Typing Quantum Superpositions and Measurement

Alejandro Díaz-Caro, Gilles Dowek

Typing Quantum Superpositions and Measurement*

Alejandro Díaz-Caro[1][0000−0002−5175−6882] and Gilles Dowek[2]

[1] Universidad Nacional de Quilmes & CONICET
Roque Sáenz Peña 352, B1876BXD Bernal, Buenos Aires, Argentina
alejandro.diaz-caro@unq.edu.ar
[2] Inria, LSV, ENS Paris-Saclay
61, avenue du Président Wilson, 94235 Cachan Cedex, France
gilles.dowek@ens-paris-saclay.fr

Abstract. We propose a way to unify two approaches of non-cloning in quantum lambda-calculi. The first approach is to forbid duplicating variables, while the second is to consider all lambda-terms as algebraic-linear functions. We illustrate this idea by defining a quantum extension of first-order simply-typed lambda-calculus, where the type is linear on superposition, while allows cloning base vectors. In addition, we provide an interpretation of the calculus where superposed types are interpreted as vector spaces and non-superposed types as their basis.

Keywords: quantum computing, lambda-calculus, algebraic linearity, linear logic, measurement

1 Introduction

In λ-calculus, applying the term $\lambda x\ (x \otimes x)$, that expresses a non-linear function for some convenient definition of $\otimes$, to a term u yields the term $(\lambda x\ (x \otimes x))u$, that reduces to $u \otimes u$. But "cloning" this vector u is forbidden in quantum computing. Various quantum λ-calculi address this problem in different ways.

One way is to forbid the construction of the term $\lambda x\ (x \otimes x)$ using a typing system inspired from linear logic [1,9], leading to logic-linear calculi [2,10,11,13,14]. Another is to consider all λ-terms expressing linear functions. The term $\lambda x\ (x \otimes x)$, for instance, expresses the linear function that maps $|0\rangle$ to $|0\rangle \otimes |0\rangle$ and $|1\rangle$ to $|1\rangle \otimes |1\rangle$[1]. This leads to restrict beta-reduction to the case where u is a base vector (in the computational basis) and to add the linearity rule $f(u+v) \longrightarrow (fu+fv)$, leading to algebraic-linear calculi [3–6, 8].

Each solution has its advantages and drawbacks. For example, let $t?u{\cdot}v$ be the conditional statement on $|0\rangle$ and $|1\rangle$. Interpreting λ-terms as algebraic-linear functions permits to reduce the term $(\lambda x\ x?|0\rangle{\cdot}|1\rangle)(\alpha.\,|0\rangle + \beta.\,|1\rangle)$ to

* Partially funded by the STIC-AmSud Project FoQCoSS and PICT-PRH 2015-1208.

[1] Where $|x\rangle$ is the Dirac notation for vectors, with $|0\rangle = \left(\begin{smallmatrix}1\\0\end{smallmatrix}\right) \in \mathbb{C}^2$ and $|1\rangle = \left(\begin{smallmatrix}0\\1\end{smallmatrix}\right) \in \mathbb{C}^2$, so $\{|0\rangle, |1\rangle\}$ is an orthonormal basis of $\mathbb{C}^2$, called here the "computational basis".

$(\alpha.(\lambda x\ x?|0\rangle\cdot|1\rangle)|0\rangle + \beta.(\lambda x\ x?|0\rangle\cdot|1\rangle)|1\rangle)$ then to $(\alpha.|1\rangle + \beta.|0\rangle)$, instead of reducing it to the term $(\alpha.|0\rangle + \beta.|1\rangle)?|0\rangle\cdot|1\rangle$ that would be blocked. This explains that this linearity rule, that is systematic in the algebraic-linear languages cited above, is also present for the condition in [2] (the so-called $\mathbf{if}^\circ$ operator).

However, interpreting all λ-terms as linear functions forbids to extend the calculus with non-linear operators, such as measurement. For instance, the term $(\lambda x\ \pi x)(|0\rangle + |1\rangle)$, where π represents a measurement in the computational basis, would reduce to $((\lambda x\ \pi x)\,|0\rangle + (\lambda x\ \pi x)\,|1\rangle)$, while it should reduce to $|0\rangle$ with probability $\frac{1}{2}$ and to $|1\rangle$ with probability $\frac{1}{2}$.

In this paper, we propose a way to unify the two approaches, distinguishing duplicable and non-duplicable data by their type, like in the logic-linear calculi; and interpreting λ-terms as linear functions, like in the algebraic-linear calculi, when they expect duplicable data. We illustrate this idea with an example of such a calculus.

In this calculus, a qubit has type $\mathbb{B}$ when it is in the computational basis, hence duplicable (a non-linear term in the sense of linear logic), and $S(\mathbb{B})$ when it is a superposition, hence non-duplicable (a linear term in the sense of linear logic). Hence, the term $|0\rangle \otimes (|0\rangle + |1\rangle)$ has type $\mathbb{B} \otimes S(\mathbb{B})$. Giving this type to this term and the type $S(\mathbb{B} \otimes \mathbb{B})$ to the term $(|0\rangle \otimes |0\rangle + |0\rangle \otimes |1\rangle)$ however jeopardizes the subject reduction property as, using the bilinearity of the product, the former should develop to the latter. This dilemma is not specific to quantum computing as computing is often a non-reversible process where some information is lost. For instance, if we express, in its type, that the term $(X-1)(X-2)$ is a product of two polynomials, developing it to $X^2 - 3X + 2$ does not preserve this type. A solution is to introduce, in the language, an explicit cast. For example, from the type of tensor products to the type of arbitrary vectors. The term $|0\rangle \otimes (|0\rangle + |1\rangle)$ then has type $\mathbb{B} \otimes S(\mathbb{B})$ and it cannot be reduced. But the term $\Uparrow (|0\rangle \otimes (|0\rangle + |1\rangle))$ has type $S(\mathbb{B} \otimes \mathbb{B})$ and can be developed to $(|0\rangle \otimes |0\rangle + |0\rangle \otimes |1\rangle)$.

This language permits expressing quantum algorithms with a very precise information about the nature of the data processed by these algorithms.

Outline of the Paper In Section 2 we introduce the calculus, without tensor. In Section 3 we extend the language with a tensor operator for multiple-qubits systems, and state the Subject Reduction property of the resulting system. In Section 4 we provide a straightforward interpretation of the calculus considering base types as sets of vectors, and types $S(\cdot)$ as vector spaces. Finally, in Section 5 we express a non-trivial example in our calculus: the Teleportation algorithm, demonstrating the expressivity of the proposed language. A long version of this paper (51 pages) with all the detailed proofs is available at `arXiv:1601.04294`.

2 No-cloning, Superpositions and Measurement

The grammar of types and terms is defined as follows, with $\alpha \in \mathbb{C}$.

$$\begin{array}{rll} \Psi := \mathbb{B} \mid S(\Psi) & & \text{Qubit types } (\mathcal{Q}) \\ A := \Psi \mid \Psi \Rightarrow A \mid S(A) & & \text{Types } (\mathcal{T}) \\ b := x \mid \lambda x : \Psi\ t \mid |0\rangle \mid |1\rangle & & \text{Base terms } (\mathcal{B}) \\ v := b \mid (v + v) \mid \vec{0}_{S(A)} \mid \alpha.v & & \text{Values } (\mathcal{V}) \end{array}$$

$t := v \mid tt \mid (t+t) \mid \pi t \mid ?\cdot \mid \alpha.t$ Terms (Λ)

Terms are variables, abstractions, applications, two constants for base qubits ($|0\rangle$ and $|1\rangle$), linear combinations of terms (built with addition and product by a scalar, addition being commutative and associative), a family of constants for the null vectors, one for each type of the form $S(A)$, ($\vec{0}_{S(A)}$), and an if-then-else construction ($?\cdot$) deciding on base vectors. We also include a symbol π for measurement in the computational basis.

The set of free variables of a term t is defined as usual in λ-calculus and denoted by $FV(t)$. We use $[\alpha.]t$ as a notation to refer indistinctly to $\alpha.t$ and to t. We use $-t$ as a shorthand notation for $-1.t$, and $(t-r)$ as a shorthand notation for $(t+(-r))$. The term $(t-t)$ has type $S(A)$, and reduces to $\vec{0}_{S(A)}$, which is not a base term.

