Quantum Programming Languages

WILLIAM MCINROY, Harvard University, USA

We provide a brief survey into the quantum computational model of "classically controlled quantum circuits", a generalization of the quantum circuit model of quantum computation, and quantum programming languages (QPLs) within this paradigm. Specifically, we focus on the categorical semantics of QPLs in the classically controlled quantum computing model, and particularly those semantics related to the teleportation protocol.

Additional Key Words and Phrases: Quantum Computing, Category Theory, Programming Languages

1 INTRODUCTION

There are many different models of quantum computation, one of the more popular being quantum circuits, used for instance in the open source framework Qiskit [Aleksandrowicz et al. 2019]. However, one problem with the quantum circuit paradigm for programming is the lack of *classical control*, or the modification of the quantum circuit based on classical computation. Of course, classical control can be simulated in the quantum circuit paradigm, for instance by the introduction of a quantum oracle of a classical function, but these often require an unwieldy number of gates and ancillary qubits to perform the computation [Fu et al. 2020].

Here, we study *classically controlled quantum computing* (CCQC), a paradigm arising from the teleportation protocol [Bennett et al. 1993], the QRAM machine [Knill 1996], and the logic-gate teleportation protocol [Gottesman and Chuang 1999], the later of which is shown to be a primitive for CCQC. In CCQC, the flow (any branchings) of the computation is handled by a classical computer, while processing without branches can be done either classically or quantumly. In comparison, quantum circuits do not describe control flow whereas quantum turing machines [Benioff 1982] quantumly describe control flow as well. It is widely thought that CCQC is the most practical paradigm for a real-life quantum computer [Knill 1996].

We are interested in studying *quantum programming languages* (QPLs) for CCQC, as such languages aim to provide both a useful syntax for conversion of theoretical quantum algorithms to runnable code, confirm that a described program is well-formed (does not violate classical or quantum laws) through type-checking, and potentially an easier conversion of classical algorithms to the quantum domain.

An interpretation of programming languages is the *compilation* of *syntax* (code) to *semantics* (a process of quantum and classical gates). A typical framework for studying the semantics of a programming language is within the context of *category theory*. The first formal treatment of CCQC in relation to QPLs is the language QPL [Selinger 2004], although QPL is notably low-level, the syntax captures only the primitives of bits, qubits, and primitve operations between the two, but lacks the typical abstractions found in classical programming languages. Within [Selinger and Valiron 2009], a quantum simply-typed λ-calculus is developed, allowing for some such abstractions, but specified no matching categorical semantics to compile this syntax to. These semantics were finally uncovered in full within [Malherbe et al. 2013], eventually leading to the Quipper family of languages [Fu et al. 2020, 2022a,b; Green et al. 2013].

We will not attempt to survey all of the categorical semantic conditions covered in [Malherbe et al. 2013; Selinger and Valiron 2009], instead we will focus on an example of the semantics necessary for the teleportation protocol, as in [Abramsky and Coecke 2004].

2 TELEPORTATION PROTOCOLS

We now summarize the teleportation protocol [Bennett et al. 1993] to motivate classically controlled quantum computing, and to eventually express the necessary primitives of a QPL. Typically, this protocols is presented with Alice and Bob characters. Here, we have instead described them in a manner to illustrate their relation to QPLs.

The *teleportation protocol* considers of three qubits, say on registers A, B, C, with the qubit in register A having arbitrary state $|\psi\rangle_A$ and both other registers set to $|0\rangle$. The goal of the protocol is to terminate computation with the B-register qubit in the state

A naive solution, based on the assumption we can reproduce the state $|\psi\rangle_A$ many times, is to measure $|\psi\rangle_A$ in the computation basis many times, store the results of the measurements, and prep the B-register to an approximation of $|\psi\rangle$, then perform whichever measurements we would like on $|\psi\rangle_A$.

However, the teleportation protocol does not assume that $|\psi\rangle$ may be reproduced many times, and requires only one iteration. It also importantly does not violate the no-cloning principle of a map $|\psi\rangle_A \otimes |0\rangle_B \not\to |\psi\rangle_A \otimes |\psi\rangle_B$. Instead, the teleportation protocol makes use of classical control.

