QPL 2005 Preliminary Version

An Algebra of Pure Quantum Programming

Thorsten Altenkirch[1] Jonathan Grattage[1]

The University of Nottingham, UK

Juliana K. Vizzotto[2]

Federal University of Rio Grande do Sul, Brazil

Amr Sabry[3]

Indiana University, USA

Abstract

We develop a sound and complete equational theory for the functional quantum programming language QML. The soundness and completeness of the theory are with respect to the previously developed denotational semantics of QML. The completeness proof also gives rise to a normalisation algorithm following the *normalisation-by-evaluation* approach. The current work focuses on the pure fragment of QML, omitting measurements.

Key words: denotational semantics, completeness, normalisation

1 Introduction

The functional quantum language QML was recently introduced by Altenkirch and Grattage [AG05]. The semantics of QML is inspired by the denotational semantics of classical reversible computations; it provides a foundation for reasoning about quantum programs by mapping them to their denotations.

The next natural step is to develop reasoning principles on QML programs themselves, which avoid the detour via the denotational semantics. For example, consider the following QML definition of the Hadamard gate:

$$
\begin{array}{l}
H\ x = \mathbf{if}^{\circ}\ x \\
\qquad\quad \mathbf{then}\ (\mathit{false} + (-1) * \mathit{true}) \\
\qquad\quad \mathbf{else}\ \ (\mathit{false} + \mathit{true})
\end{array}
$$

[1] Email: {txa,jjg}@cs.nottingham.ac.uk

[2] Email: jkv@inf.ufrgs.br

[3] Email: sabry@indiana.edu

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

We would like to verify that $H\ (H\ x)$ is observationally equivalent to x, using a derivation such as:

$$
\begin{array}{rl}
H\ (H\ x) = & \mathbf{if}^{\circ}\ (\mathbf{if}^{\circ}\ x \\
 & \qquad \mathbf{then}\ (\mathit{false} + (-1) * \mathit{true}) \\
 & \qquad \mathbf{else}\ \ (\mathit{false} + \mathit{true})) \\
 & \mathbf{then}\ (\mathit{false} + (-1) * \mathit{true}) \\
 & \mathbf{else}\ \ (\mathit{false} + \mathit{true}) \\
 & \quad \text{-- by commuting conversion for } \mathbf{if}^{\circ} \\
= & \mathbf{if}^{\circ}\ x \\
 & \mathbf{then}\ \mathbf{if}^{\circ}\ (\mathit{false} + (-1) * \mathit{true}) \\
 & \qquad \mathbf{then}\ (\mathit{false} + (-1) * \mathit{true}) \\
 & \qquad \mathbf{else}\ \ (\mathit{false} + \mathit{true}) \\
 & \mathbf{else}\ \ \mathbf{if}^{\circ}\ (\mathit{false} + \mathit{true}) \\
 & \qquad \mathbf{then}\ (\mathit{false} + (-1) * \mathit{true}) \\
 & \qquad \mathbf{else}\ \ (\mathit{false} + \mathit{true}) \\
 & \quad \text{-- by } \mathbf{if}^{\circ} \\
= & \mathbf{if}^{\circ}\ x \\
 & \mathbf{then}\ (\mathit{false} - \mathit{false} + \mathit{true} + \mathit{true}) \\
 & \mathbf{else}\ \ (\mathit{false} + \mathit{false} + \mathit{true} - \mathit{true}) \\
 & \quad \text{-- by simplification and normalisation} \\
= & \mathbf{if}^{\circ}\ x\ \mathbf{then}\ \mathit{true}\ \mathbf{else}\ \mathit{false} \\
 & \quad \text{-- by } \eta\text{-rule for } \mathbf{if}^{\circ} \\
= & x
\end{array}
$$

It is relatively easy to develop *some* set of sound equational principles. Inspired by equivalences on classical computations, one may hypothesise that certain equations should hold, and simply verify that both sides of the equation have the same denotation.

However, as QML is based on a first-order functional language with finite types, it should be possible to also develop a *complete* set of equivalences that totally capture denotational equivalence. Technically, one can prove completeness of the equational semantics by "inverting" the denotational meaning-function. The construction is subtle in parts. We present it firstly in the context of the classical sub-language of QML, and then extend it to deal with quantum data and control.

The paper is thus organised as follows. We begin with a quick survey of related work followed by an informal review of QML in Section 3. In Section 4, we present the denotational semantics of the classical sub-language of QML, and present a system of equations which is sound with respect to the denotational semantics. We then show that this set of equations is complete in Section 5. Section 6 repeats the development for the quantum constructs. Section 7 concludes.

2 Related work

Selinger's influential paper [Sel04] introduces a single-assignment (essentially functional) quantum programming language, which is based on the separation of *classical control* and *quantum data*. This language combines high-level classical structures with operations on quantum data, and has a clear mathematical semantics in the form of superoperators. Quantum data can be manipulated by using unitary operators or by measurement, which can effect the classical control flow.

Recently, Selinger and Valiron [SV05] have presented a functional language based on the same *classical control* and *quantum data* paradigm. Selinger and Valiron's approach is in some sense complementary to ours: they use an affine type system (no contraction), while we use a strict system (no weakening). The lack of contraction is justified by the no-cloning property of quantum states. However, this does not apply to our approach, since we model contraction not by copying but by sharing — this idea is also present in the calculus of Arrighi and Dowek [AD04].

Van Tonder [vT04,vT03] has proposed a quantum λ-calculus incorporating higher-order programs, but no measurements. He suggests an equational theory for strict (higher-order) computations, but shows neither completeness nor normalisation.

3 QML Syntax and Examples

The QML terms consist of those of a first-order functional language, extended with quantum data and quantum control. The full language also includes quantum measurement, which we do not consider in this paper. The syntax of terms is as follows:

$$
\begin{array}{llcl}
(\mathit{Variables}) & x, y, \ldots & \in & \mathit{Vars} \\
(\mathit{Prob.amplitudes}) & \kappa, \iota, \ldots & \in & \mathbb{C} \\
(\mathit{Patterns}) & p, q & ::= & x \mid (x, y) \\
(\mathit{Terms}) & t, u, e & ::= & x \mid () \mid (t, u) \\
& & \mid & \mathbf{let}\ p = t\ \mathbf{in}\ u \\
& & \mid & \mathbf{if}^{\circ}\ t\ \mathbf{then}\ u\ \mathbf{else}\ u' \\
& & \mid & \mathit{false} \mid \mathit{true} \mid \vec{0} \mid \kappa * t \mid t + u
\end{array}
$$

The classical sub-language consists of variables, **let**-expressions, unit, pairs, booleans, and conditionals. Quantum data is modelled using the constructs $\kappa{*}t$, $\vec{0}$, and $t{+}u$. The term $\kappa{*}t$, where κ is a complex number, associates the *probability amplitude* κ with the term t. It is convenient to have a special constant $\vec{0}$ for terms with probability amplitude zero. The term $t + u$ is a quantum *superposition* of t and u. Quantum superpositions are first-class values: when used as the first subexpression of a conditional, they turn the conditional into a *quantum control* construct. For example, $\mathbf{if}^{\circ}\ (\mathit{true}{+}\mathit{false})\ \mathbf{then}\ t\ \mathbf{else}\ u$ evaluates both t and u and combines their results in a quantum superposition.

3.1 Examples

To provide further insight into the semantics of QML, we consider a few interesting examples. In these examples, we allow the definition and use of "global" function symbols. Adding such definitions to the formalism is possible but tedious, so we keep them at an informal meta-level.

The following three functions correspond to simple rotations on qubits:

$$\begin{aligned}
&\mathit{qnot}\ x &&= \mathbf{if}^{\circ}\ x\ \mathbf{then}\ \mathit{false}\ \mathbf{else}\ \mathit{true}\\
&\mathit{had}\ x &&= \mathbf{if}^{\circ}\ x\ \mathbf{then}\ ((-1) * \mathit{true} + \mathit{false})\ \mathbf{else}\ (\mathit{true} + \mathit{false})\\
&z\ x &&= \mathbf{if}^{\circ}\ x\ \mathbf{then}\ ((-1) * \mathit{true})\ \mathbf{else}\ \mathit{false}
\end{aligned}$$

The first is the quantum version of boolean negation; it behaves as usual when applied to classical values but it also applies to quantum data. Evaluating $\mathit{qnot}\ (\kappa * \mathit{false} + \iota * \mathit{true})$ swaps the probability amplitudes associated with *false* and *true*. The second function represents the fundamental *Hadamard* matrix, and the third represents the *Pauli-Z* operator.

The function:

$$\begin{aligned}
\mathit{cnot}\ c\ x = \mathbf{if}^{\circ}\ &c\\
&\mathbf{then}\ (\mathit{true},\ \mathit{qnot}\ x)\\
&\mathbf{else}\ \ (\mathit{false}, x)
\end{aligned}$$

is the conditional-not operation, which behaves as follows: if the control qubit c is *true* it negates the second qubit x; otherwise it leaves it unchanged. When the control qubit is in some superposition of *true* and *false*, the result is a superposition of the two pairs resulting from the evaluation of each branch of the conditional. For example, evaluating $\mathit{cnot}\ (\mathit{false} + \mathit{true})\ \mathit{false}$ produces the *entangled* pair $(\mathit{false}, \mathit{false}) + (\mathit{true}, \mathit{true})$.

