
On quantum lambda calculi: a foundational perspective

MARGHERITA ZORZI

Mathematical Structures in Computer Science / *FirstView* Article / September 2015, pp 1 - 89
DOI: 10.1017/S0960129514000425, Published online: 17 November 2014

Math. Struct. in Comp. Science: page 1 of 89.
doi:10.1017/S0960129514000425

On quantum lambda calculi: a foundational perspective

MARGHERITA ZORZI[†]

Dipartimento di Informatica, Università degli Studi di Verona, Strada Le Grazie 15, 37134, Verona, Italy
Email: margherita.zorzi@univr.it, marghi.zorzi@gmail.com

Received 15 November 2011; revised 30 June 2014

In this paper, we propose an approach to quantum λ-calculi. The 'quantum data-classical control' paradigm is considered. Starting from a measurement-free untyped quantum λ-calculus called Q, we will study standard properties such as confluence and subject reduction, and some good quantum properties. We will focus on the expressive power, analysing the relationship with other quantum computational models. Successively, we will add an explicit measurement operator to Q. On the resulting calculus, called Q^*, we will propose a complete study of reduction sequences regardless of their finiteness, proving confluence results. Moreover, since the stronger motivation behind quantum computing is the research of new results in computational complexity, we will also propose a calculus which captures the three classes of quantum polytime complexity, showing an ICC-like approach in the quantum setting.

1. Introduction

Quantum computing is a computational paradigm based on abstract computational models which obey (totally or in part) to quantum mechanics' physical laws. Quantum computing is a promising paradigm as testified by a number of interesting results about computational complexity (Grover 1999; Shor 1994, 1997). The seminal ideas behind quantum computing are due to Feynman. He posed a simple but interesting question: *is a classical computer able to simulate a generic physical system?* The problem can be formulated in this way: given a physical system of N two-state (spin-$\frac{1}{2}$-like) interacting particles, can it be fully described by a function of the shape $\psi(x_1, \ldots, x_N, t)$, where t represents time? The answer to Feynman's question is the following: the full description of a classical system can be efficiently simulated by a classical computer with polynomial slowdown. In the case of a quantum system, the slowdown is exponential.

The birth of quantum computing coincided therefore with the attempt to understand how a computational device could be created in order to emulate an arbitrary physical system. Feynman's idea was resumed in Benioff (1980): he stated the first relationships

[†] The first version of this work was supported by the project ANR-08-BLANC-0211-01 'COMPLICE' (Implicit Computational Complexity, Concurrency and Extraction), Laboratoire d'Informatique de Paris Nord (LIPN), UMR CNRS 7030, Institut Galilée - Université Paris-Nord, 99 avenue Jean-Baptiste Clément, 93430, Villetaneuse.

between the quantum mechanical model and a mathematical computational model such as Turing machines. After these important introductory investigations, the first concrete proposal for a quantum abstract computer is due to Deutsch, who introduced *quantum Turing machines* (Deutsch 1985). He started from the probabilistic version of the Church–Turing thesis, attempting to understand what physical basis were required in order to define stronger and stronger versions of the thesis. Deutsch also introduced the first quantum algorithm, subsequently called *Deutsch's algorithm* (see Section 5, Example 8): we can affirm that all the algorithmic results in the field of quantum computing are partially inspired by Deutsch's intuition.

Starting from Deutsch's fundamental works, Bernstein and Vazirani defined the *quantum universal Turing machine*, giving an accurate set of foundational results about the computational model and its mathematical setting (Bernstein and Vazirani 1997). Bernstein and Vazirani also developed a complexity theory for quantum computing, revisiting and extending the results of the classical and the probabilistic cases.

Since quantum Turing machines, other models have been defined. In 1989, Deutsch introduced *quantum circuit families* and this topic was subsequently developed in Yao (1993), where the author further extended the quantum complexity theory. In Nishimura and Ozawa (2009), the authors established the 'perfect equivalence' between quantum Turing machines and quantum circuit families (this foundational result will be exploited in Section 8, see also Appendix A.5). Another quantum model, the quantum random access machine (QRAM), has been defined in Knill (1996); a QRAM is a classically controlled machine enriched with a quantum device. On the grounds of the QRAM model (see also Lanzagorta and Uhlmann (2009)), Selinger defined the first functional language based on the so-called *quantum data-classical control* paradigm, giving a statically typed language whose semantics, provided in terms of super-operators (Preskill 2006), are fully abstract (Selinger 2004). The idea behind this 'hybrid' computational model is relevant to the present work: in fact, the λ-calculi proposed in this paper are based on Selinger's approach.

The quantum computational paradigm had a strong impact on the notion of 'computational feasibility' of problems. The most surprising result is due to Shor, which proved that the *factorization of integers* and the *discrete logarithm* problems (for which nowadays an efficient classical algorithm is still unknown) could be efficiently solved, namely in polynomial time, by a quantum computer (Shor 1994, 1997). Shor's algorithm for prime factorization catalyzed the interest of the scientific community into the quantum computing research. Other important quantum algorithms are Grover's quantum algorithm for efficient data search and the polytime quantum algorithm for the so-called Simon's problem (Nakahara and Ohmi 2008).

On these bases, several attempts to define quantum programming languages, either imperative or functional, have been proposed over the last 15 years. On one hand, the design of quantum programming languages is strongly oriented to the implementation of quantum algorithms for a hoped evolution of technology towards a quantum architecture. On the other hand, the definition of functional paradigmatic languages or functional calculi can be a good instrument in order to investigate theoretical aspects of quantum computing, in particular, the foundational basis of quantum computability.

The aim of this paper is to provide an overview on a possible approach to quantum functional calculi. The results stated and proved in Dal Lago *et al.* (2009), Zorzi (2009), Dal Lago *et al.* (2010) and Dal Lago *et al.* (2011) are here revisited *a posteriori* in an unified setting.

At the same time, we hope this paper should be a 'gentle' introduction to quantum computing's mathematical framework and basic concepts to everyone which, starting from a good background on lambda calculus and linear logic, is interested to an employment of these familiar topics in a non-classical computational setting.

1.1. *Synopsis*

The paper is structured as follows:

— In Sections 2 and 3, we will introduce the mathematical framework of quantum computing and quantum computing bases respectively. The expert reader may skip this introductory part. The non-expert reader could also read Appendix A before Section 5.
— Section 4 is devoted to a discussion about quantum computing peculiarities w.r.t. classical and probabilistic computational paradigms.
— In Section 5, the quantum λ-calculus Q is introduced and studied.
— In Section 6, we extend Q with an explicit measurement operator, obtaining the calculus Q^*.
— Section 7 is an 'intermezzo' about quantum functional languages.
— In Section 8, an implicit polytime quantum λ-calculus called SQ is proposed.
— In Appendix A, we briefly recall quantum computational models and polynomial time quantum complexity classes (the expert reader may skip this part). In Appendix B, some detailed proofs about a measurement operator defined in Section 2, and used in Section 6, are proposed.