An important property of this calculus is that types $S(\cdot)$ are linear types. Indeed, those correspond to superpositions, and so no duplication is allowed on them. Instead, at this tensor-free stage, a type without an $S(\cdot)$ on head position is a non-linear type, such as $\mathbb{B}$, which correspond to base terms, i.e. terms that can be cloned. A non-linear function is allowed to be applied to a linear argument, for example, $\lambda x : \mathbb{B}\ (fxx)$ can be applied to $(\frac{1}{\sqrt{2}}.|0\rangle + \frac{1}{\sqrt{2}}.|1\rangle)$, however, it distributes in the following way: $(\lambda x : \mathbb{B}\ (fxx))\ (\frac{1}{\sqrt{2}}.|0\rangle + \frac{1}{\sqrt{2}}.|1\rangle) \longrightarrow (\frac{1}{\sqrt{2}}.(\lambda x : \mathbb{B}\ (fxx))\,|0\rangle + \frac{1}{\sqrt{2}}.(\lambda x : \mathbb{B}\ (fxx))\,|1\rangle) \longrightarrow (\frac{1}{\sqrt{2}}.(f|0\rangle\,|0\rangle) + \frac{1}{\sqrt{2}}.(f\,|1\rangle\,|1\rangle))$.

Hence, the beta reduction occurs only when the type of the argument is the same as the type expected by the abstraction. Thus, the rewrite system depends on types. For this reason, we describe first the type system, and only then the rewrite system.

A type A will be interpreted as a set of vectors and $S(A)$ as the vector space generated by the span of such a set (cf. Section 4). Hence, we naturally have $A \subseteq S(A)$ and $S(S(A)) = S(A)$. Therefore, we also define a subtyping relation on types. The type system and the subtyping relation are given below, where contexts Γ and Δ have a disjoint support.

$$\frac{\overline{A \preceq S(A)} \quad \overline{S(S(A)) \preceq S(A)} \quad \frac{A \preceq B}{\Psi \Rightarrow A \preceq \Psi \Rightarrow B} \quad \frac{A \preceq B}{S(A) \preceq S(B)}}{\overline{x : \Psi \vdash x : \Psi}\ Ax \quad \overline{\vdash \vec{0}_{S(A)} : S(A)}\ Ax_{\vec{0}} \quad \overline{\vdash |0\rangle : \mathbb{B}}\ Ax_{|0\rangle} \quad \overline{\vdash |1\rangle : \mathbb{B}}\ Ax_{|1\rangle}}$$

$$\frac{\Gamma \vdash t : A}{\Gamma \vdash \alpha.t : S(A)}\ S_I^{\alpha} \qquad \frac{\Gamma \vdash t : A \quad \Delta \vdash u : A}{\Gamma, \Delta \vdash (t+u) : S(A)}\ S_I^{+} \qquad \frac{\Gamma \vdash t : S(\mathbb{B})}{\Gamma \vdash \pi t : \mathbb{B}}\ S_E$$

$$\frac{\Gamma \vdash t : A \ (A \preceq B)}{\Gamma \vdash t : B}\ \preceq \qquad \overline{\vdash ?\cdot : \mathbb{B} \Rightarrow \mathbb{B} \Rightarrow \mathbb{B} \Rightarrow \mathbb{B}}\ If \qquad \frac{\Gamma, x : \Psi \vdash t : A}{\Gamma \vdash \lambda x : \Psi\ t : \Psi \Rightarrow A}\ \Rightarrow_I$$

$$\frac{\Gamma \vdash t : \Psi \Rightarrow A \quad \Delta \vdash u : \Psi}{\Gamma, \Delta \vdash tu : A}\ \Rightarrow_E \qquad \frac{\Gamma \vdash t : S(\Psi \Rightarrow A) \quad \Delta \vdash u : S(\Psi)}{\Gamma, \Delta \vdash tu : S(A)}\ \Rightarrow_{ES}$$

$$\frac{\Gamma \vdash t : A}{\Gamma, x : \mathbb{B} \vdash t : A}\ W \qquad \frac{\Gamma, x : \mathbb{B}, y : \mathbb{B} \vdash t : A}{\Gamma, x : \mathbb{B} \vdash (x/y)t : A}\ C$$

Remarks: Rule Ax allows typing variables only with qubit types. Hence, the system is first-order and only qubits can be passed as arguments (more when the rewrite system is presented). Rule $Ax_{\vec{0}}$ types the null vector as a non-base term, because the null vector cannot belong to the base of any vector space.

Thanks to rule $\preceq$ the term $|0\rangle$ has type $\mathbb{B}$ and also the more general type $S(\mathbb{B})$. Note that $((|0\rangle + |0\rangle) - |0\rangle)$ has type $S(\mathbb{B})$ and reduces to $|0\rangle$ that has the same type $S(\mathbb{B})$. Reducing this term to $|0\rangle$ of type $\mathbb{B}$ would not preserve its type. Moreover, this type would contain information impossible to compute, because the value $|0\rangle$ is not the result of a measurement, but of an interference.

Rule S_I^α states that a term multiplied by a scalar is not a base term. Even if the scalar is just a phase, we must type the term with an $S(\cdot)$ type, because our measurement operator removes the scalars, so having the scalar means that it has not been measured yet. Rule S_I^+ is the analog for sums to the previous rule. Rule S_E is the elimination of the superposition, which is achieved by measuring (using the π operator).

We use $r?s{\cdot}t$ as a notation for $(?\cdot)rst$. Notice that it is typed as a non-linear function by rule *If*, and so, the if-then-else linearly distributes over superpositions.

Rule $\Rightarrow_{ES}$ is the elimination for superpositions, corresponding to the linear distribution. Notice that the type of the argument is a superposition of the argument expected by the abstraction ($S(\Psi)$ vs. Ψ). Also, the abstraction is allowed to be a superposition. If, for example, we want to apply the sum of functions $(f + g)$ to the base argument $|0\rangle$, we would obtain the superposition $(f\,|0\rangle + g\,|0\rangle)$. The typing is as follows:

$$\dfrac{\dfrac{\vdash f : \mathbb{B} \Rightarrow A \quad \vdash g : \mathbb{B} \Rightarrow A}{\vdash (f+g) : S(\mathbb{B} \Rightarrow A)}\, S_I^+ \quad \dfrac{\dfrac{}{\vdash |0\rangle : \mathbb{B}}\, Ax_{|0\rangle}}{\vdash |0\rangle : S(\mathbb{B})}\, \preceq}{\vdash (f+g)\,|0\rangle : S(A)}\, \Rightarrow_{ES} \quad \longrightarrow \quad \dfrac{\dfrac{\vdash f : \mathbb{B} \Rightarrow A \quad \dfrac{}{\vdash |0\rangle : \mathbb{B}}\, Ax_{|0\rangle}}{\vdash f\,|0\rangle : A}\, \Rightarrow_E \quad \dfrac{\vdash g : \mathbb{B} \Rightarrow A \quad \dfrac{}{\vdash |0\rangle : \mathbb{B}}\, Ax_{|0\rangle}}{\vdash g\,|0\rangle : A}\, \Rightarrow_E}{\vdash (f\,|0\rangle + g\,|0\rangle) : S(A)}\, S_I^+$$

Similarly, a linear function ($\vdash f : \mathbb{B} \Rightarrow A$) applied to a superposition ($|0\rangle + |1\rangle$) reduces to a superposition ($f\,|0\rangle + f\,|1\rangle$).

Finally, Rules W and C correspond to weakening and contraction on variables with base types. The rationale is that base terms can be cloned.

The rewrite system is given bellow, where, in rule (proj), $\forall i$, $b_i = |0\rangle$ or $b_i = |1\rangle$, $\sum_{i=1}^n \alpha_i.b_i$ is normal (so $1 \leq n \leq 2$), if an α_k is absent, $|\alpha_k|^2 = 1$, and $1 \leq k \leq n$.