3.2 Copying and Discarding Quantum Data

To motivate the main aspects of the type system in the next section, we examine in detail the issues related to copying and discarding quantum data.

A simple example where quantum data appears to be copied, in violation of the *no-cloning* theorem [NC00], is:

$$\begin{aligned}
&\mathbf{let}\ x = \mathit{false} + \mathit{true}\\
&\mathbf{in}\ (x, x)
\end{aligned}$$

As the formal semantics of QML clarifies, this expression does not actually clone quantum data; rather it *shares* one copy of the quantum data. In other words the expression does *not* evaluate to $(\mathit{false} + \mathit{true}, \mathit{false} + \mathit{true})$ which would make it impossible to realise. Rather the expression evaluates to $(\mathit{false}, \mathit{false}) + (\mathit{true}, \mathit{true})$ which is realisable (and easily so). With this interpretation, one can freely duplicate variables bound to quantum data. When translated to the type system, this means that the type system imposes no restrictions on the use of the structural rule of *contraction*.

Discarding variables bound to quantum data, however, is problematic. Consider the expression:

let $(x, y) = (\mathit{false}, \mathit{false}) + (\mathit{true}, \mathit{true})$
in x

where the quantum data bound to y is discarded. According to both the physical interpretations of quantum computation and the semantics of QML, this corresponds to a *measurement* of y. Since measurement is semantically quite complicated to deal with, we insist that it should be represented explicitly. The language we consider in this paper lacks the explicit constructs for measurement so we reject the expression above. This means that the structural rule of *weakening* is never allowed in situations where information may be lost. More precisely the only value to which weakening applies is the unit value () as it carries no information.

4 The Classical Sub-language

By the classical sub-language, we mean the subset of terms excluding quantum superpositions. This also precludes the use of quantum control.

4.1 Type System

The main rôle of the type system is to control the use of variables. The typing rules of QML are based on strict linear logic, where contractions are implicit and weakenings are not allowed when they correspond to information loss. As explained in the previous section, weakenings correspond to measurements, which are not supported in the subset of the language discussed in this paper.

We use σ, τ, ρ to vary over QML types which are given by the following grammar:

$\sigma = \mathcal{Q}_1 \mid \mathcal{Q}_2 \mid \sigma \otimes \tau$

where $\mathcal{Q}_1$ is the type of (), and $\mathcal{Q}_2$ is the type of qubits. As apparent from the grammar, QML types are first-order and finite; there are no higher-order types and no recursive types. The only types we can represent are the types of collections of qubits.

Typing contexts (Γ, Δ) are given by:

$\Gamma = \bullet \mid \Gamma, x : \sigma$

where $\bullet$ stands for the empty context, but is omitted if the context is non-empty. For simplicity we assume that every variable appears at most once. Contexts correspond to functions from a finite set of variables to types. We introduce the operator $\otimes$, which maps pairs of contexts to contexts:

$$\begin{aligned}
(\Gamma, x : \sigma) \otimes (\Delta, x : \sigma) &= (\Gamma \otimes \Delta), x : \sigma \\
(\Gamma, x : \sigma) \otimes \Delta &= (\Gamma \otimes \Delta), x : \sigma \text{ if } x \notin \mathrm{dom}\,(\Delta) \\
\bullet \otimes \Delta &= \Delta
\end{aligned}$$

As explained in the typing rules, the operator allows us to share variables appearing in a given context. This operation is partial: it is only well-defined if

$$\frac{}{x:\sigma \vdash x:\sigma}\text{var} \qquad \frac{\Gamma \vdash t:\sigma \qquad \Delta, x:\sigma \vdash u:\tau}{\Gamma \otimes \Delta \vdash \textbf{let } x = t \textbf{ in } u:\tau}\textbf{let}$$

$$\frac{}{\bullet \vdash ():\mathcal{Q}_1}\text{unit} \qquad \frac{\Gamma \vdash t:\sigma \quad \Delta \vdash u:\tau}{\Gamma \otimes \Delta \vdash (t,u):\sigma \otimes \tau}\otimes\text{-intro}$$

$$\frac{\Gamma \vdash t:\sigma \otimes \tau \qquad \Delta,\, x:\sigma, y:\tau \vdash u:\rho}{\Gamma \otimes \Delta \vdash \textbf{let } (x,y) = t \textbf{ in } u:\rho}\otimes\text{-elim}$$

$$\frac{}{\bullet \vdash \mathit{false}:\mathcal{Q}_2}\text{f-intro} \qquad \frac{}{\bullet \vdash \mathit{true}:\mathcal{Q}_2}\text{t-intro}$$

$$\frac{\Gamma \vdash c:\mathcal{Q}_2 \qquad \Delta \vdash t,u:\sigma}{\Gamma \otimes \Delta \vdash \ \textbf{if}^\circ\ \ c\ \ \textbf{then}\ \ t\ \ \textbf{else}\ \ u:\sigma}\textbf{if}^\circ \qquad \frac{\Gamma \vdash t:\sigma}{\Gamma, x:\mathcal{Q}_1 \vdash t:\sigma}\text{wk-unit}$$

Fig. 1. Typing classical terms

the two contexts do not assign different types to the same variable. Whenever we use this operator we implicitly assume that it is well-defined.

Figure 1 presents the rules for deriving valid typing judgements $\Gamma \vdash t : \sigma$. The only variables that may be dropped from the context are the ones of type $\mathcal{Q}_1$ which, by definition, carry no information. Otherwise the type system forces every variable in the context to be used (perhaps more than once if it is shared).

4.2 The Category of Typed Terms

The set of typed terms can be organised in an elegant categorical structure, which facilitates some of the proofs given later. The objects of the category are contexts; the homset between the objects Γ and Δ, denoted $\mathrm{Tm}\,\Gamma\,\Delta$, consists of all terms t such that $\Gamma \vdash t : |\Delta|$ where $|\Delta|$ views the context Δ as a type. This latter map is defined as follows:

$$|\bullet| = \mathcal{Q}_1$$

$$|\Gamma, x:\sigma| = |\Gamma| \otimes \sigma$$

For each context Γ, the identity $1_\Gamma \in \mathrm{Tm}\Gamma\Gamma$ is defined as follows:

$$1_\bullet = ()$$

$$1_{\Gamma,x:\sigma} = (1_\Gamma, x)$$

Given $d \in \mathrm{Tm}\,\Delta\,\Gamma$ and $e \in \mathrm{Tm}\,\Gamma\,\Theta$, the composition $e \circ d \in \mathrm{Tm}\,\Delta\,\Theta$ is given by the term $\textbf{let}^*\ \Gamma = d \textbf{ in } e$. The **let**$*$ construct is an abbreviation for iterated **let**-expressions which bind each of the variables in the intermediate context:

$$\begin{array}{lcl} \mathbf{let}^* \bullet = d \ \mathbf{in} \ e & \equiv & e \\ \mathbf{let}^* \ \Gamma, x:\sigma = d \ \mathbf{in} \ e & \equiv & \mathbf{let} \ (x_r, x) = d \ \mathbf{in} \ \mathbf{let}^* \ \Gamma = x_r \ \mathbf{in} \ e \end{array}$$

$$\frac{\Delta \vdash d : |\Gamma| \quad \Gamma \vdash e : |\Theta|}{\Delta \vdash \mathbf{let}^* \ \Gamma = d \ \mathbf{in} \ e : |\Theta|}$$

4.3 Semantics

The intention is to interpret every type σ and every context Γ as finite sets $[\![\sigma]\!]$ and $[\![\Gamma]\!]$, and then interpret a judgement $\Gamma \vdash t : \sigma$ as a function $[\![\Gamma \vdash t : \sigma]\!] \in [\![\Gamma]\!] \to [\![\sigma]\!]$. In the classical case, the type $\mathcal{Q}_2$ is simply the type of booleans and $\otimes$ is the standard product type:

$$\begin{aligned} [\![\mathcal{Q}_1]\!] &= \{0\} \\ [\![\mathcal{Q}_2]\!] &= \{0,1\} \\ [\![\sigma \otimes \tau]\!] &= [\![\sigma]\!] \times [\![\tau]\!] \end{aligned}$$

We use the abbreviation $[\![\Gamma]\!]$ for $[\![|\Gamma|]\!]$.