2. Mathematical framework of quantum computing (in a nutshell)

The most important concepts introduced by quantum mechanics, such as superposition and entanglement (see Section 3), are essentially described in terms of properties of Hilbert spaces. We recall here only essential notions. For the basic definitions of inner product spaces, Hilbert basis, tensor product, Hilbert spaces and for an exhaustive treatment of these topics, see e.g. Birkhoff and Mac Lane (1967) and Roman (2008).

2.1. *The Hilbert space $\ell^2(S)$*

In this paper, we will deal with particular cases of the Hilbert space $\ell^2(S)$.

Let $\mathcal{S}$ be a finite or countable set and let $\ell^2(\mathcal{S})$ be the set of square summable functions

$$\left\{ \phi \mid \phi : \mathcal{S} \to \mathbb{C}, \sum_{s \in \mathcal{S}} |\phi(s)|^2 < \infty \right\}$$

equipped with

1. an inner sum $+ : \ell^2(\mathcal{S}) \times \ell^2(\mathcal{S}) \to \ell^2(\mathcal{S})$
   defined by $(\phi + \psi)(s) = \phi(s) + \psi(s)$;
2. a multiplication by a scalar $\cdot : \mathbb{C} \times \ell^2(\mathcal{S}) \to \ell^2(\mathcal{S})$
   defined by $(c \cdot \phi)(s) = c \cdot (\phi(s))$;
3. an inner product[†] $\langle \cdot | \cdot \rangle : \ell^2(\mathcal{S}) \times \ell^2(\mathcal{S}) \to \mathbb{C}$
   defined by $\langle \phi | \psi \rangle = \sum_{s \in \mathcal{S}} \phi(s)^* \psi(s)$,

it is easy to show that $\ell^2(\mathcal{S})$ is an Hilbert space.

The inner product $\langle \cdot | \cdot \rangle$ induces a norm defined as $||\psi|| = \langle \psi | \psi \rangle^{1/2}$ for each vector ψ in $\ell^2(\mathcal{S})$.

A vector in $\ell^2(\mathcal{S})$ is *normalized* if its norm is 1.

Definition 1 (quantum state). A *quantum state*, or *quantum register* is any normalized vector in $\ell^2(\mathcal{S})$.

Notation 1. In the following we will use the 'Bra/Ket-notation', introduced by Paul Dirac. Given a Hilbert space $\mathcal{H}$, a *ket* $|\phi\rangle$ indicates a generic element (column vector) of $\mathcal{H}$. Kets like $|\phi\rangle$ are typically used to describe quantum state. The matching $\langle \psi |$ is called *bra*, and denotes the conjugate transpose of $|\phi\rangle$.

The set $\mathcal{B}(\mathcal{S}) = \{|s\rangle : s \in \mathcal{S}\}$, where $|s\rangle : \mathcal{S} \to \mathbb{C}$, is defined by

$$|s\rangle(s') = \begin{cases} 1 & \text{if } s = s' \\ 0 & \text{if } s \neq s' \end{cases}$$

$\mathcal{B}(\mathcal{S})$ is a Hilbert basis of $\ell^2(\mathcal{S})$, usually called the *computational basis* in the literature.

2.2. *Unitary operators*

In quantum computing, the quantum computational steps are represented by linear transformations between normalized vectors in $\ell^2(S)$. Unitary operators are one of the most important classes of operators involved in the mathematical description of quantum mechanics. For our scope, we can restrict definitions to the finite-dimensional case.

Definition 2 (unitary operators – finite-dimensional case). Let $\mathcal{H}$ be a finite-dimensional Hilbert space, and let $\mathbf{U} : \mathcal{H} \to \mathcal{H}$ be a linear transformation. The *adjoint* of $\mathbf{U}$ is the unique linear transformation $\mathbf{U}^\dagger : \mathcal{H} \to \mathcal{H}$ such that for all ϕ, ψ
$\langle \mathbf{U}\phi, \psi \rangle = \langle \phi, \mathbf{U}^\dagger \psi \rangle$. If $\mathbf{U}^\dagger \mathbf{U}$ is the identity, we say that $\mathbf{U}$ is a *unitary operator*.

The tensor product of unitary operators is defined as follows:

[†] In order the inner product definition to make sense, we should prove that the sum $\sum_{s \in \mathcal{S}} \phi(s)^* \psi(s)$ converges, see Roman (2008).

Definition 3 (tensor product of unitary operators). Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be finite-dimensional Hilbert spaces and let $\mathbf{U} : \mathcal{H}_1 \to \mathcal{H}_1$ and $\mathbf{W} : \mathcal{H}_2 \to \mathcal{H}_2$ be two unitary operators. The linear unitary operator $\mathbf{U} \otimes \mathbf{W} : \mathcal{H}_1 \otimes \mathcal{H}_2 \to \mathcal{H}_1 \otimes \mathcal{H}_2$ is defined by

$$(\mathbf{U} \otimes \mathbf{W})(\phi \otimes \psi) = (\mathbf{U}\phi) \otimes (\mathbf{W}\psi)$$

with $\phi \in \mathcal{H}_1$ and $\psi \in \mathcal{H}_2$.

For details, see Conway (1990) and Roman (2008).

2.3. *Two important finite-dimensional Hilbert spaces*

In the rest of the paper and in the literature, the following spaces are extensively used.

The space $\ell^2(\{0,1\}^n)$

Let $S = \{0,1\}^n$, i.e. S is the set of finite binary strings of length n. The Hilbert space $\mathcal{H}(S)$ is the standard space used in the field of quantum computing. This kind of space is useful to describe bits, qubits and quantum registers.
For example, let us consider $S = \{0,1\}^2$. The computational basis of $\ell^2(S)$ is $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, and a generic quantum register may be expressed, in the computational basis, as $\alpha_1|00\rangle + \alpha_2|01\rangle + \alpha_3|10\rangle + \alpha_4|11\rangle$, where $\sum_i |\alpha_i|^2 = 1$.

The space $\mathcal{H}(\mathcal{V})$

Let $\mathcal{V}$ be a set of names and let $S = \{f | f : \mathcal{V} \to \{0,1\}\}$, i.e. S is the set of classical valuations of $\mathcal{V}$ into the set $\{0,1\}$. $\ell^2(S)$ is a Hilbert space of dimension $2^{\#\mathcal{V}}$.

Notation 2. In the following we will shorten $\ell^2(\{0,1\}^{\mathcal{V}})$ with $\mathcal{H}(\mathcal{V})$.

This space is useful to describe quantum registers when we want to assign names to qubits, without reference to their ordinal position (as it usually happens in the literature and in the calculi we propose in this paper). By definition, the set of quantum registers are normalized vectors of $\mathcal{H}(\mathcal{V})$, i.e. a quantum register will be a function $\phi : \{0,1\}^{\mathcal{V}} \to \mathbb{C}$ such that $\sum_{f \in \{0,1\}^{\mathcal{V}}} |\phi(f)|^2 = 1$ (normalization condition). The space $\mathcal{H}(\mathcal{V})$ is equipped with the *standard* or (*computational*) orthonormal[†] basis $\mathcal{B}(\mathcal{V}) = \{|f\rangle : f \in \{0,1\}^{\mathcal{V}}\}$.