Definition 2.1 (Teleportation Protocol)**.** The *teleportation protocol on* 1*-qubit* consists of the steps

(0) Start in the state $|\psi\rangle_A \otimes |00\rangle_{BC}$.

(1) Prepare the BC-registers into a Bell state $\frac{1}{\sqrt{2}}(|00\rangle_{BC} + |11\rangle_{BC})$.

(2) Measure the AC-registers in the Bell basis,

$$\begin{aligned} |b_{00}\rangle &= \frac{1}{\sqrt{2}}(|00\rangle_{AC} + |11\rangle_{AC}), \qquad |b_{01}\rangle = \frac{1}{\sqrt{2}}(|00\rangle_{AC} - |11\rangle_{AC}), \\ |b_{10}\rangle &= \frac{1}{\sqrt{2}}(|01\rangle_{AC} + |10\rangle_{AC}), \qquad |b_{11}\rangle = \frac{1}{\sqrt{2}}(|01\rangle_{AC} - |10\rangle_{AC}), \end{aligned} \tag{2.1}$$

while classically storing the measurement outcome o corresponding to $|b_o\rangle$.

(3) Based on the outcome o, apply one of the following operators to the B-register,

$$\begin{aligned} U_{00} &= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \qquad U_{01} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \\ U_{10} &= \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad U_{11} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \end{aligned} \tag{2.2}$$

(4) Now the B-register has state $|\psi\rangle_B$.

This protocol may be repeated for teleportation of multiple qubits.

End of Definition 2.1

We won't prove why it is the case that the teleportation protocol is capable of both measuring $|\psi\rangle_A$ while producing $|\psi\rangle_B$ [1]. Instead, the takeaway of this example is that an adjustment of the quantum circuit dependent on classical control flow is convenient in algorithms, but the fundamental differences of the two computing domains require programming languages to distinguish the domains.

[1] A measurement of b_o places the state $U_o|\psi\rangle$ in the B-register, which is undone by another application of U_o. This may be checked by the expansion of the joint ABC-qubit state in the computational basis.

3 SYNTAX AND SEMANTICS

We now provide an (at times informal) introduction to the view of programming languages as *compilation* of *syntax* to *semantics*. We note that several different types of semantics can be described by syntax: static (typing), dynamic (how to run the program), and denotational (a correspondence to mathematical objects, in our study these will be within *categories*).

We now recall some preliminaries in category theory. Note that these definitions are not fully rigorous, but are sufficient for our discussion of categorical semantics.

Quasidefinition 3.1. A *category* $\mathbf{C}$ is a collection of *objects* and for a collection of *morphisms* for each pair of objects X, Y, denoted $\mathbf{C}(X, Y)$, such that the composition of morphisms $X \to Y$ and $Y \to Z$ is a $X \to Z$ morphism, composition is associative, and there are identity morphisms $X \to X$.

End of Quasidefinition 3.1

Quasidefinition 3.2. Let $\mathbf{C}$ be a category with object Y. Then a *subobject* of Y is a morphism $i : X \to Y$ within $\mathbf{C}$ that is injective.

End of Quasidefinition 3.2

Quasidefinition 3.3. Let $\mathbf{C}$ be a category and X be an object $X \in \mathbf{C}$. Then the *slice category* $\mathbf{C}_{/X}$ *of* $\mathbf{C}$ *over* X consists of objects that are morphisms $(-) \to X$ in $\mathbf{C}$ and morphisms which commute with the objects.

The slice category $\mathbf{C}_{/X}$ is commonly depicted,

$$\mathbf{C}_{/X} = \left\{ \begin{array}{ccc} Y_1 & \xrightarrow{g} & Y_2 \\ & {\scriptstyle f_1}\searrow \quad \swarrow{\scriptstyle f_2} & \\ & X & \end{array} \right\}_{\substack{g:Y_1 \to Y_2 \\ f_1:Y_1 \to X \\ f_2:Y_2 \to X.}} \tag{3.1}$$

End of Quasidefinition 3.3

With these quasidefinitions, we may relate a typed λ-calculus to categorical semantics.