The meaning function is defined in Figure 2 by induction over the structure of type derivations. It uses the following auxiliary maps:

- $id : S \to S$ defined by $id(a) = a$
- $id^* : S \to [\![\mathcal{Q}_1]\!] \times S$ and its inverse id_* defined by $id^*(a) = (0, a)$ and $id_*(0, a) = a$
- For $a \in S$, the family of constant functions $const\ a : [\![\mathcal{Q}_1]\!] \to S$ defined by $(const\ a)(0) = a$.
- $\delta : S \to (S, S)$ defined by $\delta(a) = (a, a)$
- $swap : S \times T \to T \times S$ defined by $swap(a, b) = (b, a)$. We will usually implicitly use *swap* to avoid cluttering the figures with maps which just re-shuffle values.
- For any two functions $f \in S_1 \to T_1$ and $g \in S_2 \to T_2$, the function $(f \times g) : (S_1 \times S_2) \to (T_1 \times T_2)$ is defined as usual:

  $$(f \times g)(a, b) = (f\ a, g\ b)$$
- $\delta_{\Gamma,\Delta} : [\![\Gamma \otimes \Delta]\!] \to [\![\Gamma]\!] \times [\![\Delta]\!]$. This map is defined by induction on the definition of $\Gamma \otimes \Delta$ as follows:

  $$\delta_{\Gamma,\Delta} = \begin{cases} \delta_{\Gamma',\Delta'} \times \delta \text{ if } \Gamma = \Gamma', x:\sigma \text{ and } \Delta = \Delta', x:\sigma \\ \delta_{\Gamma',\Delta} \times id \text{ if } \Gamma = \Gamma', x:\sigma \text{ and } x \notin \mathrm{dom}\,(\Delta) \\ id^* \text{ if } \Gamma = \bullet \end{cases}$$

  Intuitively, the map $\delta_{\Gamma,\Delta}$ takes an incoming environment for an expression, creates shared copies of the appropriate values, and rearranges them (the shuffling is implicit and not shown in the above definition) into two environments that are then passed to the subexpressions.

$$
\begin{aligned}
[\![\bullet \vdash () : \mathcal{Q}_1]\!] &= \mathit{const}\ 0 \\
[\![\bullet \vdash \mathit{false} : \mathcal{Q}_2]\!] &= \mathit{const}\ 0 \\
[\![\bullet \vdash \mathit{true} : \mathcal{Q}_2]\!] &= \mathit{const}\ 1 \\
[\![x : \sigma \vdash x : \sigma]\!] &= \mathit{id}_* \\
[\![\Gamma \otimes \Delta \vdash \mathbf{let}\ x = t\ \ \mathbf{in}\ \ u : \tau]\!] &= g \circ (f \times \mathit{id}) \circ \delta_{\Gamma,\Delta} \\
&\quad \text{where } f = [\![\Gamma \vdash t : \sigma]\!] \\
&\quad \phantom{\text{where }} g = [\![\Delta, x : \sigma \vdash u : \tau]\!] \\
[\![\Gamma \otimes \Delta \vdash (t, u) : \sigma \otimes \tau]\!] &= (f \times g) \circ \delta_{\Gamma,\Delta} \\
&\quad \text{where } f = [\![\Gamma \vdash t : \sigma]\!] \\
&\quad \phantom{\text{where }} g = [\![\Delta \vdash u : \tau]\!] \\
[\![\Gamma \otimes \Delta \vdash \mathbf{let}\ (x, y) = t\ \mathbf{in}\ u : \rho]\!] &= g \circ (f \times \mathit{id}) \circ \delta_{\Gamma,\Delta} \\
&\quad \text{where } f = [\![\Gamma \vdash t : \sigma \otimes \tau]\!] \\
&\quad \phantom{\text{where }} g = [\![\Delta,\, x : \sigma, y : \tau \vdash u : \rho]\!] \\
[\![\Gamma \otimes \Delta \vdash\ \mathbf{if}^{\circ}\ \ c\ \ \mathbf{then}\ \ t\ \ \mathbf{else}\ \ u : \sigma]\!] &= (g|h) \circ (f \times \mathit{id}) \circ \delta_{\Gamma,\Delta} \\
&\quad \text{where } f = [\![\Gamma \vdash c : \mathcal{Q}_2]\!] \\
&\quad \phantom{\text{where }} g = [\![\Delta \vdash t : \sigma]\!] \\
&\quad \phantom{\text{where }} h = [\![\Delta \vdash u : \sigma]\!] \\
[\![\Gamma \vdash t : \sigma]\!] &= f \circ \mathit{id}^* \\
&\quad \text{where } f = [\![\Gamma, x : \mathcal{Q}_1 \vdash t : \sigma]\!]
\end{aligned}
$$

Fig. 2. Meaning of classical derivations

- For any two functions $f, g \in S \to T$, we define the conditional $f|g \in ([\![\mathcal{Q}_2]\!] \times S) \to T$ as follows:

$$
\begin{aligned}
(f|g)\ (1, a) &= f\ a \\
(f|g)\ (0, a) &= g\ a
\end{aligned}
$$

4.4 Equational Theory

We present the equational theory for the classical sub-language and then show its soundness and completeness. The equations refer to a set of syntactic values defined as follows:

$\mathit{val} \ \in\ \mathrm{Val}^{\mathrm{C}} ::= x \mid () \mid \mathit{false} \mid \mathit{true} \mid (\mathit{val}_1, \mathit{val}_2)$

Definition 4.1 The *classical equations* are grouped in four categories. The equations are implicitly typed and this entails conditions on the occurrence of variables, *e.g.*, the first commuting conversion can only be well-typed if the variables bound in p and q do not appear in t or u.

- **let**-equation

$$\textbf{let } p = val \textbf{ in } u \quad \equiv \quad u\,[val\,/\,p]$$

- β-equations

$$\begin{array}{lcl} \textbf{let } (x,y) = (t,u) \textbf{ in } e & \equiv & \textbf{let } x = t \textbf{ in let } y = u \textbf{ in } e \\ \textbf{if}^\circ\ \mathit{false} \textbf{ then } t \textbf{ else } u & \equiv & u \\ \textbf{if}^\circ\ \mathit{true} \textbf{ then } t \textbf{ else } u & \equiv & t \end{array}$$

- η-equations

$$\begin{array}{lcl} () & \equiv & t \quad \text{-- if t:}\mathcal{Q}_1 \\ \textbf{let } x = t \textbf{ in } x & \equiv & t \\ \textbf{let } (x,y) = t \textbf{ in } (x,y) & \equiv & t \\ \textbf{if}^\circ\ t \textbf{ then } \mathit{true} \textbf{ else } \mathit{false} & \equiv & t \end{array}$$

- Commuting conversions

$$\begin{array}{lcl} \textbf{let } p = t \textbf{ in let } q = u \textbf{ in } e & \equiv & \textbf{let } q = u \textbf{ in let } p = t \textbf{ in } e \\ \textbf{let } p = \textbf{if}^\circ\ t \textbf{ then } u_0 \textbf{ else } u_1 & \equiv & \textbf{if}^\circ\ t \\ \textbf{in } e & & \textbf{then let } p = u_0 \textbf{ in } e \\ & & \textbf{else} \ \ \textbf{let } p = u_1 \textbf{ in } e \end{array}$$

We write $\Gamma \vdash t \equiv u : \sigma$ if $\Gamma \vdash t, u : \sigma$ and the equation $t \equiv u$ is derivable at the type σ.

Lemma 4.2 (Soundness) *The equational theory is sound: if $\Gamma \vdash t \equiv u : \sigma$ then the functions $[\![\Gamma \vdash t : \sigma]\!]$ and $[\![\Gamma \vdash u : \sigma]\!]$ are extensionally equal.*

5 Completeness of the Classical Theory

The equational theory is *complete* in a strong technical sense: any equivalence implied by the semantics is derivable in the theory. The proof technique is based on recent work by Altenkirch with Uustalu [AU04]. The proof presented here extends and simplifies the method presented in that work.

5.1 Proof Technique

The ultimate goal is to prove the following statement.

Proposition 5.1 (Completeness) *If $[\![\Gamma \vdash t : \sigma]\!]$ and $[\![\Gamma \vdash u : \sigma]\!]$ are extensionally equal, then we can derive $\Gamma \vdash t \equiv u : \sigma$.*

In order to prove this statement, we define a function q_Γ^σ which inverts evaluation by producing a canonical syntactical representative. In fact, we define the function q_Γ^σ such that it maps a denotation $[\![\Gamma \vdash t : \sigma]\!]$ to the normal form of t.

Definition 5.2 If $\Gamma \vdash t : \sigma$, the *normal form* of t is given by $\mathrm{nf}_\Gamma^\sigma(t) = q_\Gamma^\sigma([\![\Gamma \vdash t : \sigma]\!])$.

The normal form is well-defined: given an equation $\Gamma \vdash t \equiv u : \sigma$, we know by soundness that $[\![\Gamma \vdash t : \sigma]\!]$ is extensionally equal $[\![\Gamma \vdash u : \sigma]\!]$ and hence we get that $\mathrm{nf}^{\sigma}_{\Gamma}(t) = \mathrm{nf}^{\sigma}_{\Gamma}(u)$. If we now show that the syntactic theory can prove that every term is equal to its normal form, then we can prove the main completeness result. Indeed given the following lemma, we can prove completeness.