Example 1. The standard basis of the space $\mathcal{H}(\{p,q\})$ is $\{|p \mapsto 0, q \mapsto 0\rangle, |p \mapsto 0, q \mapsto 1\rangle, |p \mapsto 1, q \mapsto 0\rangle, |p \mapsto 1, q \mapsto 1\rangle\}$.

Let $\mathcal{V}'$ and $\mathcal{V}''$ be two sets of names such that $\mathcal{V}' \cap \mathcal{V}'' = \varnothing$. With $\mathcal{H}(\mathcal{V}') \otimes \mathcal{H}(\mathcal{V}'')$ we denote the tensor product (defined in the usual way) of $\mathcal{H}(\mathcal{V}')$ and $\mathcal{H}(\mathcal{V}'')$. If $\mathcal{B}(\mathcal{V}') = \{|f_i\rangle : 0 \leqslant i < 2^n\}$ and $\mathcal{B}(\mathcal{V}'') = \{|g_j\rangle : 0 \leqslant j < 2^m\}$ are the standard

[†] Given an inner product space V, a nonempty set U of vectors is an *orthogonal* set if for all $u_i, u_j \in U$, if $u_i \neq u_j$ then $u_i \perp u_j$, i.e. $\langle u_i, u_j \rangle = 0$. If each u_i is also a normalized vector, then U is an *orthonormal* set (for details, see Conway (1990); Roman (2008)).

bases respectively of $\mathcal{H}(\mathcal{V}')$ and $\mathcal{H}(\mathcal{V}'')$, then $\mathcal{H}(\mathcal{V}') \otimes \mathcal{H}(\mathcal{V}'')$ is equipped with the basis $\{|f_i\rangle \otimes |g_j\rangle : 0 \leqslant i < 2^n, 0 \leqslant j < 2^m\}$. We will abbreviate $|f\rangle \otimes |g\rangle$ with $|f, g\rangle$.
If $\mathcal{V}$ is a set of names, then $I_{\mathcal{V}}$ is the identity on $\mathcal{H}(\mathcal{V})$, which is clearly unitary.
Moreover, it is easy to show that if $\mathcal{V}' \cap \mathcal{V}'' = \varnothing$ then there is a standard *isomorphism* i_s:

$$\mathcal{H}(\mathcal{V}') \otimes \mathcal{H}(\mathcal{V}'') \overset{i_s}{\simeq} \mathcal{H}(\mathcal{V}' \cup \mathcal{V}'').$$

In the rest of the paper, we will assume to work up to such an isomorphism (which holds even if $\mathcal{V}'$ or $\mathcal{V}''$ are empty). In particular, if $\mathcal{Q} \in \mathcal{H}(\mathcal{V})$, $r \notin \mathcal{V}$ and $|r \mapsto c\rangle \in \mathcal{H}(\{r\})$ then $\mathcal{Q} \otimes |r \mapsto c\rangle$ will denote the element $i_s(\mathcal{Q} \otimes |r \mapsto c\rangle) \in \mathcal{H}(\mathcal{V} \cup \{r\})$.
If $\mathcal{Q}' \in \mathcal{H}(\mathcal{V}')$ and $\mathcal{Q}'' \in \mathcal{H}(\mathcal{V}'')$ are two quantum registers, with a little abuse of language (authorized by the isomorphism defined above) we will say that $\mathcal{Q}' \otimes \mathcal{Q}''$ is a quantum register in $\mathcal{H}(\mathcal{V}' \cup \mathcal{V}'')$.
The smallest quantum space $\mathcal{H}(\varnothing)$ (isomorphic to the field $\mathbb{C}$), is called an *empty quantum register*; it is nothing more than a unitary element of $\mathbb{C}$ (i.e. a complex number c such that $|c| = 1$). We chose the scalar number 1 as the canonical empty quantum register. In particular, the number 1 also represents the computational basis of $\mathcal{H}(\varnothing)$.
We will sometimes use a 'compact' representation of quantum states. Let $\mathcal{V}$ be a set of names with cardinality $n \geqslant 1$. Moreover, let $\mathcal{Q} \in \mathcal{H}(\mathcal{V})$ and let $r \in \mathcal{V}$. Each state $\mathcal{Q}$ may be represented as

$$\mathcal{Q} = \sum_{i=1}^{2^{n-1}} \alpha_i |r \mapsto 0\rangle \otimes b_i + \sum_{i=1}^{2^{n-1}} \beta_i |r \mapsto 1\rangle \otimes b_i,$$

where $\{b_i\}_{i \in [1,2^{n-1}]}$ is the computational basis of $\mathcal{H}(\mathcal{V} - \{r\})$. Please note that if $\mathcal{V} = \{r\}$, then $\mathcal{Q} = \alpha|r \mapsto 0\rangle \otimes 1 + \beta|r \mapsto 1\rangle \otimes 1$, that is, via the previously stated isomorphism, $\alpha|r \mapsto 0\rangle + \beta|r \mapsto 1\rangle$.

Other definitions are now in order.
Let $u \in \mathcal{H}(\{0,1\}^n)$ be the quantum register $u = \alpha_1|0 \ldots 0\rangle + \cdots + \alpha_{2^n}|1 \ldots 1\rangle$ and let $\langle q_1, \ldots, q_n\rangle$ be a sequence of names. $u^{\langle q_1,\ldots,q_n\rangle}$ is the quantum register in $\mathcal{H}(\{q_1, \ldots, q_n\})$ defined by $u^{\langle q_1,\ldots,q_n\rangle} = \alpha_1|q_1 \mapsto 0, \ldots, q_n \mapsto 0\rangle + \cdots + \alpha_{2^n}|q_1 \mapsto 1, \ldots, q_n \mapsto 1\rangle$.
Let $\mathbf{U} : \mathcal{H}(\mathcal{V}) \to \mathcal{H}(\mathcal{V})$ be an operator and let $\langle q_1, \ldots, q_n\rangle$ be any sequence of distinguished names in $\mathcal{V}$. Considering the bijection between $\{0,1\}^n$ and $\{0,1\}^{\{q_1,\ldots,q_n\}}$, $\mathbf{U}$ and $\langle q_1, \ldots, q_n\rangle$ induce an operator $\mathbf{U}_{\langle q_1,\ldots,q_n\rangle} : \mathcal{H}(\{q_1, \ldots, q_n\}) \to \mathcal{H}(\{q_1, \ldots, q_n\})$ defined as follows: if $|f\rangle = |q_{j_1} \mapsto b_{j_1}, \ldots, q_{j_n} \mapsto b_{j_n}\rangle$ is an element of the orthonormal basis of $\mathcal{H}(\{q_1, \ldots, q_n\})$, then

$$\mathbf{U}_{\langle q_1,\ldots,q_n\rangle}|f\rangle \overset{def}{=} (\mathbf{U}|b_1, \ldots, b_n\rangle)^{\langle q_1,\ldots,q_n\rangle},$$

where $q_{j_i} \mapsto b_{j_i}$ means that to the qubit named q_{j_i} we associate the element b_{j_i} of the basis.
Let $\mathcal{V}' = \{q_{j_1}, \ldots, q_{j_k}\} \subseteq \mathcal{V}$. We naturally extend (by suitable standard isomorphisms) the unitary operator $\mathbf{U}_{\langle q_{j_1},\ldots,q_{j_k}\rangle} : \mathcal{H}(\mathcal{V}') \to \mathcal{H}(\mathcal{V}')$ to the unitary operator $\mathbf{U}_{\langle\langle q_{j_1},\ldots,q_{j_k}\rangle\rangle}$:

