Linear lambda-calculus is linear[*]

Alejandro Díaz-Caro[1] and Gilles Dowek[2]

[1] UNQ & ICC (CONICET–UBA), Argentina
[2] Inria & ENS Paris-Saclay, France

The name of linear logic [10] suggests that this logic has some relation with the algebraic notion of linearity. A common account of this relation is that a proof of a linear implication between two propositions A and B should not be any function mapping proofs of A to proofs of B, but a linear one. This idea has been fruitfully exploited to build models of linear logic (for example [3, 9, 11]), but it is seems difficult to even formulate it within the proof language itself. Indeed, expressing the properties $f(u + v) = f(u) + f(v)$ and $f(a.u) = a.f(u)$ requires an addition and a multiplication by a scalar, that are usually not present in proof languages.

The situation has changed with quantum programming languages [1,2,4,7,8,12,14] and with the algebraic λ-calculus [13], that mix some usual constructions of programming languages with algebraic operations. Several extensions of the lambda-calculus, or of a language of proof-terms, with addition and multiplication by a scalar have been proposed [2, 5, 13].

In this paper [6], we investigate an extension of linear logic with addition and multiplication by a scalar, the $\mathcal{L}^{\mathcal{S}}$-logic, and we prove a linearity theorem: if f is a proof of an implication between two propositions of some specific form, then $f(u+v) = f(u)+f(v)$ and $f(a.u) = a.f(u)$.

This work is part of a wider research program that aims at determining in which way propositional logic must be extended or restricted, so that its proof language becomes a quantum programming language. There are two main issues in the design of a quantum programming language: the first is to take into account the linearity of the unitary operators and, for instance, avoid cloning, and the second is to express the information-erasure, non-reversibility, and non-determinism of the measurement. In [5], we addressed the question of the measurement. In this paper [6], we address that of linearity.

Interstitial rules. To extend linear logic with addition and multiplication by a scalar, we proceed, like in [5, long version], in two steps: we first add interstitial rules and then scalars.

An interstitial rule is a deduction rule whose premises are identical to its conclusion. In the $\mathcal{L}^{\mathcal{S}}$-logic, we consider two such rules

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

These rules introduce constructors $\boldsymbol{+}$ and $\bullet$ in the proof language, that are used to express the addition and the multiplication by a scalar. The fact that these deduction rules are trivial expresses the fact that these operations are internal.

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

Scalars. We then consider a field $\mathcal{S}$ of scalars and replace the introduction rule of the connective $\top$ with a family of rules $\top$-i(a), one for each scalar, and the rule prod with a family of rules prod(a), also one for each scalar

$$\frac{}{\Gamma \vdash \top}\ \top\text{-i}(a) \qquad\qquad \frac{\Gamma \vdash A}{\Gamma \vdash A}\ \text{prod}(a)$$

[*]This is an extended abstract. The full paper can be found at [6]

The connective $\odot$. Besides interstitial rules and scalars, we have introduced, in [5, long version], a new connective $\odot$ (read "sup" for "superposition"), that has an introduction rule $\odot$-i similar to that of the conjunction, two elimination rules $\odot$-e1 and $\odot$-e2 similar to those of the conjunction, but also a third elimination rule $\odot$-e similar to that of the disjunction.

The elimination rules $\odot$-e1 and $\odot$-e2 are used to express the information-preserving, reversible, and deterministic operations, such as the unitary transformations of quantum computing. The elimination rule $\odot$-e is used to express the information-erasing, non-reversible, and non-deterministic operations, such as quantum measurement.

Starting from propositional logic with the interstitial rules sum and prod, we can thus either add scalars, or the connective $\odot$, or both. This yields the four logics: PL is propositional logic with the interstitial rules sum and prod, $\mathrm{PL}^{\mathcal{S}}$ is propositional logic with the interstitial rules and scalars, $\odot$ is propositional logic with the interstitial rules and the connective $\odot$, and $\odot^{\mathcal{S}}$ is propositional logic with the interstitial rules, the connective $\odot$, and scalars.

Linearity. The proof language of the $\odot^{\mathcal{S}}$-logic is a quantum programming language, as quantum algorithms can be expressed in it. However, this language addresses the question of quantum measurement, but not the that of linearity, and non-linear functions, such as cloning operators, can also be expressed in it. This leads to introduce, in our paper [6], a linear variant of the $\odot^{\mathcal{S}}$-logic, and prove a linearity theorem for it. More generally, we can introduce a linear variant for each of the previously mentioned four logics: $\mathcal{L}$ is linear logic with the interstitial rules sum and prod, $\mathcal{L}^{\mathcal{S}}$ is linear logic with the interstitial rules and scalars, $\mathcal{L}\odot$ is linear logic with the interstitial rules and the connective $\odot$, and $\mathcal{L}\odot^{\mathcal{S}}$ is linear logic with the interstitial rules, the connective $\odot$, and scalars.