Lemma 5.3 (Inversion) *If $\Gamma \vdash t : \sigma$, the equation $\Gamma \vdash nf^{\sigma}_{\Gamma}(t) \equiv t : \sigma$ is derivable.*

Proof of Proposition 5.1 (Completeness) We have:

$$\begin{aligned} \Gamma \vdash t \equiv q^{\sigma}_{\Gamma}[\![\Gamma \vdash t : \sigma]\!] : \sigma & \quad \text{by inversion} \\ \Gamma \vdash q^{\sigma}_{\Gamma}[\![\Gamma \vdash t : \sigma]\!] \equiv q^{\sigma}_{\Gamma}[\![\Gamma \vdash u : \sigma]\!] : \sigma & \quad \text{by assumption} \\ \Gamma \vdash q^{\sigma}_{\Gamma}[\![\Gamma \vdash u : \sigma]\!] \equiv u : \sigma & \quad \text{by inversion} \end{aligned}$$

□

In summary, we can establish completeness by defining a function q^{σ}_{Γ} that inverts evaluation and that satisfies Lemma 5.3.

5.2 Adequacy

We begin by defining a family of functions q^{σ} (*quote*) which invert the evaluation of *closed* terms and prove a special case of the inversion lemma for closed terms called *adequacy*. These functions and the adequacy result are then used in the next section to invert the evaluation of open terms and prove the general inversion lemma.

Definition 5.4 The *syntactic representation of denotations* is given by:

$$q^{\sigma} \in [\![\sigma]\!] \to \mathrm{Val}^{\mathrm{C}}\sigma$$

defined by induction over σ:

$$\begin{aligned} q^{\mathcal{Q}_1}\, 0 &= () \\ q^{\mathcal{Q}_2}\, 0 &= \mathit{false} \\ q^{\mathcal{Q}_2}\, 1 &= \mathit{true} \\ q^{\sigma \otimes \tau}\,(a, b) &= (q^{\sigma}\, a, q^{\tau}\, b) \end{aligned}$$

The instance of the inversion lemma for closed terms is called *adequacy*. It guarantees that the equational theory is rich enough to equate every closed term with its final observable value.

Lemma 5.5 (Adequacy) *The equation $\vdash q^{\sigma}([\![\ \vdash t : \sigma]\!]\ 0) \equiv t : \sigma$ is derivable.*

Proof sketch. During the proof of such a statement we encounter open terms that must be closed before they are "quoted." So in fact the statement to prove by induction over typing derivations is the following:

$$\text{If } g \in [\![\Gamma]\!] \text{ then } \vdash q^\sigma([\![\Gamma \vdash t : \sigma]\!]\ g) \equiv \mathbf{let}^*\ \Gamma = q^\Gamma\ (g)\ \mathbf{in}\ t : \sigma$$

□

5.3 *Inverting Evaluation*

As explained earlier, the main ingredient of the proof of completeness is the function q^σ_Γ which inverts evaluation. To understand the basic idea of how the inverse of evaluation is defined, consider the following example. Let Γ be the environment $x : (\mathcal{Q}_2 \otimes \mathcal{Q}_2), y : \mathcal{Q}_2$ and let $f \in [\![\Gamma]\!] \to [\![\mathcal{Q}_2]\!]$. To find a syntactic term corresponding to f, we proceed as follows:

- Flatten all the products by introducing intermediate names which produces an updated environment $\Gamma' = x_1 : \mathcal{Q}_2, x_2 : \mathcal{Q}_2, y : \mathcal{Q}_2$, and an updated semantic function f' such that:
$$f'\ ((((), x_1), x_2), y) = f\ (((), (x_1, x_2)), y)$$
- Enumerate all possible values for the variables, and apply f' to each enumeration to produce a result in the set $[\![\mathcal{Q}_2]\!]$. For example, it could be the case that $f\ (((), (1, 1)), 1) = 0$. The result of each enumeration can be inverted to a syntactic term using q^σ from Definition 5.4.
- Put things together using nested conditions representing all the possible values for the input variables. In the example we are considering, we get:

$$\begin{array}{l}
\mathbf{let}\ (x1, x2) = x \\
\mathbf{in\ if}^\circ\ x1 \\
\quad \mathbf{then\ if}^\circ\ x2 \\
\quad\quad \mathbf{then\ if}^\circ\ y\ \mathbf{then}\ \mathit{false} \\
\quad\quad\quad\quad\quad\quad \mathbf{else}\quad \ldots \\
\quad\quad \mathbf{else}\quad \ldots \\
\quad \mathbf{else}\quad \ldots
\end{array}$$

The idea is formalised in the following definition.

Definition 5.6 The function

$$q^\sigma_\Gamma \in ([\![\Gamma]\!] \to [\![\sigma]\!]) \to \mathrm{Tm}\,\Gamma\,\sigma$$

for *inverting evaluation* is defined by analysing the context:

$$\begin{aligned}
q^\sigma_\bullet(f) &= q^\sigma\ (f\ (0)) \\
q^\sigma_{\Gamma,x:\mathcal{Q}_1}(f) &= q^\sigma_\Gamma\ (h) \qquad \text{where } h(g) = f(g, 0) \\
q^\sigma_{\Gamma,x:\mathcal{Q}_2}(f) &= (\ \mathbf{if}^\circ\ \ x\ \ \mathbf{then}\ \ q^\sigma_\Gamma\ (h_1)\ \ \mathbf{else}\ \ q^\sigma_\Gamma\ (h_0)) \\
&\qquad \text{where } h_i(g) = f(g, i) \text{ for } i \in \{0, 1\} \\
q^\sigma_{\Gamma,x:(\tau_1 \otimes \tau_2)}(f) &= (\mathbf{let}\ (x_1, x_2) = x\ \ \mathbf{in}\ \ q^\sigma_{\Gamma,x_1:\tau_1,x_2:\tau_2}(h) \\
&\qquad \text{where } h(g, x_1, x_2) = f(g, (x_1, x_2))
\end{aligned}$$

$$\frac{}{\bullet \vdash \overrightarrow{0} : \sigma}\ \text{z-intro} \qquad \frac{\Gamma \vdash t : \sigma}{\Gamma \vdash \kappa * t : \sigma}\ \text{prob} \qquad \frac{\Gamma \vdash t, u : \sigma}{\Gamma \vdash t + u : \sigma}\ \text{sup}$$

Fig. 3. Typing quantum data (I)

The base case is straightforward: the evaluation produces a closed value which can be inverted using the *quote* function of Definition 5.4. If the context includes a variable x of type $\mathcal{Q}_1$, then we supply the only possible value for that variable (0), and inductively construct the term with the variable x bound to (). The result is of the correct type because we can add or drop bindings of variables of type $\mathcal{Q}_1$ to the environment. If the context includes a variable x of type $\mathcal{Q}_2$, then we supply the two possible values for that variable 0 and 1. A conditional is then used to select the correct branch depending on the actual value of x. Finally, if the context includes a variable of type $\tau_1 \otimes \tau_2$ then we simply flatten the product and proceed inductively. The function q_Γ^σ does indeed satisfy the inversion lemma 5.3.

6 Quantum Data and Control

We develop the typing rules and semantics of the quantum fragment of QML in two stages. First we extend the judgements $\Gamma \vdash t : \sigma$ and the semantics of Section 4 to handle quantum data in a straightforward manner. However, this simple treatment is only an intermediate step in the development, as it admits quantum programs that are not realisable in a pure quantum system without measurement. We then refine both the type system and the semantics to identify exactly the realisable quantum programs.

6.1 *The Category* **Vec**

As a first approximation to a type system for QML programs, we consider the type system of Figure 1 extended with the rules in Figure 3.

Unlike the classical case, a judgement $\Gamma \vdash t : \sigma$ is *not* interpreted as a function in $[\![\Gamma]\!] \to [\![\sigma]\!]$. Rather, because we now have superpositions of terms with complex probability amplitudes, we interpret such judgements as functions in $[\![\Gamma]\!] \to [\![\sigma]\!]^{\mathrm{Q}}$ where $[\![\sigma]\!]^{\mathrm{Q}}$ represents the complex vectors over the base set $[\![\sigma]\!]$. In other words, $[\![\sigma]\!]^{\mathrm{Q}}$ is defined to be $[\![\sigma]\!] \to \mathbb{C}$ which is sometimes denoted $\mathbf{V}\ [\![\sigma]\!]$. All functions in the space must be linear which means that if $f \in \mathbf{V}\ A \to \mathbf{V}\ B$, $\alpha \in \mathbb{C}$, and $v, v_1, v_2 \in \mathbf{V}\ A$, then $f(v_1 + v_2) = f(v_1) + f(v_2)$ and $f(\alpha v) = \alpha(f\ v)$. We call the structure described above the category **Vec**.