Beta	If b has type $\mathbb{B}$ and $b \in \mathcal{B}$, then $(\lambda x : \mathbb{B}\ t)b \longrightarrow_{(1)} (b/x)t$	(β_b)	$(\vec{0}_{S(A)} + t) \longrightarrow_{(1)} t$	(neutral)
			$1.t \longrightarrow_{(1)} t$	(unit)
	If u has type $S(\Psi)$, then $(\lambda x : S(\Psi)\ t)u \longrightarrow_{(1)} (u/x)t$	(β_n)	If t has type A, then $0.t \longrightarrow_{(1)} \vec{0}_{S(A)}$	(zero_α)
If	$\|1\rangle ?u{\cdot}v \longrightarrow_{(1)} u$	(if_1)	$\alpha.\vec{0}_{S(A)} \longrightarrow_{(1)} \vec{0}_{S(A)}$	(zero)
	$\|0\rangle ?u{\cdot}v \longrightarrow_{(1)} v$	(if_0)	$\alpha.(\beta.t) \longrightarrow_{(1)} (\alpha \times \beta).t$	(prod)
Linear distribution	If t has type $\mathbb{B} \Rightarrow A$, then $t(u + v) \longrightarrow_{(1)} (tu + tv)$	$(\mathsf{lin}_\mathsf{r}^+)$	$\alpha.(t + u) \longrightarrow_{(1)} (\alpha.t + \alpha.u)$	$(\alpha\mathsf{dist})$
			$(\alpha.t + \beta.t) \longrightarrow_{(1)} (\alpha + \beta).t$	(fact)
	If t has type $\mathbb{B} \Rightarrow A$ then $t(\alpha.u) \longrightarrow_{(1)} \alpha.tu$	$(\mathsf{lin}_\mathsf{r}^\alpha)$	$(\alpha.t + t) \longrightarrow_{(1)} (\alpha + 1).t$	(fact^1)
			$(t + t) \longrightarrow_{(1)} 2.t$	(fact^2)
	If t has type $\mathbb{B} \Rightarrow A$, then $t\vec{0}_{S(\mathbb{B})} \longrightarrow_{(1)} \vec{0}_{S(A)}$	$(\mathsf{lin}_\mathsf{r}^0)$	$(u + v) =_{AC} (v + u)$	(comm)
			$((u + v) + w) =_{AC} (u + (v + w))$	(assoc)
	$(t + u)v \longrightarrow_{(1)} (tv + uv)$	$(\mathsf{lin}_\mathsf{l}^+)$	$\pi(\sum_{i=1}^n [\alpha_i.]b_i) \longrightarrow_{(p)} b_k$ with $p = \frac{\lvert\alpha_k\rvert^2}{\sum_{i=1}^n \lvert\alpha_i\rvert^2}$	(proj)
	$(\alpha.t)u \longrightarrow_{(1)} \alpha.tu$	$(\mathsf{lin}_\mathsf{l}^\alpha)$		
	$\vec{0}_{S(\mathbb{B}\Rightarrow A)}t \longrightarrow_{(1)} \vec{0}_{S(A)}$	$(\mathsf{lin}_\mathsf{l}^0)$		
	$\dfrac{t \longrightarrow_{(p)} u}{tv \longrightarrow_{(p)} uv}$		$\dfrac{t \longrightarrow_{(p)} u}{(\lambda x : \mathbb{B}\ v)t \longrightarrow_{(p)} (\lambda x : \mathbb{B}\ v)u}$	

Vector space axioms: (neutral), (unit), (zero$_\alpha$), (zero), (prod), (αdist), (fact), (fact^1), (fact^2), (comm), (assoc), (proj)

$=$: (comm), (assoc)

Project.: (proj)

$$\frac{t \longrightarrow_{(p)} u}{t + v \longrightarrow_{(p)} u + v} \qquad \frac{t \longrightarrow_{(p)} u}{\alpha.t \longrightarrow_{(p)} \alpha u} \qquad \frac{t \longrightarrow_{(p)} u}{\pi t \longrightarrow_{(p)} \pi u}$$

The relation $\longrightarrow_{(p)}$ is a probabilistic relation where p is the probability of occurrence. Every rewrite rule has a probability 1 of occurrence, except for the projection ((proj) rule).

There are two beta rules. Rule (β_{b}) acts only when the argument is a base term, and the type expected by the abstraction is a base type. Hence, rule (β_{b}) is "call-by-base" (base terms coincides with values of λ-calculus, while values on this calculus also includes superpositions of base terms and the null vector). Instead, (β_{n}) is the usual call-by-name beta rule. They are distinguished by the type of the argument. Rule (β_{b}) acts on non-linear functions while (β_{n}) is for linear functions. The test on the type of the argument is due to the type system that allows an argument with a type not matching with the type expected by the abstraction (in such a case, one of the linear distribution rules applies).

The group If-then-else contains the tests over the base qubits $|0\rangle$ and $|1\rangle$.

The first three of the linear distribution rules (those marked with subindex r), are the rules that are used when a non-linear abstraction is applied to a linear argument (that is, when an abstraction expecting a base term is given a superposition). In these cases the beta reductions cannot be used since the side conditions on types are not met. Hence, these distributivity rules apply instead. The remaining rules in this group deal with a superposition of functions. For example, rule ($\mathsf{lin}_{\mathsf{l}}^{+}$) is the sum of functions: A superposition is a sum, therefore, if an argument is given to a sum of functions, it needs to be given to each function in the sum. We use a weak reduction strategy (i.e. reduction occurs only on closed terms), hence the argument v on this rule is closed, otherwise, it could not be typed. For example $x : S(\mathbb{B}), t : \mathbb{B} \Rightarrow \mathbb{B}, u : \mathbb{B} \Rightarrow \mathbb{B} \vdash (t + u)x : S(\mathbb{B})$ is derivable, but $x : S(\mathbb{B}), t : \mathbb{B} \Rightarrow \mathbb{B}, u : \mathbb{B} \Rightarrow \mathbb{B} \vdash (tx + ux) : S(\mathbb{B})$ is not.

The vector space axioms rules are the directed axioms of vector spaces [5, 6]. The Modulo AC rules are not proper rewrite rules, but express that we consider the symbol + to be associative and commutative, and hence our rewrite system is *rewrite modulo AC* [12].

Finally, rule (proj) is the projection over weighted associative pairs, that is, the projection over a generalization of multisets where the multiplicities are given by complex numbers. This reduction rule is the only one with a probability different from 1, and it is given by the square of the modulus of the weights[2], implementing this way the quantum measurement over the computational basis.

3 Multi-qubit Systems: Tensor Products

A multi-qubit system is represented with the tensor product between single-qubit Hilbert spaces. The tensor product of base terms can be seen as an ordered list. Hence, we represent the tensor product as a conjunction-like operator. The distributivity of linear combinations over tensor products is not trivially tracked in the type system, and so an explicit cast between types is also added.

[2] We speak about weights and not amplitudes, since the vector may not have norm 1. The projection rule normalizes the vector while reducing.

Each level in the term grammar (base terms, values and general terms) is extended with the tensor of the terms in such a level. The primitives *head* and *tail* are added to the general terms. The projector π is generalized to π_j, where the subindex j stands for the number of qubits to be measured, which are those in the first j positions. Notice that it is always possible to do a swap between qubits and so place the qubits to be measured at the beginning. For instance, $\lambda x : \mathbb{B} \otimes \mathbb{B}\ (\mathit{tail}\ x \otimes \mathit{head}\ x)$.

An explicit type cast of a term t ($\Uparrow_{S(A)}^{S(B\otimes C)} t$) is included in the general terms. It is only allowed to cast a superposed type into a superposed tensor product. We also add the tensor between types, and, as a consequence, a new level.

$$
\begin{array}{rr}
B := \mathbb{B} \mid B \otimes B & \text{Base qubit types } (\mathcal{B}) \\
\Psi := B \mid S(\Psi) \mid \Psi \otimes \Psi & \text{Qubit types } (\mathcal{Q}) \\
A := \Psi \mid \Psi \Rightarrow A \mid S(A) \mid A \otimes A & \text{Types } (\mathcal{T}) \\
b := x \mid \lambda x : \Psi\ t \mid |0\rangle \mid |1\rangle \mid b \otimes b & \text{Base terms } (\mathcal{B}) \\
v := b \mid (v + v) \mid \vec{0}_{S(A)} \mid \alpha.v \mid v \otimes v & \text{Values } (\mathcal{V}) \\
t := v \mid tt \mid (t + t) \mid \pi_j t \mid ?\cdot \mid \alpha.t \mid t \otimes t \mid \mathit{head}\ t \mid \mathit{tail}\ t \mid \Uparrow_{S(A)}^{S(B\otimes C)} t & \text{Terms } (\Lambda)
\end{array}
$$

The type system includes all the typing rules given in the previous section, plus the rules for tensor, for cast, and an updated rule S_E, for which we introduce the following notation:

Let $S \subseteq \{1, \cdots, n\}$. We define Q_n^S inductively by:

$$
\begin{aligned}
Q_n^S &= \begin{cases} A_{n-1}^S(\mathbb{B}) & \text{if } n \notin S \\ A_{n-1}^{S\setminus\{n\}}(S(\mathbb{B})) & \text{if } n \in S \end{cases} \\
A_0^{\emptyset}(B) &= B \\
A_{k+1}^S(\mathbb{B}) &= \begin{cases} A_k^S(\mathbb{B}) \otimes \mathbb{B} & \text{if } k+1 \notin S \\ A_k^{S\setminus\{k+1\}}(S(\mathbb{B})) \otimes \mathbb{B} & \text{if } k+1 \in S \end{cases} \\
A_{k+1}^S(S(B)) &= \begin{cases} A_k^S(\mathbb{B}) \otimes S(B) & \text{if } k+1 \notin S \\ A_k^{S\setminus\{k+1\}}(S(\mathbb{B} \otimes B)) & \text{if } k+1 \in S \end{cases}
\end{aligned}
$$

where B is any type.

