Sujet de stage M2 (2024)

Quantum Programming Language with Inductive Types

Advisors: Emmanuel Hainry and Romain Péchoux
E-mails: hainry@loria.fr and pechoux@loria.fr
Research Lab: Équipe MOCQUA, LORIA, Inria Nancy-Grand Est, Nancy

Context: Quantum computation is a computing paradigm using qubits as the primary information unit and that is physically based on quantum mechanics. The standard way of representing quantum programs is via quantum circuits [Deu89]. Quantum circuits however represent a fixed size computation and do not allow usual programming techniques, thus yielding a quest for a high level quantum programming language, for example [Sel04, GLR+13, DCM22]. One particularly interesting feature that is seeked for those languages is quantim control [DC21]. With high level languages also comes the need to manage the resources used by a program, in other words, controlling the complexity. This is even more true in quantum computing where the number of qubits in a quntum computer is severely limited, and their coherence is short-timed. The usual quantum complexity class, FBQP, consists in Functions computing with Bounded-error in Quantum Polynomial time.

In [DGMV19, DCM22] two λ-calculi are introduced to modelize the superposition of qubits while allowing quantum control and a type system guarantees that terms represent unitary transformations, hence correspond to legal quantum transformations. Dual Light Affine Logic, presented in [BT09], is a type system for ensuring polynomial time. Those have been used as inspiration to design a λ-calculus manipulating qubits and only allowing programs in FBQP [DCHPS].

This language does not contain a fixpoint on types (aka inductive types). It is possible to encode Church numerals or lists without this feature, but their definition is not linear. On the other hand, inductive types make it possible to encode Scott numerals or lists that are linear. It has been shown [BT10] that type fixpoint can be included in Dual Light Affine Logic without losing the FP-soundness.

Objectives: The goal is to design a type system equipped with a type fixpoint for a λ-calculus capturing FBQP. It will be crucial to use the type fixpoint to define lists of qubits whose orthogonality will perfectly match with the orthogonality on the tensor product of qubits.

References

[BT09] Patrick Baillot and Kazushige Terui. Light types for polynomial time computation in lambda calculus. *Inf. Comput.*, 207(1):41–62, 2009.

[BT10] Aloïs Brunel and Kazushige Terui. Church ⇒ Scott = Ptime: an application of resource sensitive realizability. In Patrick Baillot, editor, *DICE 2010*, volume 23 of *EPTCS*, pages 31–46, 2010.

[DC21] Alejandro Díaz-Caro. A quick overview on the quantum control approach to the lambda calculus. In Mauricio Ayala-Rincón and Eduardo Bonelli, editors, *LSFA'21*, volume 357 of *EPTCS*, pages 1–17, 2021.

[DCHPS] Alejandro Díaz-Caro, Emmanuel Hainry, Romain Péchoux, and Mário Silva. Punq: Polytime normalization and unitarity with quantum control.

[DCM22] Alejandro Díaz-Caro and Octavio Malherbe. Quantum control in the unitary sphere: Lambda-s1 and its categorical model. *Log. Methods Comput. Sci.*, 18(3):1–32, 2022.

[Deu89] David Elieser Deutsch. Quantum computational networks. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 425(1868):73–90, 1989.

[DGMV19] Alejandro Díaz-Caro, Mauricio Guillermo, Alexandre Miquel, and Benoît Valiron. Realizability in the unitary sphere. In *LICS 2019*, pages 1–13. IEEE, 2019.

[GLR+13] Alexander S. Green, Peter LeFanu Lumsdaine, Neil J. Ross, Peter Selinger, and Benoît Valiron. Quipper: a scalable quantum programming language. In *PLDI 2013*, pages 333–342. ACM, 2013.

[Sel04] Peter Selinger. Towards a quantum programming language. *Math. Struct. Comput. Sci.*, 14(4):527–586, 2004.