Formalization of a Programming Language for Quantum Circuits with Measurement and Classical Control - Abstract

Dong-Ho Lee[1,3], Sébastien Bardin[1], Valentin Perrelle[1], and Benoît Valiron[2,3]

[1]CEA LIST, Université Paris-Saclay, France
[2]CentraleSupélec, Université Paris-Saclay, France
[3]LRI, Université Paris-Saclay, France

Context Quantum circuits, introduced as a model of quantum computation, consist of gates and wires. The wire represents the qubit, the basic unit of quantum data which forms quantum state space, and the gates are unitary operators over the quantum state. In the circuit model, measurements are performed after the reversible operations, hence, they do not appear in the circuit [1]. On the other hand, in QRAM model, quantum computation is performed under the control of the classical computation [2]. The classical host constructs quantum circuits by classical computation and applies the circuit by the transfer of it to the quantum co-processor. The quantum co-processor executes the quantum circuit and return the measurement values to the host. This interaction between the host and the co-processor is illustrated in Figure 1.

Although quantum circuits and QRAM are equivalent, practical quantum computation is more likely to be based on the QRAM model. For this reason, many quantum programming languages and their semantics are based on QRAM [3, 4, 5, 6, 7, 8]. An interesting implication of this model is that the quantum circuit construction in classical host can be dependent on the measurement values from the quantum co-processor. This transfer of quantum data to host within circuit construction is called dynamic lift and used in languages like Quipper [4, 9] and QWire [3, 10].

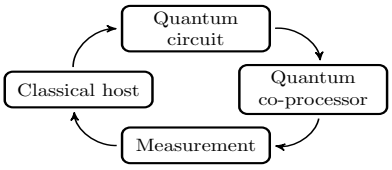

Figure 1: QRAM model

Problem However, the classical control over the circuit construction imposed by dynamic lift has not been explicitly formalized in the semantics of the quantum circuit languages. In the case of QWire, the operational semantics finds the normal form of the circuit but the normal form includes the lift followed by any program. Besides the operational semantics, quantum circuit languages have been formalized in various models as the denotational semantics based on density matrices [3] and categorical semantics [11, 12, 13, 14, 15]. However, these examples of formalization do not solve the problem in that they either ignore the structure of circuit or keep the term with dynamic lift abstract.

Goal Our goal in this paper is to find the model and formalize the semantics of dynamic lift which reveals the structure of the quantum circuit within the circuit construction. The problem rests in how to analyze the structure of the circuit without requiring the quantum co-processor to decide the value of measurement.

Contribution We solve the problem by making circuits not only lists but trees branching over the result of measurements, which makes them quantum channels. The semantics of dynamic lift, hence, can be formalized by the generation of control flows in the classical computation each of which is interpreted as a quantum channel. In this paper, we propose a small step operational semantics for a typed language extending quantum lambda calculus [16] with circuit operators (box and unbox) and circuit constants. The formalization relies on the semantics of Proto-Quipper [17] but circuits are generalized to quantum channels which allows us to formalize the semantics of measure operator (dynamic lift). Consequently, the language is extended with a notion of probabilistically branching term for the circuit construction in each control flow.

Discussion An advantage of having semantics in the quantum channel model is that it gives a new way to formalize circuit operators. For example, the reversibility of a term can be defined by the existence of a quantum circuit consisting only of unitary operators which is equivalent to the quantum channel representing the term. Still our programming language lacks important features of programming language like recursion and inductive data types and the linear type system imposes that both classical and quantum data is utilised exactly once. Our future goal is to extend the formalization of the language and use type system with useful properties to help the implementation of quantum algorithms.

References

[1] Peter Selinger. A brief survey of quantum programming languages. In *International Symposium on Functional and Logic Programming*, pages 1–6. Springer, 2004.

[2] Emmanuel Knill. Conventions for quantum pseudocode. Technical report, Los Alamos National Lab., NM (United States), 1996.

[3] Jennifer Paykin, Robert Rand, and Steve Zdancewic. Qwire: a core language for quantum circuits. In *ACM SIGPLAN Notices*, volume 52, pages 846–858. ACM, 2017.

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

[5] Dave Wecker and Krysta M Svore. Liqui|>: a software design architecture and domain-specific language for quantum computing. *arXiv preprint arXiv:1402.4467*, 2014.

[6] B Omer. Structured quantum programming. *Ph.D. Thesis, Technical University of Vienna*, 2003.

[7] Krysta M Svore, Alan Geller, Matthias Troyer, John Azariah, Christopher Granade, Bettina Heim, Vadym Kliuchnikov, Mariia Mykhailova, Andres Paz, and Martin Roetteler. Q#: Enabling scalable quantum computing and development with a high-level domain-specific language. *arXiv preprint arXiv:1803.00652*, 2018.

[8] Robert S Smith, Michael J Curtis, and William J Zeng. A practical quantum instruction set architecture. *arXiv preprint arXiv:1608.03355*, 2016.

[9] Linda Anticoli, Carla Piazza, Leonardo Taglialegne, and Paolo Zuliani. Towards quantum programs verification: from quipper circuits to qpmc. In *International Conference on Reversible Computation*, pages 213–219. Springer, 2016.

[10] Robert Rand, Jennifer Paykin, and Steve Zdancewic. Qwire practice: Formal verification of quantum circuits in coq. *arXiv preprint arXiv:1803.00699*, 2018.

[11] Peter Selinger. A survey of graphical languages for monoidal categories. In *New structures for physics*, pages 289–355. Springer, 2010.

[12] Francisco Rios and Peter Selinger. A categorical model for a quantum circuit description language. *arXiv preprint arXiv:1706.02630*, 2017.

[13] Bert Lindenhovius, Michael Mislove, and Vladimir Zamdzhiev. Enriching a linear/non-linear lambda calculus: A programming language for string diagrams. In *Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 659–668. ACM, 2018.

[14] Sam Staton. Algebraic effects, linearity, and quantum programming languages. *SIGPLAN Not.*, 50(1):395–406, January 2015.

[15] Mathys Rennela and Sam Staton. Classical control and quantum circuits in enriched category theory. *Electronic Notes in Theoretical Computer Science*, 336:257–279, 2018.

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

[17] Neil J Ross. Algebraic and logical methods in quantum computation. *PhD thesis, Dalhousie University*, 2015.