This change requires that we revisit the semantics of the classical terms given in Figure 2 so that each denotation returns a complex vector. For example, we now have:

$$[\![\bullet \vdash \mathit{false} : \mathcal{Q}_2]\!]^{\mathrm{Q}} = \mathit{const}\ v \quad \text{where } v\ 0 = 1.0 \text{ and } v\ 1 = 0.0$$

Instead of mapping the value representing the empty context to the denotation

$$
\begin{array}{rll}
[\![\bullet \vdash \overrightarrow{0} : \sigma]\!]^{\mathrm{Q}} = & \mathit{const}\ v & \text{where } \forall a \in [\![\sigma]\!].\ v\ a = 0.0 \\
[\![\Gamma \vdash \kappa * t : \sigma]\!]^{\mathrm{Q}} = & g & \text{where } \forall r \in [\![\Gamma]\!], a \in [\![\sigma]\!].\ g\ r\ a = \kappa * (f\ r\ a) \\
& & \qquad f = [\![\Gamma \vdash t : \sigma]\!]^{\mathrm{Q}} \\
[\![\Gamma \vdash t + u : \sigma]\!]^{\mathrm{Q}} = & h & \text{where } \forall r \in [\![\Gamma]\!], a \in [\![\sigma]\!].\ h\ r\ a = \frac{1}{\sqrt{2}}(f\ r\ a + g\ r\ a) \\
& & \qquad f = [\![\Gamma \vdash t : \sigma]\!]^{\mathrm{Q}} \\
& & \qquad g = [\![\Gamma \vdash u : \sigma]\!]^{\mathrm{Q}}
\end{array}
$$

Fig. 4. Meaning function for quantum data

of *false*, we now return a vector v which associates the denotation of *false* with probability amplitude 1.0 and the denotation of *true* with probability amplitude 0.0.

This change can be done fairly systematically by using a monad whose *unit* and *lift* operations are defined below:

$$\mathit{return} \ :: \ S \to \mathbf{V}\ S$$

$$\mathit{return}\ a\ (b) = 1.0 \text{ if } a = b \text{ and } 0.0 \text{ otherwise}$$

$$(.^*) \ :: \ (S \to \mathbf{V}\ T) \to (\mathbf{V}\ S \to \mathbf{V}\ T)$$

$$f^*(v)(b) = \Sigma a.(v\ a) * (f\ a\ b)$$

More precisely, the changes to Figure 2 consist of:

- Every result is explicitly tagged with the monadic *return* . This turns every function of type $S \to T$ to a function of type $S \to \mathbf{V}\ T$ and hence all the composition operators on functions must be lifted to account for the fact that the result type is monadic.
- The composition $g \circ f$ becomes $g^* \circ f$
- The composition $f \times g$ uses the tensor product on the vectors instead of the classical product, *i.e.*: $(f \times g)(a, b)(x, y) = f\ a\ x * g\ b\ y$
- The composition $f|g$ requires non-trivial changes as explained in Section 6.3

The updated meaning function of Figure 2 together with the new cases in Figure 4 give the complete definition of the meaning function for QML.

6.2 Orthogonality

The type system presented so far correctly tracks the uses of variables and prevents variables from being weakened, yet the situation is more subtle. It turns out that the type system accepts terms which implicitly perform measurements and as a consequence accepts programs which are not realisable as pure quantum computations.

$$\frac{\Gamma \vdash^{\circ} c : \mathcal{Q}_2 \qquad \Delta \vdash^{\circ} t, u : \sigma \qquad t \perp u}{\Gamma \otimes \Delta \vdash^{\circ} \mathbf{if}^{\circ} \ c \ \mathbf{then} \ t \ \mathbf{else} \ u : \sigma} \, \mathbf{if}^{\circ}$$

$$\frac{\Gamma \vdash^{\circ} t, u : \sigma \qquad t \perp u \qquad |\lambda|^2 + |\kappa|^2 = 1}{\Gamma \vdash^{\circ} \lambda * t + \kappa * u : \sigma} \, \mathrm{sup}^{\circ}$$

$$\frac{\Gamma \vdash^{\circ} t : \sigma \qquad \Gamma \vdash t \equiv u : \sigma}{\Gamma \vdash^{\circ} u : \sigma} \, \mathrm{subst}$$

Fig. 5. Typing quantum data (II)

Consider the expression $\mathbf{if}^{\circ}\ x$ **then** *true* **else** *true*. This expression appears to use x, syntactically at least. However given the semantics of $\mathbf{if}^{\circ}$, which returns a superposition of the branches, the expression happens to return *true* without *really* using any information about x, *i.e.*, it effectively forgets or measures x. In order to maintain the invariant that all measurements are explicit, the type system should therefore reject such an expression.

More precisely, the expression $\mathbf{if}^{\circ}\ x$ **then** t **else** u should only be accepted if t and u are *orthogonal* quantum values ($t \perp u$). This notion intuitively ensures that the conditional operator does not implicitly discard any information about x during the evaluation. Indeed the incoming value of x is a superposition of the two orthogonal basis vectors representing *false* and *true*. If we require that the result of the $\mathbf{if}^{\circ}$ expression is another superposition of orthogonal values, then it essentially becomes a rotation. Because of a similar concern, the two branches of a superposition should also be orthogonal.

We therefore introduce a new typing judgement $\Gamma \vdash^{\circ} t : \sigma$ which is similar to the judgement $\Gamma \vdash t : \sigma$ except that it uses modified rules for conditionals and superpositions which require that the relevant subexpressions are orthogonal. The modified rules are given in Figure 5. The modification also achieves that programs are normalised, *i.e.*, the sum of the probabilities of a superposition add up to 1. The judgement $\vdash^{\circ}$ is not automatically closed under the equality judgement, hence we add the rule (subst). Our philosophy is that we allow equivalent representations of QML programs which do not satisfy the orthogonality criteria locally, as long as the program as a whole is equivalent to one which does satisfy the criteria.

It remains to define the syntactic orthogonality judgement $t \perp u$. Semantically the vectors v_t and v_u representing the values of the terms t and u are orthogonal if their inner product $\langle v_t | v_u \rangle$ is 0. We provide a syntactic approximation of the inner product in Figure 6: the syntactic version assigns to any pair of terms $\Gamma \vdash t, u : \sigma$ a value $\langle t | u \rangle \in \mathbb{C} \cup \{?\}$ where ? is a "don't know" answer. In the figure $\overline{\lambda}$ is the conjugate of the complex number λ. We extend multiplication to this domain by $0 * x = x * 0$ for any $x \in \mathbb{C} \cup \{?\}$ and $x * ? = ? * x = ?$ for $x \neq 0$. Addition is also extended by $x + ? = ? + x = ?$. The approximation is sound but clearly incomplete.

$$
\begin{array}{rlrl}
\langle \mathrm{t}|\mathrm{t}\rangle &= 1 \text{ if } \mathrm{t} \neq \overrightarrow{0} & \langle \lambda * t + \lambda' * t' \mid u\rangle &= \overline{\lambda} * \langle t|u\rangle + \overline{\lambda'} * \langle t'|u\rangle \\
\langle \mathit{false}|\mathit{true}\rangle &= 0 & \langle t \mid \kappa * u + \kappa' * u'\rangle &= \kappa * \langle t|u\rangle + \kappa' * \langle t|u'\rangle \\
\langle \mathit{true}|\mathit{false}\rangle &= 0 & & \\
 & & \langle \lambda * t|u\rangle &= \overline{\lambda}\langle t|u\rangle \\
\langle \overrightarrow{0}|\mathit{true}\rangle &= 0 = \langle \mathit{true}|\overrightarrow{0}\rangle & \langle t|\lambda * u\rangle &= \lambda * \langle t|u\rangle \\
\langle \overrightarrow{0}|\mathit{false}\rangle &= 0 = \langle \mathit{false}|\overrightarrow{0}\rangle & \langle t + t'|u\rangle &= \langle t|u\rangle + \langle t'|u\rangle \\
\langle \overrightarrow{0}|x\rangle &= 0 = \langle x|\overrightarrow{0}\rangle & \langle t|u + u'\rangle &= \langle t|u\rangle + \langle t|u'\rangle \\
 & & & \\
\langle (t,t') \mid (u,u')\rangle &= \langle t|u\rangle * \langle t'|u'\rangle & \langle t|u\rangle &= \text{? otherwise}
\end{array}
$$

Fig. 6. Inner products and orthogonality

6.3 The Category $\mathbf{Q}^\circ$

The restriction of the set of typable terms requires a similar semantic restriction. We require that all the functions preserve the inner products of vectors, *i.e.*, that they are *isometries*. A function $f \in \mathbf{V}\ A \to \mathbf{V}\ B$ is an isometry if for all $v_1, v_2 \in \mathbf{V}\ A$, we have that $\langle v_1|v_2\rangle = \langle f\ v_1|f\ v_2\rangle$. Intuitively such functions can never "forget" any information which is consistent with our desire to prevent situations which would accidently measure a value.