In simple words, notation Q_n^S stands for a tensor of n qubits, where those indexed by the set S are superposed and typed with the most general type, for example $Q_3^{\{1,2\}}$ stands for $S(\mathbb{B} \otimes \mathbb{B}) \otimes \mathbb{B}$ and not for $S(\mathbb{B}) \otimes S(\mathbb{B}) \otimes \mathbb{B}$. The following example may be clarifying. $Q_5^{\{1,2,4\}} = A_4^{\{1,2,4\}}(\mathbb{B}) = A_3^{\{1,2\}}(S(\mathbb{B})) \otimes \mathbb{B} = A_2^{\{1,2\}}(\mathbb{B}) \otimes S(\mathbb{B}) \otimes \mathbb{B} = A_1^{\{1\}}(S(\mathbb{B})) \otimes \mathbb{B} \otimes S(\mathbb{B}) \otimes \mathbb{B} = A_0^{\emptyset}(S(\mathbb{B} \otimes \mathbb{B})) \otimes \mathbb{B} \otimes S(\mathbb{B}) \otimes \mathbb{B} = S(\mathbb{B} \otimes \mathbb{B}) \otimes \mathbb{B} \otimes S(\mathbb{B}) \otimes \mathbb{B}$.

In addition, we update the subtyping relation adding the following two rules.

$$\frac{A \preceq B}{A \otimes C \preceq B \otimes C} \quad \text{and} \quad \frac{A \preceq B}{C \otimes A \preceq C \otimes B}.$$

The updated type system is given below.

$$
\frac{}{x : \Psi \vdash x : \Psi}\ Ax \qquad \frac{}{\vdash \vec{0}_{S(A)} : S(A)}\ Ax_{\vec{0}} \qquad \frac{}{\vdash |0\rangle : \mathbb{B}}\ Ax_{|0\rangle} \qquad \frac{}{\vdash |1\rangle : \mathbb{B}}\ Ax_{|1\rangle}
$$

$$
\frac{\Gamma \vdash t : A}{\Gamma \vdash \alpha.t : S(A)}\ S_I^{\alpha} \qquad \frac{\Gamma \vdash t : A \quad \Delta \vdash u : A}{\Gamma, \Delta \vdash (t + u) : S(A)}\ S_I^{+} \qquad \frac{\Gamma \vdash t : Q_n^S}{\Gamma \vdash \pi_j t : Q_n^{S\setminus\{1,\ldots,j\}}}\ S_E\ \left(^{S\subseteq\mathbb{N}^{\leq n}}_{j\leq n}\right)
$$

$$\frac{\Gamma \vdash t : A \ (A \preceq B)}{\Gamma \vdash t : B} \preceq \quad \frac{}{\vdash ?\cdot : \mathbb{B} \Rightarrow \mathbb{B} \Rightarrow \mathbb{B} \Rightarrow \mathbb{B}} \ If \quad \frac{\Gamma, x : \Psi \vdash t : A}{\Gamma \vdash \lambda x : \Psi \ t : \Psi \Rightarrow A} \Rightarrow_I$$

$$\frac{\Gamma \vdash t : \Psi \Rightarrow A \quad \Delta \vdash u : \Psi}{\Gamma, \Delta \vdash tu : A} \Rightarrow_E \quad \frac{\Gamma \vdash t : S(\Psi \Rightarrow A) \quad \Delta \vdash u : S(\Psi)}{\Gamma, \Delta \vdash tu : S(A)} \Rightarrow_{ES}$$

$$\frac{\Gamma \vdash t : A}{\Gamma, x : B \vdash t : A} \ W \quad \frac{\Gamma, x : B, y : B \vdash t : A}{\Gamma, x : B \vdash (x/y)t : A} \ C$$

$$\frac{\Gamma \vdash t : A \quad \Delta \vdash u : B}{\Gamma, \Delta \vdash t \otimes u : A \otimes B} \otimes_I \quad \frac{\Gamma \vdash t : \mathbb{B} \otimes B}{\Gamma \vdash \mathit{head}\ t : \mathbb{B}} \otimes_{Er} \quad \frac{\Gamma \vdash t : \mathbb{B} \otimes B}{\Gamma \vdash \mathit{tail}\ t : B} \otimes_{El}$$

$$\frac{\Gamma \vdash t : S(S(A) \otimes B)}{\Gamma \vdash \Uparrow^{S(A \otimes B)}_{S(S(A) \otimes B)} t : S(A \otimes B)} \Uparrow_r \quad \frac{\Gamma \vdash t : S(A \otimes S(B))}{\Gamma \vdash \Uparrow^{S(A \otimes B)}_{S(A \otimes S(B))} t : S(A \otimes B)} \Uparrow_l$$

$$\frac{\Gamma \vdash \Uparrow^{S(A)}_{S(B)} t : S(A)}{\Gamma \vdash \Uparrow^{S(A)}_{S(B)} \alpha.t : S(A)} \Uparrow^{\alpha} \quad \frac{\Gamma \vdash \Uparrow^{S(A)}_{S(B)} t : S(A) \quad \Delta \vdash \Uparrow^{S(A)}_{S(B)} r : S(A)}{\Gamma, \Delta \vdash \Uparrow^{S(A)}_{S(B)} (t + r) : S(A)} \Uparrow^{+}$$

The new rule S_E types the generalized projection: we force the term to be measured to be typed with a type of the form Q^S_n, and then, after measuring the first j qubits, the new type becomes $Q^{S \setminus \{1,\ldots,j\}}_n$, that is, we remove the superposition mark $S(\cdot)$ from the first j types in the tensor product.

The added rules $\otimes_I$, $\otimes_{E_r}$, $\otimes_{E_l}$ are the standard introduction and eliminations for lists. Rules $\Uparrow_r$ and $\Uparrow_l$ type the castings. The only valid casts are $S(S(A) \otimes B)$ and $S(A \otimes S(B))$ into $S(A \otimes B)$. Rules $\Uparrow^{\alpha}$ and $\Uparrow^{+}$ allow for compositional reasoning. Indeed, casting a linear combination of terms will rewrite to casting each term in the combination.

The rewrite system is given below. It includes all the rules from the previous section plus the rules for tensors: (head) and (tail) to deal with lists, and the typing casts rules, which normalize superpositions to sums of base terms, while update the types.

In the rule (proj), $j \leq m$, $k \leq n$, $\forall i \leq n$, $\forall h \leq m$, $b_{hi} = |0\rangle$ or $b_{ih} = |1\rangle$, if an α_i is absent, it is taken as 1, $\sum_{i=1}^{n} [\alpha_i.](b_{1i} \otimes \cdots \otimes b_{mi})$ is in normal form (hence, $1 \leq n \leq 2^m$), and $P \subseteq \mathbb{N}^{\leq n}$, such that $\forall i \in P$, $\forall h \leq j$, $b_{hi} = b_{hk}$.