Our goal is to prove a linearity theorem for the proof language of the $\mathcal{L}\odot^{\mathcal{S}}$-logic. But such a theorem does not hold for the full $\mathcal{L}\odot^{\mathcal{S}}$-logic, that contains the rule $\odot$-e, that enables to express measurement operators, which are not linear. Thus, our linearity theorem should concern the fragment of the $\mathcal{L}\odot^{\mathcal{S}}$-logic without this rule. But, if $\odot$-e rule is excluded, the connective $\odot$ is just the conjunction, and this fragment of the $\mathcal{L}\odot^{\mathcal{S}}$-logic is the $\mathcal{L}^{\mathcal{S}}$-logic.

So, for a greater generality, we prove our linearity theorem for the $\mathcal{L}^{\mathcal{S}}$-logic: linear logic with the interstitial rules and scalars, but without the $\odot$ connective, and discuss, at the end of the paper [6], how this result extends to the $\mathcal{L}\odot^{\mathcal{S}}$-logic.

Linear connectives. In the $\mathcal{L}^{\mathcal{S}}$-logic, we have to make a choice of connectives.

In intuitionistic linear logic, there is no multiplicative falsehood, no additive implication, and no multiplicative disjunction. Thus, we have two possible truths and two possible conjunctions, but only one possible falsehood, implication, and disjunction.

In the $\mathcal{L}^{\mathcal{S}}$-logic, we have chosen a multiplicative truth, an additive falsehood, a multiplicative implication, an additive conjunction, and an additive disjunction. The rule sum also is additive. The choice is not arbitrary, some variants do not exists in intuitionistic linear logic, others where necessary due to the interstitial rules.

Plan of the paper [6]. We first define the $\mathcal{L}^{\mathcal{S}}$-logic and its proof-language: the $\mathcal{L}^{\mathcal{S}}$-calculus, and prove that it verifies the subject reduction, confluence, termination, and introduction properties. We then show how the vectors of $\mathcal{S}^n$ can be expressed in this calculus and how the irreducible closed proofs of some propositions are equipped with a structure of vector space. We prove that all linear functions from $\mathcal{S}^m$ to $\mathcal{S}^n$ can be expressed as proofs of an implication between such propositions. We then prove the main result of this paper: that, conversely, all the proofs of implications between such propositions are linear. Finally, we show how this result extents to the proof language of the $\mathcal{L}\odot^{\mathcal{S}}$-logic and how this language is a quantum programming language.

References

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

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

[3] R. Blute. Hopf algebras and linear logic. *Mathematical Structures in Computer Science*, 6(2):189–217, 1996.

[4] B. Coecke and A. Kissinger. *Picturing Quantum Processes: A First Course in Quantum Theory and Diagrammatic Reasoning.* Cambridge University Press, 2017.

[5] A. Díaz-Caro and G. Dowek. A new connective in natural deduction, and its application to quantum computing. In A. Cerone and P. Csaba Ölveczky, editors, *Prooceedings of the International Colloquium on Theoretical Aspects of Computing*, volume 12819 of *Lecture Notes in Computer Science*, pages 175–193. Springer, 2021. Long version accessible at arXiv:2012.08994.

[6] A. Díaz-Caro and G. Dowek. Linear lambda-calculus is linear. In A. Felty, editor, *7th International Conference on Formal Structures for Computation and Deduction (FSCD 2022)*, volume 228 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 22:1–22:16. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2022.

[7] A. Díaz-Caro, M. Guillermo, A. Miquel, and B. Valiron. Realizability in the unitary sphere. In *Proceedings of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS 2019)*, pages 1–13, 2019.

[8] A. Díaz-Caro, G. Dowek, and J.P. Rinaldi. Two linearities for quantum computing in the lambda calculus. *Biosystems*, 2019.

[9] Th. Ehrhard. On Köthe sequence spaces and linear logic. *Mathematical Structures in Computer Science*, 12(5):579–623, 2002.

[10] J.-Y. Girard. Linear logic. *Theoreoretical Computer Science*, 50:1–102, 1987.

[11] J.-Y. Girard. Coherent banach spaces: A continuous denotational semantics. *Theoretical Computer Science*, 227(1-2):275–297, 1999.

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

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

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