It is fairly straightforward to show that all the functions introduced by the denotational semantics are isometries except of course for the cases of superpositions and $\mathbf{if}^\circ$ which require side-conditions on their inputs as motivated in the previous section. Semantically the syntactic condition of orthogonality translates to the following requirement on the corresponding meaning functions. These functions must map arbitrary input environments to orthogonal vectors. We say that two *morphisms* f, g in $\mathbf{V}\ A \to \mathbf{V}\ B$ are *orthogonal* is for any two vectors $v_1, v_2 \in \mathbf{V}\ A$, we have that $\langle f\ v_1|g\ v_2\rangle = 0$. To explain this semantic restriction, we consider the case of superpositions in detail; the case for $\mathbf{if}^\circ$ is similar. Looking at the semantic definition in Figure 4, we would like to guarantee that h^* is an isometry given that both f^* and g^* are both isometries. We calculate as follows. Let r_1 and r_2 be vectors in $\mathbf{V}\ \Gamma$, then using some fairly simple but tedious calculations we get:

$$
\begin{aligned}
\langle h^* r_1|h^* r_2\rangle &= \tfrac{1}{2}(\langle f^* r_1|f^* r_2\rangle + \langle g^* r_1|g^* r_2\rangle + \langle f^* r_1|g^* r_2\rangle + \langle g^* r_1|f^* r_2\rangle) \\
&= \langle r_1|r_2\rangle + \tfrac{1}{2}(\langle f^* r_1|g^* r_2\rangle + \langle g^* r_1|f^* r_2\rangle) \\
&= \langle r_1|r_2\rangle
\end{aligned}
$$

The first step is because f^* and g^* are both isometries that preserve the inner product. The second step is because of the additional requirement that f^* is orthogonal g^*.

We call the resulting category of vectors and isometries the category of strict quantum computations, $\mathbf{Q}^\circ$. The homset of morphisms in $[\![\Gamma]\!] \to [\![\sigma]\!]^{\mathrm{Q}}$ is called $\mathbf{Q}^\circ\ [\![\Gamma]\!]\ [\![\sigma]\!]^{\mathrm{Q}}$. The meaning function is given as before but with the maps interpreted in the category $\mathbf{Q}^\circ$, *i.e.*, the meaning of a derivation $\Gamma \vdash t : \sigma$ is a morphism $[\![\Gamma \vdash t : \sigma]\!]^{\mathrm{Q}} \in \mathbf{Q}^\circ\ [\![\Gamma]\!]\ [\![\sigma]\!]^{\mathrm{Q}}$. As explained above, the requirement for orthogonality in the type system is reflected semantically: the superposition is an isometry if the two components are orthogonal; similarly, the conditional $f|g$ is an isometry if f and g are orthogonal.

6.4 Quantum Equational Theory

The equational theory for the quantum language inherits all the equations for the classical case. This can be informally verified by noting that the meaning function in the case of the quantum language is essentially identical to the classical case. Formally, the proof technique explained in Section 4 applies equally well to the quantum case and yields the same equations for the classical core plus additional equations to deal with quantum data.

Definition 6.1 The *quantum equations* are:

(if$^\circ$)

$$\begin{array}{ll} & \mathbf{if}^\circ\ (\lambda * t_0 + \kappa * t_1)\ \mathbf{then}\ u_0\ \mathbf{else}\ u_1 \\ \equiv & \lambda * (\mathbf{if}^\circ\ t_0\ \mathbf{then}\ u_0\ \mathbf{else}\ u_1) + \kappa * (\mathbf{if}^\circ\ t_1\ \mathbf{then}\ u_0\ \mathbf{else}\ u_1) \end{array}$$

(superpositions)

$$\begin{array}{lcl} t + u & \equiv & u + t \\ t + \vec{0} & \equiv & t \\ t + (u + v) & \equiv & (t + u) + v \\ \lambda * (t + u) & \equiv & \lambda * t + \lambda * u \\ \lambda * t + \kappa * t & \equiv & (\lambda + \kappa) * t \\ 0 * t & \equiv & \vec{0} \end{array}$$

Lemma 6.2 (Soundness) *The equational theory is sound. Given $\Gamma \vdash t \equiv u : \sigma$ then the isometries $[\![\Gamma \vdash t : \sigma]\!]^Q$ and $[\![\Gamma \vdash u : \sigma]\!]^Q$ are extensionally equal.*

The additional equations are used to prove equality between different quantum values. Semantically, two quantum values are the same if they denote the same vector, which is the case if the sum of the paths to each classical value is the same. For example, to find a simplified quantum value equivalent to:

$(\mathit{false} + \mathit{true}) + (\mathit{false} + (-1) * \mathit{true})$

we first normalise to:

$(1\ /\ \sqrt{2}) * ((1\ /\ \sqrt{2}) * \mathit{false} + (1\ /\ \sqrt{2}) * \mathit{true}) +$
$(1\ /\ \sqrt{2}) * ((1\ /\ \sqrt{2}) * \mathit{false} + (-1\ /\ \sqrt{2}) * \mathit{true})$

This term has two paths to *false*; along each of them the product of the amplitudes is $(1\ /\ \sqrt{2}) * (1\ /\ \sqrt{2})$ which is $1\ /\ 2$. The sum of all the paths to *false* is 1, and the sum of all the paths to *true* is 0. In other words, the entire term is equivalent to simply *false*. The above calculation proves that the Hadamard operation is self-inverse, as discussed in the introduction.

6.5 *Quoting quantum values*

We will now adapt the techniques developed in section 4 to the quantum case. A classical value $v \in \mathrm{Val}^{\mathrm{C}}\sigma$ is simply a term representing an element in $[\![\sigma]\!]$. A quantum value represents a vector in $\mathbf{V}\ [\![\sigma]\!]^{\mathrm{Q}}$, hence we have to close values under superpositions. We define $\mathrm{Val}^{\mathrm{Q}}\,\sigma \subseteq \mathrm{Tm}\ \sigma$ inductively as a subset of closed terms of type σ:

$$\frac{v \in \mathrm{Val}^{\mathrm{C}}\,\sigma}{\mathrm{val}\,v \in \mathrm{Val}^{\mathrm{Q}}\,\sigma} \qquad 0 \in \mathrm{Val}^{\mathrm{Q}}\,\sigma$$

$$\frac{v, w \in \mathrm{Val}^{\mathrm{Q}}\,\sigma}{v + w \in \mathrm{Val}^{\mathrm{Q}}\,\sigma} \qquad \frac{v \in \mathrm{Val}^{\mathrm{Q}}\,\sigma}{\kappa * v \in \mathrm{Val}^{\mathrm{Q}}\,\sigma}$$

We write $\mathrm{Val}^{\mathrm{Q}}_{\circ}\sigma$ for isometric quantum values which satisfy the restrictions introduced in Figure 5.

We have already seen that there is a monadic structure on $\mathbf{V}\ A = A \to \mathbb{C}$. Correspondingly, we have a Kleisli structure on $\mathrm{Val}^{\mathrm{Q}}$. The return is $\mathrm{val} \in \mathrm{Val}^{\mathrm{C}}\sigma \to \mathrm{Val}^{\mathrm{Q}}\sigma$, and bind is defined as: given $v \in \mathrm{Val}^{\mathrm{Q}}\sigma$ and $f \in \mathrm{Val}^{\mathrm{C}}\sigma \to \mathrm{Val}^{\mathrm{Q}}\tau$, we define $v \gg\!= f \in \mathrm{Val}^{\mathrm{Q}}\,\tau$ by induction over v:

$$\begin{array}{lll}
(val\ x) & \gg\!= f & = f\ x \\
0 & \gg\!= f & = 0 \\
v + w & \gg\!= f & = (v \gg\!= f) + (w \gg\!= f) \\
\kappa * v & \gg\!= f & = \kappa * (v \gg\!= f)
\end{array}$$

Lemma 6.3 $(\mathrm{Val}^{\mathrm{C}}, \mathrm{Val}^{\mathrm{Q}}, val, (\gg\!=))$ *is a Kleisli structure, i.e. it satisfies the following equations:*

(i) $val\ x \gg\!= f \equiv f\ x$

(ii) $v \gg\!= \lambda x.val\ x \equiv v$

(iii) $v \gg\!= \lambda x.(f\ x) \gg\!= g \equiv (v \gg\!= f) \gg\!= g$

Proof. Case (i) follows from the definition. Cases (ii) and (iii) can be shown by induction over the structure of v. □

While the classical definition of q^{σ} (def. 5.4) was straightforward, its quantum counterpart is a bit more subtle, in particular in the case of tensor products. As a special case consider $q^{\mathcal{Q}_2 \otimes \mathcal{Q}_2}$. Given an element

$$\overrightarrow{v} \in [\![\mathcal{Q}_2 \otimes \mathcal{Q}_2]\!]^{\mathrm{Q}} = [\![\mathcal{Q}_2]\!] \times [\![\mathcal{Q}_2]\!] \to \mathbb{C}$$

we have to construct a value $q^{\mathcal{Q}_2 \otimes \mathcal{Q}_2}\,\overrightarrow{v} \in \mathrm{Val}^{\mathrm{Q}}\ \mathcal{Q}_2 \otimes \mathcal{Q}_2$. This can be done by calculating the probabilities that the first qubit is i, $\mathrm{fst}\,\overrightarrow{v}\,i \in \mathbb{R}^+$, given by

$$\mathrm{fst}\,\overrightarrow{v}\,i = \sqrt{|\overrightarrow{v}(i,0)|^2 + |\overrightarrow{v}(i,1)|^2}$$