Quasidefinition 3.4. A λ-calculus describes a syntax through *terms*, consisting of

(i) *types* $X_{(-)}$ which each term is assigned,
(ii) *variables* representing free values x,
(iii) *values* c and *computations* f,

Term forming operations induce new terms,

(i) *binding* $t' = \lambda x.t$ to augment a term t with a free variable x,
(ii) *application* $t'' = (t\ t')$ to represent that the term t which was bound with a free variable is now evaluated on the term t'.

Note that the term forming rules are often subject to *type checking*, i.e. application/binding is compatible with term types.

End of Quasidefinition 3.4

Definition 3.1. Let $\mathbf{C}$ be a category. The *internal language of* $\mathbf{C}$ consists of

(i) *Types* given by the objects of $\mathbf{C}$.
(ii) Elements of an object $X \in \mathbf{C}$ are the *values* of type X. A morphism $X \to Y$ is a term of type Y with a *free variable* of type X.
(iii) The terms are given by are subobjects.
  (a) Every object X has a subobject $X \xrightarrow{\mathrm{id}_X} X$, so we may interpret objects as both terms and as types.
  (b) Morphisms $X \xrightarrow{f_1} Z$ and $Y \xrightarrow{f_2} Z$ satisfying with X is a subobject of $Y \in \mathbf{C}_{/Z}$ may be intrepeted as *the computation* f_1 *uses* f_2.

End of Definition 3.1

So a λ-calculus constant $c : X$ is an element of an object $c \in X \in \mathbf{C}$. A computation f of type Y bound to a free variable of type X is a morphism $(\lambda x.f) : X \to Y$. Application is evaluation $f(c)$ or composition $f \circ f'$ of morphisms. So a λ-calculus' categorical semantics are given by any category $\mathbf{C}$ with corresponding *categorical structure*. However, note that there are many choices of such categories, with different mathematical interpretations. In the following sections, we choose one such categorical structure with an interpretation that is similar to the quantum circuit model to develop our QPL for the teleportation protocol.

4 PRIMITIVES FROM TELEPORTATION

We now specify a fragment of a λ-calculus syntax to implement the teleportation protocol of Section 2, typical quantum circuit operations (for our cases, we'll assume arbitrary unitaries and computational base measurements), typical classical operations, and notably we must take care to not violate any quantum (or classical principles). This λ-calculus fragment is distilled from the work of [Selinger 2004; Selinger and Valiron 2009], and its categorical semantics are analyzed in line with [Abramsky and Coecke 2004].

4.1 Data Types

The teleportation protocol (Definition 2.1) on arbitrary qubits clearly requires any QPL implementing the protocol to have the types of bit and qubit (henceforth abbreviated qbit), as well as arbitrary combinations and functions between these types. In abstract syntax terms,

$$\text{Types} \qquad T := \quad \text{bit} \quad | \quad \text{qbit} \quad | \quad T_1 \times T_2 \quad | \quad T_1 + T_2. \tag{4.1}$$

To this end, we recall that qbit may be thought of a density matrix over $\mathbb{C} \otimes \mathbb{C}$, and by analogy classical bits may be thought of as a pair of two matrices over $\mathbb{C}$, i.e. $0 \mapsto (\mathrm{id}, 0)$ and $1 \mapsto (0, \mathrm{id})$. We can then encode all of the types of Equation 4.1 within the category $\mathbf{Vect}_{\mathbb{C}}$ of (finite dimensional) $\mathbb{C}$-vector spaces and $\mathbb{C}$-linear maps.

Precisely, we may represent a type as a list of (positive, power of 2) natural numbers $n_1, \ldots, n_m$ rorresponding to the vector space $\otimes_{i \leq m} \mathbb{C}^{n_1 \times n_1}$. Then bit $= (1, 1)$ and qbit $= (2)$. Then if $T_1 = (n_1, \ldots n_m)$ and $T_2 = (n'_1, \ldots, n'_{m'})$, we define $T_1 + T_2 = (n_1, \ldots, n_m, n'_1, \ldots, n'_{m'})$ and $T_1 \times T_2 = (n_1 n'_1, \ldots, n_1 n'_{m'}, \ldots n_m n'_{m'})$.

