A report on "Algebraic Effects, Linearity, and Quantum Programming Languages", by Sam Staton

Raphaël Monat

January 3, 2017

1 Introduction

Although quantum computers are not yet built, there has been a lot of work on quantum computations, discovering interesting circuits such as the quantum Fourier transform. However, describing quantum Fourier transform, or even Shor's algorithm using circuits is quite cumbersome, and quantum programming languages would be interesting to use. There has been a lot of work on quantum programming languages in the last decade, such as [Sel04]. It would be interesting to have a complete, formal theory describing the whole nature of quantum computations. Another point made by the author is that using this formalization, one can use general methods [AS13, KP12] to immediately get static analyses and normalization by evaluation. Previous work [DKP07, CD08] introduced formalisms to reason on quantum computations, but none of them were complete (although they had other interesting properties, such as interesting rewriting properties). Having a complete theory is really interesting, as it describes quantum computations in a standalone way. In "Algebraic Effects, Linearity, and Quantum Programming Languages" [Sta15], Sam Staton presents a new framework permitting to infer equality of quantum computations. This theory is complete. This work is much more general, and presents algebraic theories with linear parameters (here, linear means that a resource cannot be used more than once, so this is useful when working on qubits), but its main application is quantum computation.

To avoid some complications, the author first presents a theory using continuation-passing style rather than explicit control. This means that the result of a computation is transmitted to the next computation. In the theory presented below, two different kind of objects will be handled: qubits, and computations. There are three quantum operators, described intuitively as:

- `new`, taking one computation with one parameter in argument. Writing `new(a.x)` means that a new qubit a is allocated and given to the computation x.
- $\texttt{apply}_U$, taking as arguments n qubits and a computation having n parameters. $\texttt{apply}_U(\vec{a}, \vec{b}.x)$ stands for: apply the unitary U to the qubit vector $\vec{a}$, bind the result to $\vec{b}$ and give it to the computation x.

- `measure`, taking as arguments a qubit, and two computations (without parameters). `measure`(a, x, y) stands for: measure $a = \alpha|0\rangle + \beta|1\rangle$, choose to execute either computation x (with probability $|\alpha|^2$) or computation y (with probability $|\beta|^2$).

For example, $t = \texttt{new}(a.\texttt{apply}_{\texttt{Had}}(a, a.\texttt{measure}(a, x, y)))$ means: create a new qubit a, and execute $\texttt{apply}_{\texttt{Had}}(a, a.\texttt{measure}(a, x, y))$ (here, `Had` is the Hadamard matrix). Thus, t is equivalent to executing either x or y with equal probability.

2 Main contributions of the paper

In this paper, the author defines the notion of algebraic theory with linear parameters. He develops a theory of quantum computations, and shows that this theory is complete.

2.1 General framework

The general framework presented in this article is about algebraic theories with linear parameters. A theory is defined using a *signature*, rules to form terms and *axioms*. Informally, the signature defines operations in the theory, these operations being used in the terms. Axioms define equality between terms. Using a theory, we can then establish other identities.

In this framework, we distinguish two kinds of variables:

- Computation variables (usually denoted x, y, ...), representing functions. These functions can have arguments. If a function x has p arguments, we write $x : p$.
- Parameter variables (usually denoted a, b, ...). In the following, these variables denote qubits.

Remark. In the following, a function may require arguments, but I tried to avoid the use of the word parameter. The word *parameter* is only used to denote parameter variables, that is, qubits.

Definition 2.1. A signature is a set of operations. An operation O is defined by its arity $(p \mid m_1 \dots m_k)$. This means that operation O takes p parameter variables in argument, and k computation variables in argument. The i-th computation argument has m_i arguments itself. This is written $O : (p \mid m_1 \dots m_k)$.

Example 2.2. A simple example of operation is `new`: $(0 \mid 1)$. As seen in the introduction, we would like this operation to take no qubit, and give the newly created qubit to a computation. This computation has thus only one argument.