Beta	If b has type B and $b \in \mathcal{B}$, then $(\lambda x : B\ t)b \longrightarrow_{(1)} (b/x)t$	(β_{b})	$(\vec{0}_{S(A)} + t) \longrightarrow_{(1)} t$	(neutral)
			$1.t \longrightarrow_{(1)} t$	(unit)
	If u has type $S(\Psi)$, then $(\lambda x : S(\Psi)\ t)u \longrightarrow_{(1)} (u/x)t$	(β_{n})	If t has type A, then $0.t \longrightarrow_{(1)} \vec{0}_{S(A)}$	(zero_{α})
If	$\lvert 1\rangle ?u \cdot v \longrightarrow_{(1)} u$	(if_1)	$\alpha.\vec{0}_{S(A)} \longrightarrow_{(1)} \vec{0}_{S(A)}$	(zero)
	$\lvert 0\rangle ?u \cdot v \longrightarrow_{(1)} v$	(if_0)	$\alpha.(\beta.t) \longrightarrow_{(1)} (\alpha \times \beta).t$	(prod)
Linear distribution	If t has type $B \Rightarrow A$, then $t(u + v) \longrightarrow_{(1)} (tu + tv)$	$(\mathsf{lin}^{+}_{\mathsf{r}})$	$\alpha.(t + u) \longrightarrow_{(1)} (\alpha.t + \alpha.u)$	$(\alpha\mathsf{dist})$
			$(\alpha.t + \beta.t) \longrightarrow_{(1)} (\alpha + \beta).t$	(fact)
	If t has type $B \Rightarrow A$ then $t(\alpha.u) \longrightarrow_{(1)} \alpha.tu$	$(\mathsf{lin}^{\alpha}_{\mathsf{r}})$	$(\alpha.t + t) \longrightarrow_{(1)} (\alpha + 1).t$	(fact^1)
			$(t + t) \longrightarrow_{(1)} 2.t$	(fact^2)
	If t has type $B \Rightarrow A$, then $t\vec{0}_{S(\mathbb{B})} \longrightarrow_{(1)} \vec{0}_{S(A)}$	$(\mathsf{lin}^{0}_{\mathsf{r}})$	$(u + v) =_{AC} (v + u)$	(comm)
			$((u + v) + w) =_{AC} (u + (v + w))$	(assoc)
	$(t + u)v \longrightarrow_{(1)} (tv + uv)$	$(\mathsf{lin}^{+}_{\mathsf{l}})$	If $h \neq u \otimes v$ and $h \in \mathcal{B}$, then $\mathit{head}\ (h \otimes t) \longrightarrow_{(1)} h$	(head)
	$(\alpha.t)u \longrightarrow_{(1)} \alpha.tu$	$(\mathsf{lin}^{\alpha}_{\mathsf{l}})$		
	$\vec{0}_{S(\mathbb{B} \Rightarrow A)}t \longrightarrow_{(1)} \vec{0}_{S(A)}$	$(\mathsf{lin}^{0}_{\mathsf{l}})$	If $h \neq u \otimes v$ and $h \in \mathcal{B}$, then $\mathit{tail}\ (h \otimes t) \longrightarrow_{(1)} t$	(tail)

Vector space axioms: (neutral), (unit), (zero$_{\alpha}$), (zero), (prod), (αdist), (fact), (fact^1), (fact^2), (comm), (assoc)

$\equiv$

Lists: (head), (tail)

Typing casts

$$\Uparrow^{S(A\otimes B)}_{S(S(A)\otimes B)} ((r+s)\otimes u) \longrightarrow_{(1)} (\Uparrow^{S(A\otimes B)}_{S(S(A)\otimes B)} (r\otimes u) + \Uparrow^{S(A\otimes B)}_{S(S(A)\otimes B)} (s\otimes u)) \quad (\mathsf{dist}^{+}_{\mathsf{r}})$$

$$\Uparrow^{S(B\otimes A)}_{S(B\otimes S(A))} (u\otimes (r+s)) \longrightarrow_{(1)} (\Uparrow^{S(B\otimes A)}_{S(B\otimes S(A))} (u\otimes r) + \Uparrow^{S(B\otimes A)}_{S(B\otimes S(A))} (u\otimes s)) \quad (\mathsf{dist}^{+}_{\mathsf{l}})$$

$$\Uparrow^{S(A\otimes B)}_{S(S(A)\otimes B)} ((\alpha.r)\otimes u) \longrightarrow_{(1)} \alpha.\Uparrow^{S(A\otimes B)}_{S(S(A)\otimes B)} (r\otimes u) \quad (\mathsf{dist}^{\alpha}_{\mathsf{r}})$$

$$\Uparrow^{S(B\otimes A)}_{S(B\otimes S(A))} (u\otimes (\alpha.r)) \longrightarrow_{(1)} \alpha.\Uparrow^{S(B\otimes A)}_{S(B\otimes S(A))} (u\otimes r) \quad (\mathsf{dist}^{\alpha}_{\mathsf{l}})$$

$$\Uparrow^{S(A\otimes B)}_{S(S(A)\otimes B)} (\vec{0}_{S(A)}\otimes u) \longrightarrow_{(1)} \vec{0}_{S(A\otimes B)} \quad (\mathsf{dist}^{0}_{\mathsf{r}})$$

$$\Uparrow^{S(B\otimes A)}_{S(B\otimes S(A))} (u\otimes \vec{0}_{S(A)}) \longrightarrow_{(1)} \vec{0}_{S(B\otimes A)} \quad (\mathsf{dist}^{0}_{\mathsf{l}})$$

$$\Uparrow^{S(B\otimes C)}_{S(A)} (t+u) \longrightarrow_{(1)} (\Uparrow^{S(B\otimes C)}_{S(A)} t + \Uparrow^{S(B\otimes C)}_{S(A)} u) \quad (\mathsf{dist}^{+}_{\Uparrow})$$

$$\Uparrow^{S(B\otimes C)}_{S(A)} (\alpha.t) \longrightarrow_{(1)} \alpha.\Uparrow^{S(B\otimes C)}_{S(A)} t \quad (\mathsf{dist}^{\alpha}_{\Uparrow})$$

$$\text{If } u\in\mathcal{B}\text{, then, } \Uparrow^{S(A\otimes B)}_{S(S(A)\otimes B)} (u\otimes v) \longrightarrow_{(1)} u\otimes v \quad (\mathsf{neut}^{\Uparrow}_{\mathsf{r}})$$

$$\text{If } u\in\mathcal{B}\text{, then, } \Uparrow^{S(A\otimes B)}_{S(A\otimes S(B))} (v\otimes u) \longrightarrow_{(1)} v\otimes u \quad (\mathsf{neut}^{\Uparrow}_{\mathsf{l}})$$

Project.

$$\pi_j\left(\sum_{i=1}^{n}[\alpha_i.](b_{1i}\otimes\cdots\otimes b_{mi})\right) \longrightarrow_{(p)} \bigotimes_{h=1}^{j} b_{hk}\otimes\sum_{i\in P}\left(\frac{\alpha_i}{\sqrt{\sum_{i\in P}|\alpha_i|^2}}\right).(b_{j+1,i}\otimes\cdots\otimes b_{mi}) \quad (\mathsf{proj})$$

with $p=\sum_{i\in P}\left(\frac{|\alpha_i|^2}{\sum_{i=1}^{n}|\alpha_i|^2}\right)$

$$\frac{t\longrightarrow_{(p)} u}{tv\longrightarrow_{(p)} uv} \qquad \frac{t\longrightarrow_{(p)} u}{(\lambda x:B\ v)t\longrightarrow_{(p)} (\lambda x:B\ v)u}$$

$$\frac{t\longrightarrow_{(p)} u}{t+v\longrightarrow_{(p)} u+v} \qquad \frac{t\longrightarrow_{(p)} u}{\alpha.t\longrightarrow_{(p)} \alpha u} \qquad \frac{t\longrightarrow_{(p)} u}{\pi_j t\longrightarrow_{(p)} \pi_j u} \qquad \frac{t\longrightarrow_{(p)} u}{t\otimes v\longrightarrow_{(p)} u\otimes v}$$

$$\frac{t\longrightarrow_{(p)} u}{\mathit{head}\ t\longrightarrow_{(p)} \mathit{head}\ u} \qquad \frac{t\longrightarrow_{(p)} u}{\mathit{tail}\ t\longrightarrow_{(p)} \mathit{tail}\ u} \qquad \frac{t\longrightarrow_{(p)} u}{\Uparrow^{S(B)}_{S(A)} t\longrightarrow_{(p)}\Uparrow^{S(B)}_{S(A)} u}$$

The rule (proj) has been updated to account for multiple qubits systems. It normalizes (as in norm 1) the scalars on the obtained term.