$\mathcal{H}(\mathcal{V}) \to \mathcal{H}(\mathcal{V})$ that acts as the identity on variables not in $\mathcal{V}'$ and as $\mathbf{U}_{\langle q_{j_1},\ldots,q_{j_k}\rangle}$ on variables in $\mathcal{V}'$.

In the following, we assume that, when writing $\mathbf{U}_{\langle\langle p_1,\ldots,p_n\rangle\rangle}$, the order in which the names appear in the subscript matters. This convention is exploited in the following example.

Example 2. Let us consider the standard operator $\mathbf{cnot} : \ell^2(\{0,1\}^2) \to \ell^2(\{0,1\}^2)$, which acts on the computational basis as follows:

$$\begin{aligned} \mathbf{cnot}|00\rangle &= |00\rangle & \mathbf{cnot}|10\rangle &= |11\rangle \\ \mathbf{cnot}|01\rangle &= |01\rangle & \mathbf{cnot}|11\rangle &= |10\rangle. \end{aligned}$$

The **cnot** operator is one of the most important quantum operators (see also Example 4). Intuitively, the **cnot** operator complements the *target* bit (the second one) if the *control* bit (the first one) is 1, and otherwise does not perform any action. Let us fix the sequence $\langle p, q\rangle$ of variables, **cnot** induces the operator

$$\mathbf{cnot}_{\langle\langle p,q\rangle\rangle} : \mathcal{H}(\{p,q\}) \to \mathcal{H}(\{p,q\})$$

such that

$$\begin{aligned} \mathbf{cnot}_{\langle\langle p,q\rangle\rangle}|q \mapsto 0, p \mapsto 0\rangle &= |q \mapsto 0, p \mapsto 0\rangle; \\ \mathbf{cnot}_{\langle\langle p,q\rangle\rangle}|q \mapsto 0, p \mapsto 1\rangle &= |q \mapsto 1, p \mapsto 1\rangle; \\ \mathbf{cnot}_{\langle\langle p,q\rangle\rangle}|q \mapsto 1, p \mapsto 0\rangle &= |q \mapsto 1, p \mapsto 0\rangle; \\ \mathbf{cnot}_{\langle\langle p,q\rangle\rangle}|q \mapsto 1, p \mapsto 1\rangle &= |q \mapsto 0, p \mapsto 1\rangle. \end{aligned}$$

Please note that $|q \mapsto c_1, p \mapsto c_2\rangle = |p \mapsto c_2, q \mapsto c_1\rangle$, since the two expressions denote the same function. Consequently $\mathbf{cnot}_{\langle\langle p,q\rangle\rangle}|q \mapsto c_1, p \mapsto c_2\rangle = \mathbf{cnot}_{\langle\langle p,q\rangle\rangle}|p \mapsto c_2, q \mapsto c_1\rangle$. On the other hand, the operators $\mathbf{cnot}_{\langle\langle p,q\rangle\rangle}$ and $\mathbf{cnot}_{\langle\langle q,p\rangle\rangle}$ are different: both act as controlled not, but $\mathbf{cnot}_{\langle\langle p,q\rangle\rangle}$ uses p as control qubit while $\mathbf{cnot}_{\langle\langle q,p\rangle\rangle}$ uses q.

2.4. *A measurement in* $\mathcal{H}(\mathcal{V})$

In this section, we give a mathematical definition of a measurement operator in $\mathcal{H}(\mathcal{V})$.

We define two linear maps which perform a *destructive measurement* on a quantum register.

It will be used in Section 6, where we propose a quantum λ-calculus with explicit measurement operator (Dal Lago *et al.* 2011; Zorzi 2009).

Definition 4 (destructive measurements). Let $\mathcal{QV}$ be a set of names with cardinality $n = |\mathcal{QV}| \geqslant 1$, $r \in \mathcal{QV}$, $\{b_i\}_{i\in[1,2^{n-1}]}$ is the computational basis of $\mathcal{H}(\mathcal{QV} - \{r\})$ and let $\mathcal{Q}$ be $\sum_{i=1}^{2^{n-1}} \alpha_i|r \mapsto 0\rangle \otimes b_i + \sum_{i=1}^{2^{n-1}} \beta_i|r \mapsto 1\rangle \otimes b_i \in \mathcal{H}(\mathcal{QV})$. The two linear functions

$$\mathsf{m}_{r,0}, \mathsf{m}_{r,1} : \mathcal{H}(\mathcal{QV}) \to \mathcal{H}(\mathcal{QV} - \{r\})$$

such that

$$\mathsf{m}_{r,0}(\mathcal{Q}) = \sum_{i=1}^{2^{n-1}} \alpha_i b_i \qquad \mathsf{m}_{r,1}(\mathcal{Q}) = \sum_{i=1}^{2^{n-1}} \beta_i b_i$$

are called *destructive measurements*. If $\mathcal{Q}$ is a quantum register, the *probability* p_c *of observing* $c \in \{0,1\}$ *when observing* r *in* $\mathcal{Q}$ is defined as $\langle \mathcal{Q}|\mathsf{m}_{r,c}{}^{\dagger}\mathsf{m}_{r,c}|\mathcal{Q}\rangle$ (called *expected value*).

The name 'destructive' comes from the fact that the functions $\mathsf{m}_{r,c}$ ($c \in \{0,1\}$) map an element of a Hilbert space of dimension 2^n in an element of a Hilbert space 2^{n-1}. Let us define now the 'normalized' versions of $\mathsf{m}_{r,0}$ and $\mathsf{m}_{r,1}$:

Definition 5 (normalized destructive measurement). Given a set of names $\mathcal{QV}$ and $r \in \mathcal{QV}$, the linear maps

$$\mathcal{M}_{r,0}, \mathcal{M}_{r,1} : \mathcal{H}(\mathcal{QV}) \to \mathcal{H}(\mathcal{QV} - \{r\})$$

are defined as follows:

1. if $\langle \mathcal{Q}|\mathsf{m}^{\dagger}_{r,c}\mathsf{m}_{r,c}|\mathcal{Q}\rangle = 0$ then $\mathcal{M}_{r,c}(\mathcal{Q}) = \mathsf{m}_{r,c}(\mathcal{Q})$;
2. if $\langle \mathcal{Q}|\mathsf{m}^{\dagger}_{r,c}\mathsf{m}_{r,c}|\mathcal{Q}\rangle \neq 0$ then $\mathcal{M}_{r,c}(\mathcal{Q}) = \frac{\mathsf{m}_{r,c}(\mathcal{Q})}{\sqrt{\langle \mathcal{Q}|\mathsf{m}^{\dagger}_{r,c}\mathsf{m}_{r,c}|\mathcal{Q}\rangle}}$.