As every n_i is said to be a power of two, then we obtain a natural interpretation of a list (2^d) as a qu*d*its. Then concatenated lists may be interpreted as collections of (unentangled) bits and q*d*its, and we seem to have captured the "ground" types necessary for our QPL.

However, we have not yet discussed types $T_1 \to T_2$ nor the important concept of dual vector spaces, which will represent the primitive quantum and classical operations as well as measurements. To this end, we note that there is additional structure than $\otimes$ used to correspond lists on $\mathbf{Vect}_{\mathbb{C}}$, namely the presence of dual-vector spaces [2] and the identification $\{\text{maps } V_1 \to V_2\} \simeq V_1^* \otimes V_2$. Thus we may incorporate to our types of Equation 4.1 the type forming operation $T_1 \to T_2$ with categorical semantics of $T_1 \to T_2 = V_1^* \otimes V_2$,

$$\text{Types} \qquad T := \quad \text{bit} \quad | \quad \text{qbit} \quad | \quad T_1 \times T_2 \quad | \quad T_1 + T_2 \quad | \quad T_1 \to T_2. \tag{4.2}$$

So we have specified the types of our QPL necessary for the teleportation protocol. But we still haven't specified the primitive computations of our QPL, and we cannot inherit these from $\mathbf{Vect}_{\mathbb{C}}$ as there are many non-unitary or measurement morphisms, and thus non-unitary allowable terms in the syntax.

4.2 Primitive Computations

To remedy the issues described at the end of Section 4.1, we now describe a modified typing category $\mathfrak{C}$ constructed from $\mathbf{Vect}_{\mathbb{C}}$ to match the categorical semantics necessary for a QPL capable of performing the teleportation protocol.

Firstly, before specifying $\mathfrak{C}$, we recall the primitives we would like to define syntactically.

[2] General categories with these properties are known as $*$-*autonomous and compact closed categories*.

$$\begin{aligned}\text{Computations} \qquad f := \;& f_1 + f_2 \;\mid\; f_1 \times f_2 \\ \mid\; & (x : \mathsf{qbit})\; U \cdot x \to \mathsf{qbit} \\ \mid\; & (x : \mathsf{qbit})\; \mathsf{meas}\, x \to \mathsf{bit} + \mathsf{bit} \\ \mid\; & (x : \mathsf{bit})\; \mathsf{init}\, x \to \mathsf{qbit} \\ \mid\; & (x : \mathsf{bit})\; \mathsf{flip}\, x \to \mathsf{bit} \\ \mid\; & (x : \mathsf{bit} + \mathsf{bit})\; \mathsf{merge}\, x \to \mathsf{bit} \\ \mid\; & (x : \mathsf{bit})\; \mathsf{copy}\, x \to \mathsf{bit} + \mathsf{bit}.\end{aligned} \tag{4.3}$$

These computations simply express the types of computations we expect to be able to do with a CCQC. Now, we are prepared to define $\mathfrak{C}$. The objects are the $\mathbb{C}$-vector spaces of the form

$$\otimes_{i \le m} \mathbb{C}^{2^{d_i} \times 2^{d_i}} \qquad \text{and} \qquad \left(\otimes_{i \le m} \mathbb{C}^{2^{d_i} \times 2^{d_i}} \right)^* \otimes \left(\otimes_{i \le m'} \mathbb{C}^{2^{d'_i} \times 2^{d'_i}} \right), \tag{4.4}$$

for m, d_i, m', d'_i natural numbers. Note that these are all of the vector spaces that are constructible from the lists considered by Equation 4.2.

We will now define the generating set of morphisms of $\mathfrak{C}$ on single qbits, q$d$its, and bits, subject to the axioms of quantum mechanics. We note that the remaining generated morphisms of $\mathfrak{C}$ due to concatenation of lists generating these objects can be generated in a similar manner (via $\otimes$ or * on the morphisms themselves). We also use the shorthand $T^d = T \times T \times \cdots \times T$ and $d \cdot T = T + \cdots + T$ for d repetitions of each type (and note $\mathsf{qbit}^d = \mathsf{q}d\mathsf{it}$).

