Quantum Control in the Unitary Sphere: Lambda-$\mathcal{S}_1$ and its Categorical Model

Alejandro Díaz-Caro
ICC, CONICET–Universidad de Buenos Aires, Argentina
DCyT, Universidad Nacional de Quilmes, Argentina
adiazcaro@icc.fcen.uba.ar

Octavio Malherbe
DMA, CURE, Universidad de la República, Uruguay
IMERL, FIng, Universidad de la República, Uruguay
malherbe@fing.edu.uy

Short abstract. In a recent paper, a realizability technique has been used to give a semantics of a quantum lambda calculus. Such a technique gives rise to an infinite number of valid typing rules, without giving preference to any subset of those. In this paper, we introduce a valid subset of typing rules, defining an expressive enough quantum calculus. Then, we propose a categorical semantics for it. Such a semantics consists of an adjunction between the category of distributive-action spaces of value distributions (that is, linear combinations of values in the lambda calculus), and the category of sets of value distributions.

Extended abstract[1]

In quantum programming languages, the control flow of programs divides models in two classes. On the one hand, there is the model of the QRAM [12], or classical control [17]. The classical control refers to a scheme where the quantum operations are performed in a specialized device, known as QRAM, attached to a classical computer, which instructs the device which operations to apply over which qubits. In this model, the quantum operations are given by a series of "black boxes". An example of this is the quantum lambda calculus [18], as well as several high-level quantum programming languages such as Quipper [10] and QWIRE [16]. On the other hand, there is the model of quantum control, aiming at describing the "black boxes" explicitly. Under this scheme, Lineal [1, 2] is an untyped extension to the lambda calculus allowing for linear combinations of lambda terms. The main idea behind Lineal is that in order to describe the quantum control, it is needed to be able to superpose programs. This way, if s and t are two terms, so is its formal linear combination $\alpha \cdot s + \beta \cdot t$, with $\alpha, \beta \in \mathbb{C}$. Quantum programs can be expressed in Lineal, except for the quantum measurement, which is left out of the system. However, Lineal is not restricted to only quantum programs. In particular, to enforce the "quantumness", one would need to enforce that all the linear combinations have norm ℓ_2 equal to 1, and all the functions behave as isometries. One main feature of Lineal is the fact that all the functions, even if they are not forced to be linear or isometries, are treated linearly: if a function $\lambda x.s$ is applied to a formal linear combination $\alpha \cdot v + \beta \cdot w$, it distributes linearly as follows: $(\lambda x.s)(\alpha \cdot v + \beta \cdot w) \longrightarrow \alpha \cdot (\lambda x.s)v + \beta \cdot (\lambda x.s)w$.

A drawback in taking all functions as linear, is that adding measurement is not trivial, since a measurement is not a linear operation: If M is a measurement operator, $M(\alpha \cdot v + \beta \cdot w)$ does not behave as $\alpha \cdot Mv + \beta \cdot Mw$.

Lambda-$\mathcal{S}$ [4,5] is a typed lambda calculus based on Lineal, mainly focused on adding measurement to the calculus. Instead of treating all functions as linear, its types enforce linearity when the argument is a superposition, and allow for duplication when it is not. This is done by labeling superpositions

[1] The full paper can be found at arXiv:2012.05887.

Submitted to:
QPL 2021

with a modality $\mathcal{S}$. Any term typed by $\mathcal{S}$ is treated linearly, so only basis terms are duplicable. It is argued to be somehow the dual to Intuitionistic Linear Logic, where duplicable terms are marked (by a !). Indeed, in [7–9] a categorical model for Lambda-$\mathcal{S}$ has been proposed, obtained by a monoidal monad determined by a monoidal adjunction $(S,m) \dashv (U,n)$ and interpreting $\mathcal{S}$ as the monad US—exactly the opposite to the ! of linear logic, which in the literature is often interpreted as the comonad SU [14]. This implies that on the one hand there is a tight control of the Cartesian structure of the model, and on the other hand the world of superpositions lives inside the classical world, i.e. determined externally by classical rules until one decides to explore it. This is given by the following composition of maps: $USA \times USA \xrightarrow{n} U(SA \otimes SA) \xrightarrow{Um} US(A \times A)$, that allows us to operate in a monoidal structure explicitly allowing the algebraic manipulation and then to return to the Cartesian product. This is different from linear logic, where the ! stops any algebraic manipulation, i.e. $(!A) \otimes (!A)$ is a product inside a monoidal category. A concrete example is an adjunction between the categories Set of sets and Vec of vector spaces [7, 9].

In [6] another type system for Lineal has been worked out, this time, ensuring superpositions to be in the unitary sphere $\mathcal{S}_1$ (that is, norm-1 vectors), as it is required by quantum computing, also characterizing isometries via a specific type. This has been obtained by means of realizability techniques [11, 13, 15, 19]: Instead of deriving a computational meaning of proofs once the type system is set up, the idea of realizability is to consider the type system as a by-product of the operational semantics—programs are then potential realizers of types. For example, a program behaving as the identity will be a realizer of $A \to A$, regardless of its inner structure. Realizability is a powerful and modular framework amenable to many systems [3]. So, one particularity is that the typing rules are probable lemmas (any typing rule conforming the semantics, is a valid rule), hence, the rules are potentially infinite. On this scheme, there is a modality $\sharp$, with similar behavior to the $\mathcal{S}$ of Lambda-$\mathcal{S}$. The claimed main goal of this system has been to solve the long-standing issue of how to ensure norm-1 superpositions, and characterize unitary functions. This system does not include a measurement operator, which can be added in the future.