The just defined measurement operators belong to the class of the so-called *general measurements* (Kaye *et al.* 2007; Nielsen and Chuang 2000) and satisfy some properties, such as completeness. See Appendix B for statements and detailed proofs.

3. Quantum bits, quantum states and the framework of quantum mechanics

Quantum mechanics was born at the beginning of the 20th century, when it was clear that the classical theories (such as Newton's and Maxwell's theories) had great problems in order to explain and understand the unexpected results of several physical experiments. Quantum mechanics is the mathematical framework in which it is possible to develop new physical theories such as quantum physics, taking into account several surprising rules and postulates.

Paul Dirac wrote: '*Quantum Mechanics is more suitable in order to understand atomic phenomena, and from several points of view, it appear a more elegant theory with respect to the classical one*' (Dirac 1947). Nowadays, we can still say that we are able to understand some aspects of the world and the universe only by accepting the unusual point of view of quantum mechanics.

We will recall here the main ideas of quantum mechanics in a standard, intuitive way: we introduce the basic notions through some *postulates*, which capture the fundamental connections between the physical world and the mathematical formalism; the postulates give furthermore the basis of quantum computing.

Quantum mechanics' framework is able to interpret the structure, the evolution and the interaction of quantum systems. The first postulate of quantum mechanics assigns to quantum systems a mathematical representation in terms of Hilbert spaces.

Postulate I

The state of a system is described by a normalized vector in a Hilbert space $\mathcal{H}$

The Hilbert space $\mathcal{H}$ of a quantum system is called the *state space*, and the vector represents a *state vector*, which completely describes the system.

This very simple postulate embeds two features of the quantum state: the linearity of the system's description (see also Postulate II) and the existence of a scalar product and of a norm. In particular, the scalar product permits to link a probability to each observable.

As described in Section 2, each Hilbert space which describes a closed quantum system is defined on the field of complex numbers. Let us consider the Hilbert space $\ell^2(S)$ as defined in Section 2.3, and let us take $S = \{0,1\}^n$.

The Hilbert space $\ell^2(S)$ is the standard space used in quantum computing and it is useful to describe quantum states in a simple, intuitive (but rigorous) way.

The most simple quantum system is a two-dimensional state space whose elements are called *quantum bits* or *qubits* for short.

The more direct way to represent a quantum bit is a unitary vector in the two-dimensional Hilbert space $\ell^2(\{0,1\})$. We will denote with $|0\rangle$ and $|1\rangle$ the elements of the computational basis of $\ell^2(\{0,1\})$.

The states $|0\rangle$ and $|1\rangle$ of a qubit correspond to the boolean constants 0 and 1, which are the only possible values of a classical bit. A qubit, however, can assume other values, different from $|0\rangle$ and $|1\rangle$. In fact, every linear combination $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$, represents a possible qubit state. These states are said to be *superposed*, and the two values α and β are called *amplitudes*. The amplitudes α and β univocally represent the qubit with respect to the computational basis.

While we can determine the state of a classical bit, for a qubit we cannot establish with the same precision the values α and β: quantum mechanics says that a measurement of a qubit with state $\alpha|0\rangle + \beta|1\rangle$ has the effect of changing the state to $|0\rangle$ with probability $|\alpha|^2$ and to $|1\rangle$ with probability $|\beta|^2$. We will discuss this when we introduce the measurement postulate.

When defining quantum computational models, we need a generalization of the notion of qubit, the *quantum register* or *quantum state* (Masini *et al.* 2008, 2011; Nishimura and Ozawa 2009; Selinger 2004; Selinger and Valiron 2006; van Tonder 2004; Volpe *et al.* 2014). A quantum state can be viewed as a system of n qubits and, mathematically, it is a normalized vector in the Hilbert space $\ell^2(\{0,1\}^n)$ (Definition 1). The standard computational basis for $\ell^2(\{0,1\}^n)$ is
$\mathcal{B} = \{|i\rangle \mid i \text{ is a binary stringof length } n\}$.
In the literature, it is often written that $\ell^2(\{0,1\}^n)$ is the Hilbert space $\mathbb{C}^{2^n}$. This is not completely correct: we should say that $\ell^2(\{0,1\}^n)$ is isomorphic to $\mathbb{C}^{2^n}$. It is possible to prove that the map $v : \ell^2(\{0,1\}^n) \to \mathbb{C}^{2^n}$, such that for each element $|i\rangle \in \mathcal{B}$, $v(|i\rangle) = (0 \ldots 1 \ldots 0)^T$ (with 1 only in the $(i+1)$th position), is an isomorphism of Hilbert spaces. Note that v maps the computational basis of $\ell^2(\{0,1\}^n)$ into the standard basis of $\mathbb{C}^{2^n}$.

In the following, in order to not make the treatment heavy, we will work up to the above defined isomorphism v; namely, we will treat $\ell^2(\{0,1\}^n)$ (which has dimension 2^n) and $\mathbb{C}^{2^n}$ as the same space.

Note also that $\ell^2(\{0,1\}^n) \otimes \ell^2(\{0,1\}^m)$ is (up to isomorphism) $\ell^2(\{0,1\}^{n+m})$; the isomorphism is given by the map $|i\rangle \otimes |j\rangle \mapsto |ij\rangle$ (see also Postulate III for details about the composition of quantum system).

Example 3. Let us consider a two-level quantum system, i.e. a system of 2-qubits. Each 2-qubit quantum register is a normalized vector in $\ell^2(\{0,1\}^2)$ and the computational basis is $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$. For example, $\frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|00\rangle \in \ell^2(\{0,1\}^n)$ is a quantum register of 2-qubits.

The following postulate describes how an *isolated* physical system evolves over time. Since normalized vectors represent physical systems, the (discrete) evolution of systems can be viewed as a suitable transformation on Hilbert spaces. Postulate II ensures that the evolution is linear and unitary[†]:

Postulate II

The time evolution of the state of a *closed* quantum system is described by a unitary operator. Giving an initial state $|\psi_1\rangle$ for the closed system, for each evolution to a state $|\psi_2\rangle$, there exists a unitary operator U such that $|\psi_2\rangle = U|\psi_1\rangle$.

Thus, a quantum physical system can be described in term of linear operators and in a *deterministic* way[‡].