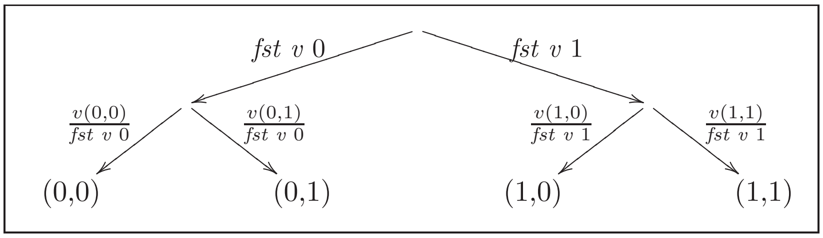

Fig. 7. Value tree for $\mathcal{Q}_2 \otimes \mathcal{Q}_2$

creating the first level of the value as a tree, and then for the second level normalising the amplitudes with respect to the probabilities of the previous level (see figure 7 for the corresponding tree.) We write $[\![\sigma]\!]^{\mathrm{P}} = [\![\sigma]\!] \to \mathbb{R}^{+}$ for the set of probability distributions; obviously we have $[\![\sigma]\!]^{\mathrm{P}} \subseteq [\![\sigma]\!]^{\mathrm{Q}}$. We observe that $\mathrm{fst}\ \overrightarrow{v} \in [\![\sigma]\!]^{\mathrm{P}}$. Generalising the idea given above we arrive at the following definition of the "quote" function.

Definition 6.4 The *syntactic representations of denotations* is given by:

$$q^{\sigma} \in [\![\sigma]\!]^{\mathrm{Q}} \to \mathrm{Val}^{\mathrm{Q}}\, \sigma$$

which is defined by induction over σ:

$$\begin{aligned}
q^{\mathcal{Q}_1}\, \overrightarrow{v} &= (\overrightarrow{v}\, 0) * val\ () \\
q^{\mathcal{Q}_2}\, \overrightarrow{v} &= (\overrightarrow{v}\, 1) * val\ \mathrm{true} + (\overrightarrow{v}\, 0) * val\ \mathrm{false} \\
q^{\sigma \otimes \tau}\, \overrightarrow{v} &= q^{\sigma} (\mathrm{fst}\ \overrightarrow{v}) \\
&\qquad\qquad \gg\!= \lambda x \in \mathrm{Val}^{\mathrm{C}} \sigma. (1/(\mathrm{fst}\ \overrightarrow{v})\, x) * q^{\tau} (\lambda y \in \mathrm{Val}^{\mathrm{C}} \sigma. \overrightarrow{v}\,(x, y)) \\
&\qquad\qquad \gg\!= \lambda y \in \mathrm{Val}^{\mathrm{C}} \sigma. val\ (x, y)
\end{aligned}$$

where:

$$\begin{aligned}
\mathrm{fst} &\in [\![\sigma \otimes \tau]\!]^{\mathrm{Q}} \to [\![\sigma]\!]^{\mathrm{P}} \\
\mathrm{fst}\ \overrightarrow{v}\, x &= \sqrt{\Sigma y. |\overrightarrow{v}\,(x, y)|^2}
\end{aligned}$$

$$\begin{aligned}
1/- &\in [\![\sigma]\!]^{\mathrm{P}} \to [\![\sigma]\!]^{\mathrm{P}} \\
1/\overrightarrow{v}\, x &= \lambda x. \mathbf{if}\ \ \overrightarrow{v}\ \ x \equiv 0\ \mathbf{then}\ 0\ \mathbf{else}\ 1/(\overrightarrow{v} x)
\end{aligned}$$

To show adequacy we have to establish a number of properties of q^{σ}. We have to show that it is linear and isometric, and that it preserves tensor products. This is summarised in the following proposition:

Proposition 6.5

(i) $q^{\sigma}\, (\kappa * \overrightarrow{v}) \equiv \kappa * (q^{\sigma}\, \overrightarrow{v})$

(ii) $q^{\sigma}\, (\overrightarrow{v} + \overrightarrow{w}) \equiv (q^{\sigma}\, \overrightarrow{v}) + (q^{\sigma}\, \overrightarrow{w})$

(iii) $\langle \overrightarrow{v} | \overrightarrow{w} \rangle = \langle q^{\sigma}\, \overrightarrow{v} | q^{\sigma}\, \overrightarrow{w} \rangle$

(iv) $q^{\sigma \otimes \tau}\, (\overrightarrow{v} \otimes \overrightarrow{w}) \equiv (q^{\sigma}\, \overrightarrow{v}, q^{\tau}\, \overrightarrow{w})$

The proof of the above proposition again isn't completely straightforward, as linearity cannot just be proven by induction over σ. It is essential that we first establish some properties of re-normalising a vector with respect to a probability distribution. We define the product of a probability distribution $p \in [\![\sigma]\!]^{\mathrm{P}}$ and a vector $\overrightarrow{v} \in [\![\sigma]\!]^{\mathrm{Q}}$ as:

$$\begin{aligned} p * \overrightarrow{v} &\in [\![\sigma]\!]^{\mathrm{Q}} \\ p * \overrightarrow{v} &= \lambda x \in [\![\sigma]\!].(px) * (\overrightarrow{v}\, x) \end{aligned}$$

Now we see that an analogous operation can be defined on values, given $v \in \mathrm{Val}^{\mathrm{Q}}\,\sigma$ and $p \in [\![\sigma]\!]^{\mathrm{P}}$ as above, we define:

$$\begin{aligned} p * v &\in \mathrm{Val}^{\mathrm{Q}}\,\sigma \\ p * v &= v \ggg \lambda x \in [\![\sigma]\!].(px) * (\mathrm{val}\, x) \end{aligned}$$

The key property we establish is the following.

Lemma 6.6 *Given $p \in [\![\sigma]\!]^{P}$ and $\overrightarrow{v} \in [\![\sigma]\!]^{Q}$*

$$p * (q^{\sigma}\, \overrightarrow{v}) \equiv q^{\sigma}\, (p * \overrightarrow{v})$$

The lemma can be verified by induction over σ, and observing that while $1/-$ isn't a proper inverse, it nevertheless satisfies the following property:

$$1/(p+q) * (p+q) = (1/p) * p$$

Using the fact that q^{σ} is isometric we can show that it produces values satisfying the orthogonality constraints.

Proposition 6.7 *Given $v \in [\![\sigma]\!]^{Q}$*

$$\vdash^{\circ} q^{\sigma}\, v : \sigma$$

6.6 Adequacy

We define a syntactic counterpart to $\delta_{\Gamma,\Delta} \in \mathbf{Q}^{\circ}\, [\![\Gamma \otimes \Delta]\!]\, ([\![\Gamma]\!]^{\mathrm{Q}} \otimes [\![\Delta]\!]^{\mathrm{Q}})$. which denoted by:

$$\hat{\delta}_{\Gamma,\Delta} \in \mathrm{Tm}\,(\Gamma \otimes \Delta)\,(|\Gamma| \otimes |\Delta|)$$

This term is defined as follows:

$$\hat{\delta}_{\Gamma,\Delta} = \begin{cases} \textbf{let } (g,d) = \delta_{\Gamma',\Delta'} \textbf{in } ((g,x),(d,x)) & \text{if } \Gamma = \Gamma', x : \sigma \\ & \quad \text{and } \Delta = \Delta', x : \sigma \\ \textbf{let } (g,d) = \delta_{\Gamma',\Delta} \textbf{in } ((g,x),d) & \text{if } \Gamma = \Gamma', x : \sigma \\ & \quad \text{and } x \notin \mathrm{dom}\,\Delta \\ 1_{\Delta} & \text{if } \Gamma = \bullet \end{cases}$$

To establish that q^{σ} commutes with the context operations we have to show by induction on σ that contraction corresponds to $\delta \in \mathbf{Q}^{\circ}\, [\![\sigma]\!]\, ([\![\sigma]\!]^{\mathrm{Q}} \otimes [\![\sigma]\!]^{\mathrm{Q}})$.

Lemma 6.8 *Given $v \in [\![\sigma]\!]^Q$ we have* $\textbf{let}\ x = q^\sigma\ v\ \textbf{in}\ (x, x) \equiv q^{\sigma\otimes\sigma}\ v$.

Exploiting this property we can show that the context operations commute with "quote".