(i) The generating set of $\mathfrak{C}(\mathsf{qbit}^d, \mathsf{qbit}^d)$ are giveny conjugations of 2^d-unitary maps,

$$U : \mathbb{C}^{2^d} \to \mathbb{C}^{2^d}, \qquad \rho \mapsto U \rho U^\dagger. \tag{4.5}$$

(ii) The generating set of $\mathfrak{C}(\mathsf{qbit}^d, \mathsf{qbit}^{d'})$ for $d \neq d'$ is empty.

(iii) The generating set of $\mathfrak{C}(\mathsf{qbit}^d, d \cdot \mathsf{bit})$ is given by the 2^d different maps corresponding to the 2^d different measurement operators in the computational basis, and should be thought of inducing the *quantum non-determinism branching*.

(iv) The generating set of $\mathfrak{C}(\mathsf{qbit}^d, d' \cdot \mathsf{bit})$ for $d \neq d'$ is empty.

(v) The generating set of $\mathfrak{C}(d \cdot \mathsf{bit}, \mathsf{qbit}^{d'})$ is given by the 2^d different maps corresponding to the 2^d different initializations of a qu$d$it in the computational basis.

(vi) The set of $\mathfrak{C}(d \cdot \mathsf{bit}, d' \cdot \mathsf{bit})$ is $\mathbf{Vect}_{\mathbb{C}}((\mathbb{C}^{1\times 1})^{\otimes 2d}, (\mathbb{C}^{1\times 1})^{\otimes 2d'})$.

Then we obtain a semantic-syntactic correspondence

(1) (i) and (ii) correspond to $(x : \mathsf{qbit})\; U \cdot x \to \mathsf{qbit}$,
(2) (iii) and (iv) correspond to $(x : \mathsf{qbit})\; \mathsf{meas}\, x \to 2 \cdot \mathsf{bit}$,
(3) (v) corresponds to $(x : \mathsf{bit})\; \mathsf{init}\, x \to \mathsf{qbit}$,
(4) All $\mathbb{C}$-linear maps of (vi) correspond to the purely-bit computations.

We now have all of the necessary syntax and semantics to express the teleportation protocol. Notably, the protocol is of type $\mathsf{qbit}^3 \to 4 \cdot \mathsf{qbit}^3$, where the 4 indicates the 4 possibilities due to branching caused by the measurement and quantum non-determinism.

5 FURTHER READING: MERGING, ABSTRACTION, AND COMBINATION

5.1 Merging

One notable downside of the QPL described in Section 4 is the branching factor, there is no "combination" of qbit states. The language QPL [Selinger 2004] achieves such "branch merging" by incorporating trace operators into the language (as well as other primitives we have not mentioned here, and a choice of different categorical semantics than mentioned here). In [Abramsky and Coecke 2004], traces are also used to formally prove the Born rule, with the categorical semantics they derived, to achieve a similar result (although they do not "merge branches" explicitly). There are great technicalities associated to formally defining categorical semantics for "branch merging", as such syntax allows the danger of a programmer writing an unsound program that "merges" physical qubits, rather than qbits created as the result of branching, however in practice the dynamic semantics for these cases are not difficult to define.

5.2 Abstraction and Combination

The QPL of Section 4 is also as low-level as the original QPL, but our QPL is less abstractly categorical. Notably, there are no primitive types other than bit, qbit, and combinations thereof. We made this choice as our constructed $\mathfrak{C}$ is more approachable for those unacquainted with category theory[3], and its categorical semantics are friendly to those with familiar with the quantum circuit model of computation. However, the low-levelness of our QPL makes it even more painful syntactically to use in practice than QPL. Further, our QPL is not capable of performing loops, which is an easy syntactic (but difficult semantic – due to halting problems) extension.

Steps to make a less low-level QPL have been pursued. In the quantum λ-calculus of [Selinger and Valiron 2009], abstract ground types are allowed (with caveats associated to whether they are classical or quantum data types). The categorical semantics for these more abstract ground types were fully described until [Malherbe et al. 2013], but proved this quantum λ-calculus to be sound.