Now that we have defined the operations available in our theory, we need to define how to form terms. Later, well-formed terms will represent quantum computations. To define this, we need a context of things that we supposed defined. Actually, we will use two contexts, Γ for the computation variables, and Δ for the parameter variables. Then, writing $\Gamma \mid \Delta \vdash t$ means that the term t is well-formed under the contexts Γ and Δ. There are

theory-independent inference rules to define terms. The first rule defines that a computation variable needs to have all its parameter to defined to be well formed. The second rule states that parameter variables can be permuted in the context. The third rule states that an operation-based term is well defined when its arity is respected. In a fashion similar to the one of lambda-calculus, the notation $b_1 \dots b_{m_1}.t_1$ binds variables $b_1 \dots b_{m_1}$ to the computation variable t_1.

$$\frac{}{\Gamma, x : p, \Gamma' \mid a_1 \dots a_p \vdash x(a_1 \dots a_p)}$$

$$\frac{\Gamma \mid a_1 \dots a_p \vdash t}{\Gamma \mid a_{\sigma(1)} \dots a_{\sigma(p)} \vdash t}\ \sigma \text{ is a permutation of } \{1, \dots, p\}$$

$$\frac{\Gamma \mid \Delta, b_1 \dots b_{m_1} \vdash t_1 \qquad \dots \qquad \Gamma \mid \Delta, b_1 \dots b_{m_k} \vdash t_k}{\Gamma \mid \Delta, a_1 \dots a_p \vdash O(a_1, \dots, a_p, b_1 \dots b_{m_1}.t_1, \dots, b_1 \dots b_{m_k}.t_k)}\ O : (p \mid m_1 \dots m_k)$$

Example 2.3. Using operation `new`, and the rule to form operation-based terms, we get the following rule:

$$\frac{\Gamma \mid \Delta, a \vdash t}{\Gamma \mid \Delta \vdash \texttt{new}(a.t)}$$

There is a substitution rule not presented here. There are also weakening and contraction rules in the computation context:

$$\frac{\Gamma \mid \Delta \vdash t}{\Gamma, \Gamma' \mid \Delta \vdash t} \qquad \frac{\Gamma, \Gamma', \Gamma' \mid \Delta \vdash t}{\Gamma, \Gamma' \mid \Delta \vdash t}$$

The linearity, preventing a copy of qubits is expressed here by forbidding weakening and contraction in the parameter context Δ, as well as the destruction of the arguments when using an operation.

Definition 2.4. An axiom is just a pair of terms in the same context, written $\Gamma \mid \Delta \vdash t = u$.

2.2 Constructing quantum computations terms

There are three kind of operations in the case of quantum computations:

1. `new`: $(0 \mid 1)$
2. `measure`: $(1 \mid 0, 0)$
3. for every $2^n \times 2^n$ unitary matrix `U`, $\texttt{apply}_{\texttt{U}}$: $(n \mid n)$

Example 2.5. We can show that if x and y are two functions taking no parameters, then $\texttt{new}(a.\texttt{apply}_{\texttt{Had}}(a, b.\texttt{measure}(b, x, y)))$ is a well-formed term:

$$\dfrac{\dfrac{\dfrac{\overline{x:0,y:0 \mid - \vdash x} \qquad \overline{x:0,y:0 \mid - \vdash y}}{x:0,y:0 \mid b \vdash \texttt{measure}(b,x,y)}}{x:0,y:0 \mid a \vdash \texttt{apply}_{\texttt{Had}}(a, b.\texttt{measure}(b,x,y))}}{x:0,y:0 \mid - \vdash \texttt{new}(a.\texttt{apply}_{\texttt{Had}}(a, b.\texttt{measure}(b,x,y)))}$$

2.3 Axioms of our quantum theory

The author proposes the twelve following axioms:

Quantum not negates a measurement Let $\texttt{X} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$ be the quantum not gate. Then:

$$\texttt{apply}_{\texttt{X}}(a, a.\texttt{measure}(a,x,y)) = \texttt{measure}(a,y,x)$$

Quantum control and measurements Let $D(U,V)$ be $\begin{pmatrix} U & 0 \\ 0 & V \end{pmatrix}$:

$$\texttt{measure}(a, \texttt{apply}_{\texttt{U}}(\vec{b}, \vec{b}.x(\vec{b})), \texttt{apply}_{\texttt{V}}(\vec{b}, \vec{b}.y(\vec{b}))) = \texttt{apply}_{\texttt{D(U, V)}}((a,\vec{b}), (a,\vec{b}).\texttt{measure}(a, x(\vec{b}), y(\vec{b})))$$

Discarding after a rotation We introduce a discarding operator: $\texttt{discard}_p(a_1, \ldots, a_p, t)$ measures qubits $a_1, \ldots, a_p$ and continues as t whatever the outcomes of the measurements are. Let $\texttt{U}$ be a $2^n \times 2^n$ unitary:

$$\texttt{apply}_{\texttt{U}}(\vec{a}, \vec{a}.\texttt{discard}_n(\vec{a}, x)) = \texttt{discard}_n(\vec{a}, x)$$

We can notice that when $n = 0$, this axioms means that the global phase of a qubit is not important.