Lemma 6.9 *Given $\overrightarrow{v} \in [\![\Gamma \otimes \Delta]\!]^Q$, we have:*

$$q^{|\Gamma|\otimes|\Delta|}\,(\delta_{\Gamma,\Delta}\ \overrightarrow{v}) \equiv \hat{\delta}_{\Gamma,\Delta}\, q^{|\Gamma\otimes\Delta|}\ \overrightarrow{v}$$

Theorem 6.10 *If $\Gamma \vdash t : \sigma$ and $g \in [\![\Gamma]\!]^Q$ then:*

$$\vdash q^\sigma([\![\Gamma \vdash t : \sigma]\!]^Q g) \equiv \textbf{\textit{let}}^*\ \Gamma = q^\Gamma\ g\ \textbf{in}\ t : \sigma.$$

Proof. By induction over the derivation of $\Gamma \vdash t : \sigma$, as an example consider the case for **let**:

$$
\begin{aligned}
& q^\rho\,([\![\Gamma \otimes \Delta \vdash \textbf{let}\ x = t\ \textbf{in}\ u : \rho]\!]^{\mathrm{Q}}) \\
& \equiv \{\text{definition of } [\![\ldots]\!]^{\mathrm{Q}}\} \\
& q^\rho\,([\![u]\!]^{\mathrm{Q}} \circ ([\![t]\!]^{\mathrm{Q}} \otimes id) \circ \delta_{\Gamma,\Delta}) \\
& \equiv \{\text{induction hypothesis for } u \text{ and } t\} \\
& u \circ (t \circ q^\Gamma \otimes q^\Delta) \circ \delta_{\Gamma,\Delta}) \\
& \equiv \{\ \text{Lemma } 6.9\ \} \\
& u \circ (t \otimes id) \circ \hat{\delta}_{\Gamma,\Delta} \circ q^{|\Gamma\otimes\Delta|} \\
& \equiv (\textbf{let}\ x = t\ \textbf{in}\ u) \circ q^{|\Gamma\otimes\Delta|}
\end{aligned}
$$

The other cases use the same style of reasoning to deal with the structural properties and exploit proposition 6.5. Note that the case for $\textbf{if}^\circ$ can be reduced to linearity. □

Corollary 6.11 (Adequacy) *If $\vdash t : \sigma$ then $\vdash q^\sigma([\![\ \vdash t : \sigma]\!]^Q) \equiv t : \sigma$*

6.7 Completeness and Normalisation

The development here follows closely the one in the classical case as presented in Section 5.3.

Definition 6.12 The function $q_\Gamma^\sigma \in \mathbf{Q}^\circ\ [\![\Gamma]\!]\ [\![\sigma]\!]^{\mathrm{Q}} \to \mathrm{Tm}\,\Gamma\,\sigma$ for *inverting evaluation* is defined by analysing the context:

$$
\begin{aligned}
q_\bullet^\sigma(f) &= q^\sigma\ (f\ (\mathit{return}\ 0)) \\
q_{\Gamma,x:\mathcal{Q}_1}^\sigma(f) &= \phi_{\Gamma,x:\mathcal{Q}_1}^{-1} \circ (q_\Gamma^\rho) \circ \Phi_{\Gamma,x:\mathcal{Q}_1} \\
q_{\Gamma,x:\mathcal{Q}_2}^\sigma(f) &= \phi_{\Gamma,x:\mathcal{Q}_2}^{-1} \circ (q_\Gamma^\sigma \times q_\Gamma^\sigma) \circ \Phi_{\Gamma,x:\mathcal{Q}_2} \\
q_{\Gamma,x:(\tau_1\otimes\tau_2)}^\sigma(f) &= \phi_{\Gamma,x:\tau_1\otimes\tau_2}^{-1} \circ q_{\Gamma,x_1:\tau_1,x_2:\tau_2}^\sigma \circ \Phi_{\Gamma,x:\tau_1\otimes\tau_2}
\end{aligned}
$$

The auxiliary isomorphisms are defined as follows:

$$\begin{aligned}
\phi_{\Gamma,x:\mathcal{Q}_1} &\in \mathrm{Tm}\,(\Gamma, x : \mathcal{Q}_1)\,\sigma \to \mathrm{Tm}\,\Gamma\,\sigma \\
\phi_{\Gamma,x:\mathcal{Q}_1} t &= \textbf{let } x = () \textbf{ in } t \\
\phi_{\Gamma} t &= t
\end{aligned}$$

$$\begin{aligned}
\phi_{\Gamma,x:\mathcal{Q}_2} &\in \mathrm{Tm}\,(\Gamma, x : \mathcal{Q}_2\,\sigma) \to \{(t_0, t_1) \in (\mathrm{Tm}\,\Gamma\,\sigma)^2 \mid t_0 \perp t_1\} \\
\phi_{x:\mathcal{Q}_2}\, t &= (\textbf{let } x = \mathit{false} \textbf{ in } t, \textbf{let } x = \mathit{true} \textbf{ in } t) \\
\phi^{-1}_{\Gamma,x:\mathcal{Q}_2}(t, u) &= \textbf{ if}^{\circ}\ x \textbf{ then } t \textbf{ else } u
\end{aligned}$$

$$\begin{aligned}
\phi_{\Gamma,x:\tau_1\otimes\tau_2} &\in \mathrm{Tm}\,(\Gamma, x : \tau_1 \otimes \tau_2)\,\rho \to \mathrm{Tm}\,(\Gamma, x_1 : \tau_1, x_2 : \tau_2) \\
\phi_{\Gamma,x:\tau_1\otimes\tau_2}\, t &= \textbf{let } x = (x_1, x_2) \textbf{ in } t \\
\phi^{-1}_{\Gamma,x:\tau_1\otimes\tau_2}(t) &= \textbf{let } (x_1, x_2) = x \textbf{ in } t
\end{aligned}$$

Since the isomorphisms ϕ are defined as an operation on terms, we have corresponding isomorphisms in the semantic category ($\mathbf{Q}^{\circ}$) which we denote by Φ.

For the inversion proof we only need the provability of one side of the isomorphisms which follows from the η-equalities.

Lemma 6.13 *The family of equalities $\phi_{\Gamma}^{-1}(\phi_{\Gamma} t) \equiv t$ is derivable.*

Definition 6.14 The *normal form* of t is given by $\mathrm{nf}^{\sigma}_{\Gamma}(t) = q^{\sigma}_{\Gamma}([\![\Gamma \vdash t : \sigma]\!]^{\mathrm{Q}})$.

Lemma 6.15 (Inversion) *The equation $\Gamma \vdash nf^{\sigma}_{\Gamma}(t) \equiv t$ is derivable.*

Proof. By induction over the definition of q^{σ}_{Γ}. In the case of $\Gamma = \bullet$ the result follows from adequacy, Corollary 6.11. In all the other cases we exploit Lemma 6.13. □

Since all our definitions are effective *nf* indeed gives rise to a normalisation algorithm. As a consequence, our equational theory is decidable, modulo deciding equalities of the complex number terms which occur in our programs. We also note that as in the classical case, our theory is complete.

Proposition 6.16 (Completeness) *If $[\![\Gamma \vdash t : \sigma]\!]^{Q}$ and $[\![\Gamma \vdash u : \sigma]\!]^{Q}$ are extensionally equal, then we can derive $\Gamma \vdash t \equiv u : \sigma$.*

7 Conclusions and Further Work

We have developed a sound and complete equational theory for a functional quantum programming language, while at the same time providing a normalisation algorithm. The construction is a modular extension of a classical theory; indeed the quantum theory inherits not just all the equations and term formers, it is also possible to generalise our proof technique to the quantum case. The quantum theory introduces additional constructs corresponding to superpositions and equations relating them.

The obvious next step is to generalise this approach to the full language QML including measurements. The equational theory is already a challenge, since a measurement can have non-local effects on shared data. Semantically, we will use superoperators to model programs with measurements. Clearly, we have to extend our quote operator to work on density matrices.

Another interesting direction would be to consider higher-order quantum programs and develop a complete equational theory and normalisation algorithm for this calculus. A likely semantic domain is given by presheaves, here the tensor product can be modelled using Day's construction, which is automatically closed, *i.e.*, provides an interpretation for higher types.

References

[AD04] P. Arrighi and G. Dowek. Operational semantics for a formal tensorial calculus. In *Proceedings of the* 2^{nd} *International Workshop on Quantum Programming Languages*, 2004.

[AG05] Thorsten Altenkirch and Jonathan Grattage. A functional quantum programming language. In Prakash Panangaden, editor, *Proceedings of the Twentieth Annual IEEE Symp. on Logic in Computer Science, LICS 2005*, pages 249–258. IEEE Computer Society Press, June 2005.

[AU04] T. Altenkirch and T. Uustalu. Normalization by evaluation for $\lambda^{\rightarrow 2}$. In *Functional and Logic Programming*, number 2998 in LNCS, pages 260 – 275. Springer-Verlag, 2004.

[NC00] M. Nielsen and I. Chuang. *Quantum Computation and Quantum Information.* Cambridge University Press, Cambridge, 2000.

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

[SV05] P. Selinger and B. Valiron. A lambda calculus for quantum computation with classical control. In *Proceedings of the Seventh International Conference on Typed Lambda Calculi and Applications (TLCA), Springer-Verlag, LNCS 3461*, pages 354–368, 2005.

[vT03] A. van Tonder. Quantum computation, categorical semantics and linear logic. Available as quant-ph/0312174, 2003.

[vT04] A. van Tonder. A lambda calculus for quantum computation. *SIAM Journal on Computing*, 33(5):1109–1135, 2004.