The first six rules in the group typing casts—$(\mathsf{dist}^{+}_{\mathsf{r}})$, $(\mathsf{dist}^{\alpha}_{\mathsf{r}})$, and $(\mathsf{dist}^{0}_{\mathsf{r}})$, and their analogous $(\mathsf{dist}^{+}_{\mathsf{l}})$, $(\mathsf{dist}^{\alpha}_{\mathsf{l}})$, and $(\mathsf{dist}^{0}_{\mathsf{l}})$—deal with the distributivity of sums, scalar product and null vector respectively. If we ignore the type cast $\Uparrow^{S(B)}_{S(A)}$ on each rule, these rules are just distributivity rules. For example, rule $(\mathsf{dist}^{+}_{\mathsf{r}})$ acts on the term $(r+s)\otimes u$, distributing the sum with respect to the tensor product, producing $(r\otimes u+s\otimes u)$ (distribution to the right). However, the term $(r+s)\otimes u$ may have type $S(A)\otimes B$, $S(A)\otimes S(B)$ or $S(A\otimes B)$, while, among those, the term $(r\otimes u+s\otimes u)$ can only have type $S(A\otimes B)$. Hence, we cannot reduce the first term to the second without losing subject reduction. Instead, we need to cast the term explicitly to the valid type in order to reduce. Notice that in the previous example it would have been enough to use $\Uparrow^{S(A\otimes B)}_{S(A)\otimes B}$. Indeed, the term $(r+s)\otimes u$ can be typed with $S(A)\otimes B$. However, we prefer the more general $S(S(A)\otimes B)$ and hence to use the same rule when, for example, a sum is given.

The next two rules, $(\mathsf{dist}^{+}_{\Uparrow})$ and $(\mathsf{dist}^{\alpha}_{\Uparrow})$, distribute the cast over sums and scalars. For example $\Uparrow^{S(\mathbb{B}\otimes\mathbb{B})}_{S(S(\mathbb{B})\otimes\mathbb{B})} ((\alpha.|1\rangle)\otimes|0\rangle+(\beta.|0\rangle)\otimes|1\rangle)$ reduces by rule $(\mathsf{dist}^{+}_{\Uparrow})$ to $(\Uparrow^{S(\mathbb{B}\otimes\mathbb{B})}_{S(S(\mathbb{B})\otimes\mathbb{B})} (\alpha.|1\rangle)\otimes|0\rangle+\Uparrow^{S(\mathbb{B}\otimes\mathbb{B})}_{S(S(\mathbb{B})\otimes\mathbb{B})} (\beta.|0\rangle)\otimes|1\rangle)$, and hence, the distributivity rule can act. The last two rules in the group, $(\mathsf{neut}^{\Uparrow}_{\mathsf{r}})$ and $(\mathsf{neut}^{\Uparrow}_{\mathsf{l}})$, remove the cast when it is not needed anymore. For example $\Uparrow^{S(\mathbb{B}\otimes\mathbb{B})}_{S(S(\mathbb{B})\otimes\mathbb{B})} (\alpha.\beta.|0\rangle)\otimes|1\rangle \xrightarrow{(\mathsf{dist}^{\alpha}_{\Uparrow})}_{(1)} \alpha.\Uparrow^{S(\mathbb{B}\otimes\mathbb{B})}_{S(S(\mathbb{B})\otimes\mathbb{B})} (\beta.|0\rangle)\otimes|1\rangle \xrightarrow{(\mathsf{dist}^{\alpha}_{\Uparrow})}_{(1)} \alpha.\beta.\Uparrow^{S(\mathbb{B}\otimes\mathbb{B})}_{S(S(\mathbb{B})\otimes\mathbb{B})} |0\rangle\otimes|1\rangle \xrightarrow{(\mathsf{neut}^{\Uparrow}_{\mathsf{r}})}_{(1)} \alpha.\beta.|0\rangle\otimes|1\rangle$.

The measurement rule (proj) is updated to measure the first j qubits. Hence, a n-qubits in normal form (that is, a sum of tensors of qubits with or without a scalar in front), for example, the term $((2.(|0\rangle \otimes |1\rangle \otimes |1\rangle) + |0\rangle \otimes |1\rangle \otimes |0\rangle) + 3.(|1\rangle \otimes |1\rangle \otimes |1\rangle))$ can be measured and will produce a n-qubits where the first j qubits are the same and the remaining are untouched, with its scalars changed to have norm 1. In this 3-qubits example, measuring the first two can produce either $|0\rangle \otimes |1\rangle \otimes (\frac{2}{\sqrt{5}}.|1\rangle + \frac{1}{\sqrt{5}}.|0\rangle)$ or $|1\rangle \otimes |1\rangle \otimes (1.|1\rangle)$. The probability of producing the first is $\frac{|2|^2}{(|2|^2+|1|^2+|3|^2)} + \frac{|1|^2}{(|2|^2+|1|^2+|3|^2)} = \frac{5}{14}$ and the probability of producing the second is $\frac{|3|^2}{(|2|^2+|1|^2+|3|^2)} = \frac{9}{14}$.

Remark, to conclude, that since the calculus presented in this paper is call-by-base for the functions expecting a non-linear argument, it avoids a well-known problem in others λ-calculi with a linear logic type system including modalities. To illustrate this problem, consider the following typing judgement: $y : S(\mathbb{B}) \vdash (\lambda x : \mathbb{B}\ (x \otimes x))(\pi y) : S(\mathbb{B}) \otimes S(\mathbb{B})$. If we allow to β-reduce this term, we would obtain $(\pi y) \otimes (\pi y)$ which is not typable in the context $y : S(\mathbb{B})$. A standard solution to this problem is illustrated in [7], where the terms that can be cloned are distinguished by a mark, and used in a *let* construction, while non-clonable terms are used in λ abstractions.

Thanks to the explicit casts, the resulting system has the Subject Reduction property (Theorem 2), that is, the typing is preserved by weak-reduction (i.e. reduction on closed terms). The proof of this theorem is not trivial, specially due to the complexity of the system itself. The detailed proof is given in a seven-page long appendix in a preprint submitted to `arXiv:1601.04294`.

Lemma 1 (Substitution lemma). *Let $FV(u) = \emptyset$, then if $\Gamma, x : \Psi \vdash t : A$, $\Delta \vdash u : \Psi$, where if $\Psi = B$ then $u \in \mathcal{B}$, we have $\Gamma, \Delta \vdash (u/x)t : A$.*

Theorem 2 (Subject reduction on closed terms). *For any closed terms t and u and type A, if $t \longrightarrow_{(p)} u$ and $\vdash t : A$, then $\vdash u : A$.*

4 Interpretation

We consider vector spaces equipped with a canonical base, and subsets of such spaces.

Let E and F be two vector spaces with canonical bases $B = \{\vec{b_i} \mid i \in I\}$ and $C = \{\vec{c_j} \mid j \in J\}$. The tensor product $E \otimes F$ of E and F is the vector space of canonical base $\{\vec{b_i} \otimes \vec{c_j} \mid i \in I \text{ and } j \in J\}$, where $\vec{b_i} \otimes \vec{c_j}$ is the ordered pair formed with the vector $\vec{b_i}$ and the vector $\vec{c_j}$. The operation $\otimes$ is extended to the vectors of E and F bilinearly: $(\sum_i \alpha_i \vec{b_i}) \otimes (\sum_j \beta_j \vec{c_j}) = \sum_{ij} \alpha_i \beta_j (\vec{b_i} \otimes \vec{c_j})$.

Let E and F be two vector spaces equipped with bases B and C, and S and T be two subsets of E and F respectively, we define the set $S \times T$, subset of the vector space $E \otimes F$, as follows: $S \times T = \{\vec{u} \otimes \vec{v} \mid \vec{u} \in S, \vec{v} \in T\}$.

Remark that $E \times F$ differs from $E \otimes F$. For instance, if E and F are $\mathbb{C}^2$ equipped with the base $\{\vec{i}, \vec{j}\}$, then $E \times F$ contains $\vec{i} \otimes \vec{i}$ and $\vec{j} \otimes \vec{j}$ but not $\vec{i} \otimes \vec{i} + \vec{j} \otimes \vec{j}$, that is not a tensor product of two vectors of $\mathbb{C}^2$.

Let E be a vector space equipped with a base B, and S a subset of E. We write $\mathcal{S}(S)$ for the vector space over $\mathbb{C}$ generated by the span of S, that is, containing all the linear combinations of elements of S.