In quantum computing we refer to a unitary operator U acting on a n-qubit quantum register as an n-qubit *quantum gate*. Via the isomorphism v we can represent operators on the 2^n-dimensional Hilbert space $\ell^2(\{0,1\}^n)$ with respect to the standard basis of $\mathbb{C}^{2^n}$ as $2^n \times 2^n$ matrices, and it is possible to prove that to each unitary operator on a Hilbert space it is possible to associate an algebraic representation.

The application of quantum gates to quantum registers represents the pure quantum computational step and captures the internal evolution of quantum systems.

The most simple quantum gates act on a single qubit: they are operators on the space $\ell^2(\{0,1\})$, represented in $\mathbb{C}^2$ by 2×2 complex matrices.

For example, the quantum gate X is the unitary operator which maps $|0\rangle$ to $|1\rangle$ and $|1\rangle$ to $|0\rangle$ and it is represented by the matrix

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

[†] We refer to a closed system, i.e. to a system that interacts in no way with other systems or with the rest of the world, the observer included. This is an approximation of reality, but it is a very common approximation in physical theories. We accept this terminology in order to distinguish unitary evolution from quantum measurement, which implies an explicit interaction of the system with the environment.

[‡] It is possible to put in evidence the evolution with respect to time by adopting the notation '$|\psi(t)\rangle$', which describes the state at the time t. As a consequence, Postulate II can be written as $|\psi(t_2)\rangle = U(t_2, t_1)|\psi(t_1)\rangle$, where we also pick out the temporal information for the unitary operator. It is possible to prove that U is an unitary operator if the following condition holds: there exists a Hermitian operator A such that $U = \exp[iA]$. By means of simple algebraic steps it is easy to derive that $U(t_2, t_1) = \exp[iA'(t_2 - t_1)]$ where A' is again a Hermitian operator. It is useful to write A' in terms of an operator H (the *Hamiltonian operator*) which has energy dimension, i.e. $U(t_2, t_1) = \exp[iH(t_2 - t_1)/h]$ where h is Planck's constant. This condition gives an alternative formulation (in differential form) of the second postulate: $\frac{d}{dt}|\psi(t)\rangle = -\frac{i}{h}H|\psi(t)\rangle$. This is the famous Schrödinger equation, formulated in 1926 (Maccone and Salasnich 2008).

Being a linear operator, it maps a linear combination of inputs to the corresponding linear combination of outputs, and so X maps the general qubit state $\alpha|0\rangle + \beta|1\rangle$ into the state $\alpha|1\rangle + \beta|0\rangle$ i.e.

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \beta \\ \alpha \end{pmatrix}.$$

Other important 1-qubit quantum gates are

$$Y \equiv \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad Z \equiv \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}.$$

The quantum gates X, Y, Z are the so-called *Pauli gates.*

Another interesting unitary gate is the *Hadamard gate* denoted by H which acts on the computational basis in the following way:

$$|0\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) \qquad |1\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).$$

The Hadamard gate, which therefore is given by the matrix

$$H \equiv \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

is useful when we want to create a superposition starting from a classical state. It also holds that $H(H(|c\rangle)) = |c\rangle$ for $c = \{0, 1\}$.

1-qubit quantum gates can be used in order to build gates acting on n-qubit quantum states, since an n-qubit quantum register with $n \geqslant 2$ can be viewed as a composite system. It is possible to combine two (or more) distinct physical systems into a composite one. Postulate III tells us how tensor product of Hilbert space can describe the state space of a composite system.

Postulate III

When two physical systems are treated as one combined system, the state space of the two combined physical system is the tensor product space $\mathcal{H}_1 \otimes \mathcal{H}_2$ of the state spaces $\mathcal{H}_1$ and $\mathcal{H}_2$ of the component subsystems. If the first system is in the state $|\phi_1\rangle$ and the second system is in the state $|\phi_2\rangle$, then the state of the combined system is $|\phi_1\rangle \otimes |\phi_2\rangle$.

Notation 3. We will often omit the '$\otimes$' symbol, and will write the joint state as $|\psi_1\rangle|\psi_2\rangle$ or as $|\psi_1\psi_2\rangle$.

The physical principles behind Postulate III thrust the composition law of probability. In fact, the probability of obtaining two independent observables is exactly the product of the probabilities of the single results.

If we have a 2-qubit quantum system, we can apply a 1-qubit quantum gate only to one component of the system, and we implicitly apply the identity operator to the other one.

For example, suppose we want to apply X to the first qubit. The 2-qubits input $|\psi_1 \otimes \psi_2\rangle$ gets mapped to $X|\psi_1\rangle \otimes I|\psi_2\rangle = (X \otimes I)|\psi_1\rangle \otimes |\psi_2\rangle$.

3.1. *Entanglement*

Not all quantum states can be viewed as composite systems. In other words, *if* $|\psi\rangle$ *is a state of a tensor product space* $\mathcal{H}_1 \otimes \mathcal{H}_2$*, it is not generally the case that there exists* $|\psi_1\rangle \in \mathcal{H}_1$ *and* $|\psi_2\rangle \in \mathcal{H}_2$ *such that* $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$. Indeed, *it is not always possible to decompose an n-qubit state as the tensorial product of n qubits.*

These non-decomposable state are called *entangled* and enjoy properties that we cannot find in any object of classical physics. If n qubits are entangled, they behave as if connected, independently of the real physical distance. The strength of quantum computation is essentially based on the existence of entangled states (see, for example, the *teleportation protocol* (Nielsen and Chuang 2000)).

Example 4. The 2-qubit states $|\psi\rangle = \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle$, $|\psi\rangle = \frac{1}{\sqrt{2}}|01\rangle + \frac{1}{\sqrt{2}}|10\rangle$ are entangled. The 2-qubit state $\phi = \alpha|00\rangle + \beta|01\rangle$ is not entangled. In fact, it is possible to express it in the mathematically equivalent form $\phi = |0\rangle \otimes (\alpha|0\rangle + \beta|1\rangle)$.

3.2. *Measurement*

Describing unitary evolution of a quantum system, Postulate II assumes that the system is *closed*, i.e. that it is not allowed to interact with its environment. This is a good assumption in order to describe several properties, but a real system cannot always be closed.

In a realistic perspective, a quantum system interacts with another one, and also with a measurement apparatus. Since the evolution of the state during a measurement is not unitary, so we need a new postulate.

Postulate IV

Let A be a physical system, and let $B = \{|\phi_i\rangle\}$ be an orthonormal basis of a state space $\mathcal{H}_A$ for A. It is possible to perform a *measurement* on $\mathcal{H}_A$ w.r.t. B that given a state $|\psi\rangle = \sum_i \alpha_i|\phi_i\rangle$ leaves the system in the state ϕ_i with probability $|\alpha_i|^2$.

The described measurement is called *von Neumann measurement*, and it is a special kind of *projective* measurements (see e.g. Isham 1995; Kaye *et al.* 2007; Nielsen and Chuang 2000). Projective measurement (here proposed in a simplified setting) is intuitive and it is commonly used to explain the measurement postulate. It enjoys some properties. For example, repeated measurements give the same result: if after a measurement the system is (with the related probability) in the state ϕ_i, a second measurement applied on ϕ_i acts

as the identity. Moreover, projective measurement destroys the superposition[†] but does not modify the dimension of the Hilbert space.