Allocation of qubit

$$\texttt{new}(a.\texttt{measure}(a,x,y)) = x$$

Unitary control of a new qubit

$$\texttt{new}(a.\texttt{apply}_{\texttt{D(U, V)}}((a,\vec{b}), (a,\vec{b}).x(a,\vec{b}))) = \texttt{apply}_{\texttt{U}}(\vec{b}, \vec{b}.\texttt{new}(a, x(a,\vec{b})))$$

Applying swap unitary Let $\texttt{swap} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ be the unitary matrix swapping two qubits.

$$\texttt{apply}_{\texttt{swap}}((a,b), (a,b).x(a,b)) = x(b,a)$$

Applying identity Let $\mathtt{I}$ be the identity unitary.

$$\mathtt{apply}_{\mathtt{I}}(\vec{a}, \vec{a}.x(\vec{a})) = x(\vec{a})$$

Applying a product of unitaries

$$\mathtt{apply}_{\mathtt{UV}}(\vec{a}, \vec{a}.x(\vec{a})) = \mathtt{apply}_{\mathtt{U}}(\vec{a}, \vec{a}.\mathtt{apply}_{\mathtt{V}}(\vec{a}, \vec{a}.x(\vec{a})))$$

Applying a Kronecker product of unitaries

$$\mathtt{apply}_{\mathtt{U}\otimes\mathtt{V}}((\vec{a}, \vec{b}), (\vec{a}, \vec{b}).x(\vec{a}, \vec{b})) = \mathtt{apply}_{\mathtt{U}}(\vec{a}, \vec{a}.\mathtt{apply}_{\mathtt{V}}(\vec{b}, \vec{b}.x(\vec{a}, \vec{b})))$$

Commutativity of measurements

$$\mathtt{measure}(a, \mathtt{measure}(b, u, v), \mathtt{measure}(b, x, y)) = \mathtt{measure}(b, \mathtt{measure}(a, u, x), \mathtt{measure}(a, v, y))$$

Commutativity of allocations

$$\mathtt{new}(a.\mathtt{new}(b.x(a, b))) = \mathtt{new}(b.\mathtt{new}(a.x(a, b)))$$

Commutativity of allocation and measurerement

$$\mathtt{new}(a.\mathtt{measure}(b, x(a), y(a))) = \mathtt{measure}(b, \mathtt{new}(a.x(a)), \mathtt{new}(a.y(a)))$$

Remark. In the article, the author also presents the axioms as equality of circuits. This presentation is more visual, but less formal.

2.4 Two simple examples

We can show that a rotation of $D(1, e^{i\theta})$ does not affect measurement:

$$\begin{aligned} & \mathtt{apply}_{D(1,e^{i\theta})}(a, a.\mathtt{measure}(a, x, y)) \\ = \ & \mathtt{measure}(a, \mathtt{apply}_{\mathtt{1}}(x), \mathtt{apply}_{e^{i\theta}}(y)) \\ = \ & \mathtt{measure}(a, x, y) \end{aligned}$$

We can also show that a circuit taking two qubits a and b, applying Hadamard matrices to both qubits, then a controlled not, then two Hadamard matrices is the same as applying a controlled not the other way around (although this is more a matrix computation). This

is usually called CNOT in the Hadamard basis.

$$
\begin{aligned}
&\texttt{apply}_{\texttt{Had}\otimes\texttt{Had}}((a,b),(a,b).\texttt{apply}_{\texttt{CNOT}}((a,b),(a,b).\texttt{apply}_{\texttt{Had}\otimes\texttt{Had}}((a,b),(a,b).x)))\\
=\ &\texttt{apply}_{\texttt{(Had}\otimes\texttt{Had) CNOT (Had}\otimes\texttt{Had)}}((a,b),(a,b).x)\\
=\ &\texttt{apply}_{\texttt{swap CNOT swap}}((a,b),(a,b).x)\\
=\ &\texttt{apply}_{\texttt{swap}}((a,b),(a,b).\texttt{apply}_{\texttt{CNOT swap}}((a,b),(a,b).x))\\
=\ &\texttt{apply}_{\texttt{CNOT swap}}((b,a),(a,b).x)\\
=\ &\texttt{apply}_{\texttt{CNOT}}((b,a),(a,b).\texttt{apply}_{\texttt{swap}}((a,b),(a,b).x))\\
=\ &\texttt{apply}_{\texttt{CNOT}}((b,a),(a,b).x)
\end{aligned}
$$