5.3 Conclusions

The classically controlled quantum computing model of quantum computing is crucially important to a physical implementation, rigor in quantum algorithms, and for the development of any QPL.

We have explored the canonical example of CCQC in the teleportation protocol (Section 2); introduced the syntactic and semantics formalisms of programming languages (Section 3); and applied these formalisms to describe the primitive types, computations, and abstractions for a QPL capable of implementing the teleportation protocol (Section 4). The categorical semantics $\mathfrak{C}$ of our QPL were chosen specifically for their familiarity to the quantum circuit model, at the cost of expressability. Future work may examine more abstract and intricate categorical semantics for greater expressability and abstraction.

ACKNOWLEDGMENTS

The author would like to thank Richard Allen for the guiding words of wisdom that helped shape this project: *"if you want to understand [quantum information], category theory is not the answer!"*

[3]Opposed to the impressive work of [Abramsky and Coecke 2004; Malherbe et al. 2013; Selinger and Valiron 2009] in describing the purely categorical properties necessary for a QPL.

REFERENCES

Samson Abramsky and Bob Coecke. 2004. A categorical semantics of quantum protocols. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science, 2004.* 415–425. https://doi.org/10.1109/LICS.2004.1319636

Gadi Aleksandrowicz, Thomas Alexander, Panagiotis Barkoutsos, Luciano Bello, Yael Ben-Haim, David Bucher, Francisco Jose Cabrera-Hernández, Jorge Carballo-Franquis, Adrian Chen, Chun-Fu Chen, and et al. 2019. Qiskit: An Open-source Framework for Quantum Computing. (Jan 2019). https://doi.org/10.5281/zenodo.2562111

Paul Benioff. 1982. Quantum mechanical Hamiltonian models of Turing machines. *Journal of Statistical Physics* 29 (11 1982), 515–546. https://doi.org/10.1007/BF01342185

Charles Bennett, Gilles Brassard, Claude Crépeau, Richard Jozsa, Asher Peres, and William Wootters. 1993. Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels. *Physical review letters* 70 (04 1993), 1895–1899. https://doi.org/10.1103/PhysRevLett.70.1895

Peng Fu, Kohei Kishida, Neil J. Ross, and Peter Selinger. 2020. A tutorial introduction to quantum circuit programming in dependently typed Proto-Quipper. *CoRR* abs/2005.08396 (2020). arXiv:2005.08396 https://arxiv.org/abs/2005.08396

Peng Fu, Kohei Kishida, Neil J. Ross, and Peter Selinger. 2022a. A biset-enriched categorical model for Proto-Quipper with dynamic lifting. https://doi.org/10.48550/ARXIV.2204.13039

Peng Fu, Kohei Kishida, Neil J. Ross, and Peter Selinger. 2022b. Proto-Quipper with dynamic lifting. https://doi.org/10.48550/ARXIV.2204.13041

Daniel Gottesman and Isaak Chuang. 1999. Quantum Teleportation is a Universal Computational Primitive. (08 1999).

Alexander Green, Peter LeFanu Lumsdaine, Neil J. Ross, Peter Selinger, and Benoît Valiron. 2013. Quipper: a Scalable Quantum Programming Language. In *Proceedings of the 34th Annual ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2013, Seattle (ACM SIGPLAN Notices, Vol. 48(6))*. 333–342. https://doi.org/10.1145/2499370.2462177

E Knill. 1996. Conventions for quantum pseudocode. (6 1996). https://doi.org/10.2172/366453

Octavio Malherbe, Philip Scott, and Peter Selinger. 2013. Presheaf Models of Quantum Computation: An Outline. (02 2013). https://doi.org/10.1007/978-3-642-38164-5_13

Peter Selinger. 2004. Towards a Quantum Programming Language. *Mathematical. Structures in Comp. Sci.* 14, 4 (aug 2004), 527–586. https://doi.org/10.1017/S0960129504004256

Peter Selinger and Benoît Valiron. 2009. *Quantum Lambda Calculus*. Cambridge University Press, 135–172. https://doi.org/10.1017/CBO9781139193313.005