Nevertheless, this is not the most general type of measurement in quantum mechanics. The measurement process can be generalized according to the different ways one wants to manipulate a given system. For example, one could want to apply two or more successive measurements or apply unitary transforms both before and after a measurement. A more comprehensive formalism for the description of measurement is the *general measurement* (Isham 1995; Kaye *et al.* 2007; Nielsen and Chuang 2000). An example of general measurement is the destructive measurement defined in Section 2.4. It is possible, however, under certain assumptions, to show equivalence between general measurements and projective measurements (see Nielsen and Chuang (2000) for a detailed discussion about this topic).

3.3. *No-cloning theorem*

The no-cloning theorem states that quantum mechanics does not allow us to make a copy of an unknown quantum state. It was discovered in the early 1980's (Wooters and Zurek 1982) and it captures one of the fundamental properties of quantum systems and of quantum information.

One of the primitive operations in information theory is the copy of a datum but when we deal with quantum data as qubits, quantum information suffers from lack of accessibility in comparison to classical one.

Why is it not possible to duplicate a quantum bit? Let $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ be a 1-qubit quantum state. We could try to make a copy using a CNOT gate, as for the classical case[‡]. Let then CNOT gate take the state $|\psi\rangle$ as the control input, and a state initialized to $|0\rangle$ as the target input. The input state is therefore $\alpha|00\rangle + \beta|10\rangle$. As output, could the CNOT gate give the tensor state $|\psi\rangle \otimes |\psi\rangle$?

The function of CNOT is to complement the second qubit only if the first is 1, and thus the output state will be $\alpha|00| + \beta|11\rangle$. This is equal to the state $|\psi\rangle \otimes |\psi\rangle = \alpha^2|00| + \alpha\beta|01| + \alpha\beta|10\rangle + \beta^2|11\rangle$ if and only if $\alpha\beta = 0$.

In general, we can prove the following:

Theorem 1 (no-cloning theorem). There does not exist a unitary transformation U such that, given a quantum state $|\phi\rangle$ and a quantum state[§] $|s\rangle$

$$U(|\phi\rangle \otimes |s\rangle) = |\phi\rangle \otimes |\phi\rangle.$$

[†] In Postulate IV, the state $|\psi\rangle$ collapses to an element of the orthonormal basis, totally loosing the quantum superposition. It is possible, however, to define projective measurements also with respect to a single qubit, i.e. it is possible to observe the ith qubit of the basis vectors. In this case, only a part of the superposition is destroyed, possibly leaving the system in quantum superposition.

[‡] If we take as control input a bit i and as target input a bit 0, the CNOT result is obviously $i \otimes i$.

[§] The state $|s\rangle$ is assumed to be *pure*, i.e. a quantum state which is not a probabilistic distribution of other quantum states. In quantum mechanics, the notion of pure state is opposed to the notion of *mixed* state, see Kitaev *et al.* (2002) and Kaye *et al.* (2007) for detailed discussions.

Proof. We propose an elementary proof, frequently proposed in the literature (see, for example Nielsen and Chuang (2000)). Suppose, there exists the cloning operator U and suppose this copying procedure works for two particular state $|\psi_1\rangle$ and $|\psi_2\rangle$.

We have

$$U(|\psi_1\rangle \otimes |s\rangle) = |\psi_1\rangle \otimes |\psi_1\rangle$$
$$U(|\psi_2\rangle \otimes |s\rangle) = |\psi_2\rangle \otimes |\psi_2\rangle.$$

If we take the inner product of the two equations (remember that U is unitary and preserves the inner product) we obtain

$$\langle\psi_1|\psi_2\rangle = (\langle\psi_1|\psi_2\rangle)^2$$

which has only the solutions 0 and 1. So, either $|\psi_1\rangle = |\psi_2\rangle$ or the two states are orthogonal. Thus, a cloning device can only clone the states of the computational basis (or classical states), but it is not possible to make a copy of a general quantum state.

□

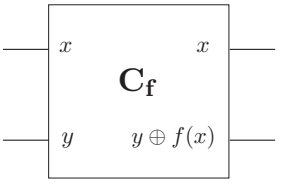
4. Quantum computing: more than probabilistic computing!

The aim of the present section is to offer some intuitions about the potentiality of quantum computing. The 'thesis' of quantum computer scientists is that quantum algorithms (or quantum models), could outperform their classical and probabilistic counterparts. This section wishes to provide some convincing arguments in this direction.

The 'superpower' of quantum computing essentially comes from exploiting two peculiar phenomena. The first one is quantum parallelism (Nielsen and Chuang 2000).

Oversimplifying, quantum parallelism allows a quantum computer to evaluate a function $f(x)$ for different values x at the same time and it is able to extract information about more than one of the values $f(x)$ from a superposition state.

Given a function $f : \{0,1\} \to \{0,1\}$, and a 2-qubit input $|x, y\rangle$, the following circuit



is able to compute the value $|x, y \oplus f(x)\rangle$, where the operator $\oplus$ represents the addition modulo 2. The transformation $|x, y\rangle \to |x, y \oplus f(x)\rangle$ is easily shown to be unitary. In particular, taking $x = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $y = 0$ as inputs, the output state will be $\frac{|0,f(0)\rangle+|1,f(1)\rangle}{\sqrt{2}}$, which provides information about $f(0)$ and $f(1)$ simultaneously.

A noticeable application of this situation can be found in Example 8, Section 5.13.1, where we will propose the encoding of the popular Deutsch's algorithm.

The second phenomenon (strongly related to quantum parallelism) is called *quantum negative interference*. To explain it, we discuss the quantum logical gate $\sqrt{\mathsf{NOT}}$ (Deutsch *et al.* 2000) (also called quantum coin flip QCF (Brassard 1994)) which has no classical

counterpart. The $\sqrt{\mathsf{NOT}}$ gate is described by the following unitary matrix

$$\sqrt{\mathsf{NOT}} \equiv \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$$

which takes a vector in the plane and rotates it by 45 degrees counterclockwise.

For example, on input $|0\rangle$, one has that $\sqrt{\mathsf{NOT}}(|0\rangle) = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. If one measures the state, following Postulate IV (Section 3), this means that one can observes $|0\rangle$ and $|1\rangle$ with the same probability $|\frac{1}{\sqrt{2}}|^2 = \frac{1}{2}$. Actually, one single application of the matrix behaves as a random negation, yielding a fair superposition of basis vectors $|0\rangle$ and $|1\rangle$.