2.5 Main result: completeness theorem

There are other axiomatizations of quantum computation [DKP07, CD08], but this one is the first complete one. This means that every property of a quantum computation can be proved in the theory presented in the last section. To prove this result, the author shows an equivalence between its theory and C^*-algebras, which are a "well-established model of quantum mechanics". More precisely, the author exhibits a bijection between the terms of his theory and completely positive unital maps using a lot of category theory. The real advantage of this theory is that it does not require any knowledge of operator algebra to work on quantum computations, but only the axioms stated above.

2.6 From well-formed terms to a programming language

The theory presented above completely represents quantum computations. However, the result is always passed to a continuation and this is not really practical. We show how to derive a small, functional, typed, quantum programming language from the theory presented above. Programming language operators are underlined (measure), on the contrary to their counterparts from the algebraic theory.

As [Sel04] mentions, there are different kinds of quantum computations models, depending on if the control part of a quantum computation is classical or not. Here, the programming language is similar to the one presented in [Sel04] and uses classical control.

The types of this programming language are defined as: $A, B ::= \text{qubit} \mid I \mid A \otimes B \mid 0 \mid A + B$ (where $+$ is the usual sum type, so that $\text{bool} = I + I$). The typing rules are as follows:

$$\frac{}{x : A \vdash x : A} \qquad \frac{\Gamma \vdash t : A \otimes B \qquad \Delta, x : A, y : B \vdash u : C}{\Gamma, \Delta \vdash \text{let } (x, y) = t \text{ in } u : C}$$

$$\frac{}{\vdash \underline{\text{new}}() : \text{qubit}} \qquad \frac{\Gamma \vdash t : \text{qubit}^{\otimes n}}{\Gamma \vdash \underline{\text{apply}}_U(t) : \text{qubit}^{\otimes n}} \qquad \frac{\Gamma \vdash t : \text{qubit}}{\Gamma \vdash \underline{\text{measure}}(t) : \text{bool}}$$

Then, the second axiom establishes a correspondence between the two following commands:

- let $(a', x') = \underline{\text{apply}}_{\mathtt{D(U,\ V)}}(a, x)$ in $(\underline{\text{measure}}(a'), x')$,
- if $\underline{\text{measure}}(a) = 0$ then $(0, \underline{\text{apply}}_{\mathtt{U}}(x))$ else $(1, \underline{\text{apply}}_{\mathtt{V}}(x))$.

2.7 Extensions

By adding or removing at most one or two axioms, and changing the valuation of unitaries from complex numbers to $\{0, 1\}$, the author shows that he can also obtain theories for classical computations, classical local store, and QRAM.

3 Comments and conclusion

I found this article quite abstract and interesting to read, although (at least) the proofs were using a lot of category theory. I hope that this report is more accessible than the initial article. I read in [Sel04] that the semantics of a programming language with quantum control were not understood yet, but I think it would be interesting to see what has been done in this area, and if quantum control brings additional expressivity to the programming language.

References

[AS13] Danel Ahman and Sam Staton. Normalization by evaluation and algebraic effects. *Electronic Notes in Theoretical Computer Science*, 298:51–69, 2013.

[CD08] Bob Coecke and Ross Duncan. Interacting quantum observables. In *ICALP 2008 Proceedings Part II*, volume 5126 of *Lecture Notes in Computer Science*, pages 298–310. Springer, 2008.

[DKP07] Vincent Danos, Elham Kashefi, and Prakash Panangaden. The measurement calculus. *J. ACM*, 54(2), 2007.

[KP12] Ohad Kammar and Gordon D. Plotkin. Algebraic foundations for effect-dependent optimisations. In John Field and Michael Hicks, editors, *Proceedings of the 39th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2012, Philadelphia, Pennsylvania, USA, January 22-28, 2012*, pages 349–360. ACM, 2012.

[Sel04] Peter Selinger. Towards a quantum programming language. *Mathematical Structures in Computer Science*, 14(4):527–586, 2004.

[Sta15] Sam Staton. Algebraic effects, linearity, and quantum programming languages. In Sriram K. Rajamani and David Walker, editors, *Proceedings of the 42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2015, Mumbai, India, January 15-17, 2015*, pages 395–406. ACM, 2015.