenio Moggi. 1991. Notions of computation and monads. *Information and Computation* 93, 1 (1991), 55–92.

[34] Michael Nielsen and Isaac Chuang. 2010. *Quantum Computation and Quantum Information* (10th years anniversary ed.). Cambridge University Press, Cambridge, UK.

[35] Jennifer Paykin, Robert Rand, and Steve Zdancewic. 2017. QWIRE: a core language for quantum circuits. *ACM SIGPLAN Notices* 52, 1 (2017), 846–858.

[36] Lorenzo M. Procopio, Amir Moqanaki, Mateus Araújo, Fabio Costa, Irati Alonso Calafell, Emma G. Dowd, Deny R. Hamel, Lee A. Rozema, Časlav Brukner, and Philip Walther. 2015. Experimental superposition of orders of quantum gates. *Nature communications* 6 (2015), 7913.

[37] Giulia Rubino, Lee A. Rozema, Adrien Feix, Mateus Araújo, Jonas M. Zeuner, Lorenzo M. Procopio, Časlav Brukner, and Philip Walther. 2017. Experimental verification of an indefinite causal order. *Science advances* 3, 3 (2017), e1602589.

[38] Peter Selinger. 2004. Toward a semantics for higher-order quantum computation. In *2nd International Workshop on Quantum Programming Languages (QPL 2004) (TUCS General Publication, Vol. 33)*, Peter Selinger (Ed.). Turku Centre for Computer Science.

[39] Peter Selinger. 2004. Towards a quantum programming language. *Mathematical Structures in Computer Science* 14, 4 (2004), 527–586.

[40] Peter Selinger and Benoît Valiron. 2006. A lambda calculus for quantum computation with classical control. *Mathematical Structures in Computer Science* 16, 3 (2006), 527–552.

[41] Heine Sørensen and Paweł Urzyczyin. 2006. *Lectures on the Curry-Howard Isomorphism.* Studies in Logic and the Foundations of Mathematics, Vol. 149. Elsevier.

[42] Krysta Svore, Alan Geller, Matthias Troyer, John Azariah, Christopher Granade, Bettina Heim, Vadym Kliuchnikov, Mariia Mykhailova, Andres Paz, and Martin Roetteler. 2018. Q#: Enabling Scalable Quantum Computing and Development with a High-Level DSL. In *Real World Domain Specific Languages Workshop (RWDSL 2018) (ICPS Proceedings).* ACM, New York, USA, 7:1–7:10.

[43] André van Tonder. 2004. A Lambda Calculus for Quantum Computation. *SIAM Journal of Computing* 33, 5 (2004), 1109–1135.

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

[45] Margherita Zorzi. 2016. On quantum lambda calculi: a foundational perspective. *Mathematical Structures in Computer Science* 26, 7 (2016), 1107–1195.