Hence, if E and F are two vector spaces of bases B and C then $E \otimes F = \mathcal{S}(B \times C) = \mathcal{S}(E \times F)$.

Let S and T be two sets. We write $S \to T$ for the vector space of formal linear combination of functions from S to T. The set $S \Rightarrow T$ of the functions from S to T is a subset—and even a basis—of this vector space.

Note that if S and T are two sets , then $S \to T = \mathcal{S}(S \Rightarrow T)$.

To each type we associate the subset of some vector space

$\llbracket \mathbb{B} \rrbracket = \{\binom{1}{0}, \binom{0}{1}\}$, a subset of $\mathbb{C}^2$

$\llbracket S(A) \rrbracket = \mathcal{S}\llbracket A \rrbracket$

$\llbracket \Psi \Rightarrow A \rrbracket = \llbracket \Psi \rrbracket \Rightarrow \llbracket A \rrbracket$

$\llbracket A \otimes B \rrbracket = \llbracket A \rrbracket \times \llbracket B \rrbracket$

Remark that $\llbracket S(A \otimes B) \rrbracket = \mathcal{S}(\llbracket A \rrbracket \times \llbracket B \rrbracket) = \llbracket A \rrbracket \otimes \llbracket B \rrbracket$.

If $\Gamma = x_1 : \Psi_1, ..., x_n : \Psi_n$ is a context, then a Γ-valuation is a function ϕ mapping each x_i to $\llbracket \Psi_i \rrbracket$. Notation: $\phi \vDash \Gamma$.

We now would associate to each term t of type A an element $\llbracket t \rrbracket$ of $\llbracket A \rrbracket$. But as our calculus is probabilistic, due to the presence of a measurement operator, we must associate to each term a set of elements of $\llbracket A \rrbracket$.

Let $\Gamma \vdash t : A$ and $\phi \vDash \Gamma$. We define the interpretation of t, $\llbracket t \rrbracket_\phi$ as follows.

$\llbracket x \rrbracket_\phi = \phi x$

$\llbracket \lambda x : \Psi\ t \rrbracket_\phi = \{f \mid \forall a \in \llbracket \Psi \rrbracket, fa \in \llbracket t \rrbracket_{\phi, x \mapsto \llbracket \Psi \rrbracket}\}$

$\llbracket |0\rangle \rrbracket_\phi = \{\binom{1}{0}\} \quad ; \quad \llbracket |1\rangle \rrbracket_\phi = \{\binom{0}{1}\}$

$\llbracket t \otimes u \rrbracket_\phi = \llbracket t \rrbracket_\phi \times \llbracket u \rrbracket_\phi$

$\llbracket (t + u) \rrbracket_\phi = \{a + b \mid a \in \llbracket t \rrbracket_\phi \text{ and } b \in \llbracket u \rrbracket_\phi\}$

$\llbracket \alpha.t \rrbracket_\phi = \{\alpha a \mid a \in \llbracket t \rrbracket_\phi\}$

$\llbracket \vec{0}_{S(B)} \rrbracket_\phi = \{\vec{0}\}$, the null vector of the vector space $\llbracket S(B) \rrbracket$

$$\llbracket tu \rrbracket_\phi = \begin{cases} \{\sum_{i \in I} \alpha_i g_i(a) \mid \sum_{i \in I} \alpha_i g_i \in \llbracket t \rrbracket_\phi, a \in \llbracket u \rrbracket_\phi & \text{If } \Gamma \vdash t : \Psi \Rightarrow A \\ \{\sum\limits_{i \in I, j \in J} \alpha_i \beta_j g_i(c_j) \mid \sum\limits_{i \in I} \alpha_i g_i \in \llbracket t \rrbracket_\phi, \sum\limits_{j \in J} \beta_j c_j \in \llbracket u \rrbracket_\phi\} & \text{If } \Gamma \vdash t : S(\Psi \Rightarrow A) \end{cases}$$

$$\llbracket \pi_j t \rrbracket_\phi = \{\bigotimes_{h=1}^{j} b_{hk} \otimes \sum_{i \in P} \left(\frac{\alpha_i}{\sqrt{\sum_{i \in P} |\alpha_i|^2}}\right)(b_{j+1,i} \otimes \cdots \otimes b_{mi}) \mid \forall i \in P, \forall h, b_{hi} = b_{hk}\}$$

where $\llbracket t \rrbracket_\phi = \{\sum_{i=1}^{n} \alpha_i (b_{1i} \otimes \cdots \otimes b_{mi})\}$ with $b_{ih} = \binom{0}{1}$ or $\binom{1}{0}$

$\llbracket ?\cdot \rrbracket_\phi = \{f \mid \forall a, b, c \in \llbracket \mathbb{B} \rrbracket, fabc = b \text{ if } a = \binom{0}{1} \text{ and } fabc = c \text{ if } a = \binom{1}{0}\}$

$\llbracket \mathit{head}\ t \rrbracket_\phi = \{a_1 \mid a_1 \otimes \cdots \otimes a_n \in \llbracket t \rrbracket_\phi, a_1 \in \llbracket \mathbb{B} \rrbracket\}$

$\llbracket \mathit{tail}\ t \rrbracket_\phi = \{a_2 \otimes \cdots \otimes a_n \mid a_1 \otimes \cdots \otimes a_n \in \llbracket t \rrbracket_\phi, a_1 \in \llbracket \mathbb{B} \rrbracket\}$

$\llbracket \Uparrow_{S(A)}^{S(B \otimes C)} t \rrbracket_\phi = \llbracket t \rrbracket_\phi$

Lemma 3. *If $A \preceq B$, then $\llbracket A \rrbracket \subseteq \llbracket B \rrbracket$.*

Lemma 4. *If $\Gamma \vdash t : A$ and $\phi, x \mapsto S, y \mapsto S \vDash \Gamma$ then $\llbracket t \rrbracket_{\phi, x \mapsto S, y \mapsto S} = \llbracket (x/y)t \rrbracket_{\phi, x \mapsto S}$.*

Theorem 5. *If $\Gamma \vdash t : A$, and $\phi \vDash \Gamma$ then $\llbracket t \rrbracket_\phi \subseteq \llbracket A \rrbracket$.*

Theorem 6. *If $\Gamma \vdash t : A$, $\phi \vDash \Gamma$, and $t \longrightarrow_{(p_i)} r_i$, with $\sum_i p_i = 1$, then $\llbracket t \rrbracket_\phi = \bigcup_i \llbracket r_i \rrbracket_\phi$.*