In this new paper, we extract a (finite) fixed type system following the realizability semantics [6], a calculus we call Lambda-$\mathcal{S}_1$, ensuring norm-1 superpositions. We prove its main correctness properties such as progress, subject reduction, and strong normalization, and we give a categorical model for this calculus. The developed model has some common grounds with the concrete model of Lambda-$\mathcal{S}$ [7, 9], however, the chosen categories this time are not Set and Vec, but categories that use the fact that values in our calculus form a distributive-action space (an algebraic structure similar to a vector space, where its additive structure is a semi-group). The two categories in the constructed adjunction are defined in terms of $\vec{\mathsf{V}}$, the set of values in the calculus, and their linear combinations (such a set forms a distributive-action space). Then, the categories are defined by:

- $\mathsf{Set}_{\vec{\mathsf{V}}}$: a category whose objects are the non-empty parts of $\vec{\mathsf{V}}$, and whose arrows are the arrows in Set that can be defined in Lambda-$\mathcal{S}_1$. This category also includes a product $\boxtimes$, which is the set of separable tensor products.
- $\mathsf{SVec}_{\vec{\mathsf{V}}}$: a category whose objects are the sub-distributive action spaces of $\vec{\mathsf{V}}$[2], and whose arrows are the linear maps which can be defined in Lambda-$\mathcal{S}_1$. It also includes a tensor product $\otimes$, which can also be defined as the span of the product $\boxtimes$.

Hence, the main novelty and contribution of this paper is presenting a model for quantum computing in the quantum control paradigm, which we show to be complete on qubits in the sense that if two

[2] V is a sub-distributive action space of W if $V \subseteq W$ and V forms a distributive-action space with the operations from W.

closed terms with qubit types are interpreted by the same arrows in the model, then those terms are computationally equivalent.

References

[1] Pablo Arrighi & Gilles Dowek (2008): *Linear-algebraic λ-calculus: higher-order, encodings, and confluence.* In: *RTA 2008*, *Lecture Notes in Computer Science* 5117, Springer, pp. 17–31.

[2] Pablo Arrighi & Gilles Dowek (2017): *Lineal: A linear-algebraic λ-calculus*. *Logical Methods in Computer Science* 13(1:8), pp. 1–33.

[3] Aloïs Brunel (2014): *The monitoring power of forcing transformations*. Ph.D. thesis, Université Paris 13, France.

[4] Alejandro Díaz-Caro & Gilles Dowek (2017): *Typing quantum superpositions and measurement*. In: *TPNC 2017*, *Lecture Notes in Computer Science* 10687, Springer, pp. 281–293.

[5] Alejandro Díaz-Caro, Gilles Dowek & Juan Pablo Rinaldi (2019): *Two linearities for quantum computing in the lambda calculus*. *BioSystems* 186, p. 104012. Postproceedings of TPNC 2017.

[6] Alejandro Díaz-Caro, Mauricio Guillermo, Alexandre Miquel & Benoît Valiron (2019): *Realizability in the Unitary Sphere*. In: *LICS 2019*, pp. 1–13.

[7] Alejandro Díaz-Caro & Octavio Malherbe (2019): *A concrete categorical semantics for Lambda-S*. In: *LSFA 2018*, *Electronic Notes in Theoretical Computer Science* 344, Elsevier, pp. 83–100.

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

[9] Alejandro Díaz-Caro & Octavio Malherbe (2020): *A concrete model for a linear algebraic lambda calculus*. Draft at `arXiv:1806.09236`.

[10] Alexander S. Green, Peter LeFanu Lumsdaine, Neil J. Ross, Peter Selinger & Benoît Valiron (2013): *Quipper: a scalable quantum programming language*. *ACM SIGPLAN Notices (PLDI'13)* 48(6), pp. 333–342.

[11] Stephen C. Kleene (1945): *On the Interpretation of Intuitionistic Number Theory*. *The Journal of Symbolic Logic* 10(4), pp. 109–124.

[12] Emanuel H. Knill (1996): *Conventions for quantum pseudocode*. Technical Report LA-UR-96-2724, Los Alamos National Lab.

[13] Jean-Louis Krivine (2009): *Realizability in classical logic*. *Panoramas et synthèses: Interactive models of computation and program behaviour* 27, pp. 197–229.

[14] Paul-André Melliès (2003): *Categorical models of linear logic revisited*. hal:00154229.

[15] Alexandre Miquel (2011): *A survey of classical realizability*. In: *TLCA 2011*, *Lecture Notes in Computer Science* 6690, pp. 1–2.

[16] Jennifer Paykin, Robert Rand & Steve Zdancewic (2017): *QWIRE: A Core Language for Quantum Circuits*. In: *POPL 2017*, ACM, pp. 846–858.

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

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

[19] Jaap van Oosten (2008): *Realizability. An Introduction to its Categorical Side*. *Studies in Logic and the Foundations of Mathematics* 152, Elsevier.