5 Example: The Teleportation Algorithm

In this section we show that our language is expressive enough to express the Teleportation algorithm. The circuit for this algorithm is given in Figure 1. The Hadamard gate (H) produces $\frac{1}{\sqrt{2}}.(|0\rangle + |1\rangle)$ when applied to $|0\rangle$ and $\frac{1}{\sqrt{2}}.(|0\rangle - |1\rangle)$ when applied to $|1\rangle$. Hence, it can be implemented with the if-then-else construction: $\mathsf{H} = \lambda x : \mathbb{B}\ \frac{1}{\sqrt{2}}.(|0\rangle + (x?(-|1\rangle)\cdot|1\rangle))$. Notice that the abstracted variable has a base type (i.e. non-linear). Hence, if H is applied to a superposition, say $(\alpha.|0\rangle + \beta.|1\rangle)$, it reduces, as expected, in the following way: $\mathsf{H}(\alpha.|0\rangle + \beta.|1\rangle) \xrightarrow{(\mathsf{lin}_+^{\mathsf{r}})}_{(1)} (\mathsf{H}\alpha.|0\rangle + \mathsf{H}\beta.|1\rangle) \xrightarrow{(\mathsf{lin}_{\mathsf{b}}^{\alpha})^2}_{(1)} (\alpha.\mathsf{H}|0\rangle + \beta.\mathsf{H}|1\rangle)$, and then is applied to the base terms. The *cnot* gate, which applies *not* to the second qubit only when the first qubit is $|1\rangle$, can be implemented with an if-then-else construction as follows: $\mathsf{cnot} = \lambda x : \mathbb{B} \otimes \mathbb{B}\ ((\mathit{head}\ x) \otimes ((\mathit{head}\ x)?(\mathsf{not}\ (\mathit{tail}\ x))\cdot(\mathit{tail}\ x)))$. We define H_1^3 to apply H to the first qubit of a three-qubit system. $\mathsf{H}_1^3 = \lambda x : \mathbb{B} \otimes \mathbb{B} \otimes \mathbb{B}\ ((\mathsf{H}\ (\mathit{head}\ x)) \otimes (\mathit{tail}\ x))$. In addition, we need to apply *cnot* to the two first qubits, so we define cnot_{12}^3 as $\mathsf{cnot}_{12}^3 = \lambda x : \mathbb{B} \otimes \mathbb{B} \otimes \mathbb{B}\ ((\mathsf{cnot}\ (\mathit{head}\, x \otimes (\mathit{head}\ \mathit{tail}\, x))) \otimes (\mathit{tail}\ \mathit{tail}\, x))$. The Z gate returns $|0\rangle$ when it receives $|0\rangle$, and $-|1\rangle$ when it receives $|1\rangle$. Hence, it can be implemented by: $\mathsf{Z} = \lambda x : \mathbb{B}\ (x?(-|1\rangle)\cdot|0\rangle)$. The Bob side of the algorithm will apply Z and/or *not* according to the bits it receives from Alice. Hence, for any $\vdash \mathsf{U} : \mathbb{B} \Rightarrow S(\mathbb{B})$ or $\vdash \mathsf{U} : \mathbb{B} \Rightarrow \mathbb{B}$, we define $\mathsf{U}^{(b)}$ to be the function which depending on the value of a base qubit b applies the U gate or not: $\mathsf{U}^{(b)} = (\lambda x : \mathbb{B}\ \lambda y : \mathbb{B}\ (x?\mathsf{U}y\cdot y))\ b$. Alice and Bob parts of the algorithm are defined separately. $\mathsf{Alice} = \lambda x : S(\mathbb{B}) \otimes S(\mathbb{B} \otimes \mathbb{B})(\pi_2(\Uparrow_{S(S(\mathbb{B})\otimes\mathbb{B}\otimes\mathbb{B})}^{S(\mathbb{B}\otimes\mathbb{B}\otimes\mathbb{B})} \mathsf{H}_1^3\ (\mathsf{cnot}_{12}^3 \Uparrow_{S(\mathbb{B}\otimes S(\mathbb{B}\otimes\mathbb{B}))}^{S(\mathbb{B}\otimes\mathbb{B}\otimes\mathbb{B})} \Uparrow_{S(S(\mathbb{B})\otimes S(\mathbb{B}\otimes\mathbb{B}))}^{S(\mathbb{B}\otimes S(\mathbb{B}\otimes\mathbb{B}))} x)))$. Notice that before passing to cnot_{12}^3 the parameter of type $S(\mathbb{B}) \otimes S(\mathbb{B} \otimes \mathbb{B})$, we need to fully develop the term using the two casts, and again, after the Hadamard gate. Bob side is implemented by $\mathsf{Bob} = \lambda x : \mathbb{B} \otimes \mathbb{B} \otimes \mathbb{B}\ (\mathsf{Z}^{(\mathit{head}\ x)}(\mathsf{not}^{(\mathit{head}\ \mathit{tail}\ x)}\ (\mathit{tail}\ \mathit{tail}\ x)))$.

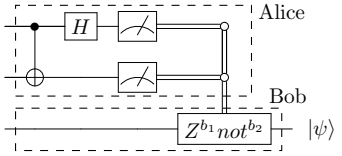

Fig. 1. Teleportation circuit

The teleportation is applied to an arbitrary qubit and to the following Bell state $\beta_{00} = (\frac{1}{\sqrt{2}}.|0\rangle \otimes |0\rangle + \frac{1}{\sqrt{2}}.|1\rangle \otimes |1\rangle)$ and it is defined by $\mathsf{Teleportation} = \lambda q : S(\mathbb{B})\ (\mathsf{Bob}(\Uparrow_{S(\mathbb{B}\otimes\mathbb{B}\otimes S(\mathbb{B}))}^{S(\mathbb{B}\otimes\mathbb{B}\otimes\mathbb{B})} \mathsf{Alice}\ (q \otimes \beta_{00})))$.

This term is typed, as expected, by: $\vdash \mathsf{Teleportation} : S(\mathbb{B}) \Rightarrow S(\mathbb{B})$ and applying the teleportation to any superposition $(\alpha.|0\rangle + \beta.|1\rangle)$ will reduce, as expected, to $(\alpha.|0\rangle + \beta.|1\rangle)$.

6 Conclusion

In this paper we have proposed a way to unify logic-linear and algebraic-linear quantum λ-calculi, by interpreting λ-terms as linear functions when they expect duplicable data and as non-linear ones when they do not, and illustrated this idea with the definition of a calculus.

This calculus is first-order in the sense that variables do not have functional types. In a higher-order version we should expect abstractions to be clonable. But, allowing cloning abstractions allows cloning superpositions, by hiding them inside. For example, $\lambda x : \mathbb{B} \Rightarrow \mathbb{B}\ (\frac{1}{\sqrt{2}}.|0\rangle + \frac{1}{\sqrt{2}}.|1\rangle)$. It has been argued [4, 5]

that what is cloned is not the superposition but a function that creates the superposition, because we had no way there to create such an abstraction from an arbitrary superposition. The situation is different in the calculus presented in this paper as the term $\lambda x : S(\mathbb{B})\ \lambda y : \mathbb{B}\ x$ precisely takes any term t of type $S(\mathbb{B})$ and returns the term $\lambda y : \mathbb{B}\ t$. So, a cloning machine could be constructed by encapsulating any superposition t under a lambda, which transform it into a basis term, so a clonable term. Extending this calculus to the higher-order will require characterizing precisely the abstractions that can be taken as arguments, not allowing to duplicate functions creating superpositions.

Acknowledgements

We would like to thank Eduardo Bonelli, Luca Paolini, Simona Ronchi della Rocca and Luca Roversi for interesting comments and suggestions.

References

1. Abramsky, S.: Computational interpretations of linear logic. Theoretical Computer Science 111(1), 3–57 (1993)
2. Altenkirch, T., Grattage, J.: A functional quantum programming language. In: Proceedings of LICS 2005. pp. 249–258. IEEE (2005)
3. Arrighi, P., Díaz-Caro, A.: A System F accounting for scalars. Logical Methods in Computer Science 8(1:11) (2012)
4. Arrighi, P., Díaz-Caro, A., Valiron, B.: The vectorial lambda-calculus. Information and Computation 254(1), 105–139 (2017)
5. Arrighi, P., Dowek, G.: Lineal: A linear-algebraic lambda-calculus. Logical Methods in Computer Science 13(1:8) (2017)
6. Assaf, A., Díaz-Caro, A., Perdrix, S., Tasson, C., Valiron, B.: Call-by-value, call-by-name and the vectorial behaviour of the algebraic λ-calculus. Logical Methods in Computer Science 10(4:8) (2014)
7. Barber, A.: Dual intuitionistic linear logic. Tech. Rep. ECS-LFCS-96-347, The Laboratory for Foundations of Computer Science, University of Edinburgh (1996)
8. Díaz-Caro, A., Petit, B.: Linearity in the non-deterministic call-by-value setting. In: Ong, L., de Queiroz, R. (eds.) Proceedings of WoLLIC 2012. LNCS, vol. 7456, pp. 216–231 (2012)
9. Girard, J.Y.: Linear logic. Theoretical Compututer Science 50, 1–102 (1987)
10. Green, A.S., Lumsdaine, P.L., Ross, N.J., Selinger, P., Valiron, B.: Quipper: a scalable quantum programming language. ACM SIGPLAN Notices (PLDI'13) 48(6), 333–342 (2013)
11. Pagani, M., Selinger, P., Valiron, B.: Applying quantitative semantics to higher-order quantum computing. ACM SIGPLAN Notices (POPL'14) 49(1), 647–658 (2014)
12. Peterson, G.E., Stickel, M.E.: Complete sets of reductions for some equational theories. Journal of the ACM 28(2), 233–264 (1981)
13. Selinger, P., Valiron, B.: Quantum lambda calculus. In: Gay, S., Mackie, I. (eds.) Semantic Techniques in Quantum Computation, chap. 9, pp. 135–172. Cambridge University Press (2009)
14. Zorzi, M.: On quantum lambda calculi: a foundational perspective. Mathematical Structures in Computer Science 26(7), 1107–1195 (2016)