One could try to perform the same transformation taking into account a random gate, i.e a bistochastic matrix (a matrix whose columns and rows entries sum up to 1). This can be obtained by replacing in the $\sqrt{\mathsf{NOT}}$ matrix the amplitudes $+\frac{1}{\sqrt{2}}$ and $-\frac{1}{\sqrt{2}}$ with *positive* probabilities $\frac{1}{2}$:

$$\mathsf{CP} \equiv \frac{1}{2} \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}.$$

Such a gate, dubbed here CP, simply performs a coin flip (on input $|c\rangle$, $c \in \{0, 1\}$, it returns $|0\rangle$ with probability $\frac{1}{2}$ and $|1\rangle$ with probability $\frac{1}{2}$). Observe that CP seems to behave exactly like $\sqrt{\mathsf{NOT}}$ (once the superposition generated by $\sqrt{\mathsf{NOT}}$ has been observed, the outputs of the two gates appear indistinguishable). The interesting quality of the quantum gate appears when we link two gates in series.

On one hand, one can plainly verify that, given $|c\rangle$ ($c \in \{0, 1\}$), $\sqrt{\mathsf{NOT}}(\sqrt{\mathsf{NOT}}(|c\rangle)) = |\overline{c}\rangle$ (where $\overline{c}$ is the complement of c): the machine built out from the concatenation of two copies of $\sqrt{\mathsf{NOT}}$ globally acts like a classical not gate.

On the other hand, a concatenation of two copies of CP works differently: the second gate randomizes the vector again, yielding a fair distribution of $|0\rangle$ and $|1\rangle$. In general, it is easy to verify that any number n of successive applications of CP is equivalent to the application of a single CP gate.

The example shows a peculiar fact: after the second application of the $\sqrt{\mathsf{NOT}}$ gate, the 'randomization' disappears: in some sense, applying a 'randomizing' operation to a quantum state (which, observed, yields a random value) one produces a deterministic outcome. What has happened? The answer comes from quantum negative interference. Let us consider again the computation $\sqrt{\mathsf{NOT}}\sqrt{\mathsf{NOT}}(|0\rangle)$. The first application of the quantum gate gives the state $\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$; applying $\sqrt{\mathsf{NOT}}$ one more time, the first summand of the superposition ($|0\rangle$) turn into $\frac{1}{\sqrt{2}}(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle))$ and the second summand of the superposition ($|1\rangle$) come to be $\frac{1}{\sqrt{2}}(\frac{1}{\sqrt{2}}(-|0\rangle + |1\rangle))$. Indeed, the global state is $\frac{1}{\sqrt{2}}(\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)) + \frac{1}{\sqrt{2}}(\frac{1}{\sqrt{2}}(-|0\rangle + |1\rangle))$. By trivial algebraic manipulations one has: $\frac{1}{2}(|0\rangle) + \frac{1}{2}(|1\rangle) - \frac{1}{2}(|0\rangle) + \frac{1}{2}(|1\rangle) = \frac{1}{2}(|1\rangle) + \frac{1}{2}(|1\rangle) = |1\rangle$.

In the classical world it is not possible to observe this sort of interference, since *in classical probability theory probabilities can only be positive* and no negative interference phenomena can occur.

At a first glance, the potential power of quantum computing collapses with the peculiarities of the randomized computation. But during a quantum computation, phenomena

such as parallelism and negative interference can be exploited, making the model radically different.

5. Q: a measurement-free quantum λ-calculus

In the main part of this article we will illustrate Q, a calculus for quantum computable functions. Q has be introduced in Dal Lago *et al.* (2009), further developed in Dal Lago *et al.* (2010) and successively extended in Dal Lago *et al.* (2011) (see also (Zorzi 2009)).

A calculus for quantum computable functions should present two very different computational 'aims'. On the one hand there is the unitary aspect of the calculus, that captures the essence of quantum computing as algebraic transformations of state vectors by means of unitary operators. On the other hand, it is possible to have the *control* of the pure quantum computation: this, of course, relaxes the unitarity of the paradigm (this means that it is not possible to describe each computational step as a unitary transformation on a Hilbert space), embedding the pure quantum evolution in a classical computation[†]. Behind this second aim, we have the usual idea of computation as a sequence of discrete steps on (the mathematical description of) an abstract machine.

The relationship between these different aspects give rise to different approaches to quantum functional calculi (as observed in Arrighi and Dowek (2008)).

If we divide the two features, i.e. we separate data from control, we adopt the so-called *quantum data – classical control approach.*

This means that classical computation is *independent* from the quantum part: a classical program (ideally in execution on a classical machine) computes some 'directives': these directives are sent to a hypothetical device which applies them to quantum data. Therefore, quantum data are manipulated by the classical program: classical computational steps control the unitary part of the calculus. In general, the classical control acts on the quantum side of the computation in two ways: by means of unitary transformation and by means of data observation, i.e. by means of measurement (usually, in quantum calculi, a syntactic primitive which acts as a measurement apparatus and returns a classical bit, see Section 6). Another possible directive the control can perform is the creation of a new quantum datum (see the function new in Section 5.8).

The calculus Q we propose strictly follows the quantum data – classical control paradigm, going behind Selinger's proposal for quantum functional languages (Selinger 2004; Selinger and Valiron 2006). In Figure 1, this concept is depicted as a client-server architecture: a lambda evaluator client normalizes lambda terms and when a quantum operation occurs, this step is 'implemented' by the quantum server. Notice that in Figure 1, there does not exist any feedback from the quantum server to the classical evaluator. In

[†] This choice is theoretically sound in a realistic perspective: a classical control should be the 'easy' part of quantum computing in a hybrid architecture built out of a classical machine and a quantum device. As suggested in Selinger (2004) it is useful to think of a particular hardware on which a quantum language can be implemented. A suitable model is the QRAM machine (Knill 1996), a general-purpose computer which controls a quantum hardware device. Our Figure 1 is reminiscent of such a model.

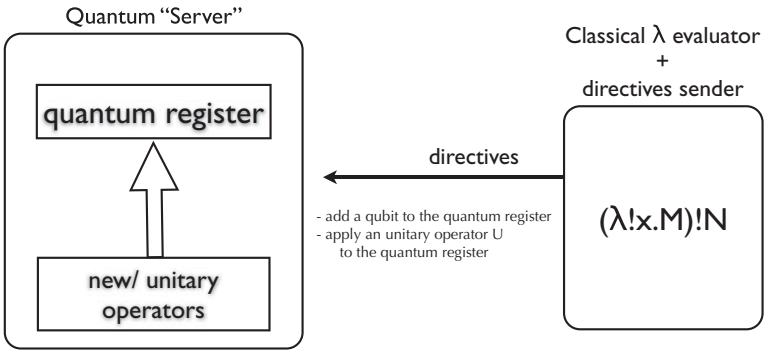


Fig. 1. Quantum client-server architecture without feedback.

fact, we will start from a measurement-free calculus (as explained and motivated in the following section).

We will start with a careful operational analysis of Q. In Section 5.14.1, we will prove that Q is 'quantum Turing complete' in the following sense: Q will be showed to be equivalent to the class of finitely generated uniform quantum circuit families and (by well-known results) to quantum Turing machines (see Appendix A.5). This is an essential result, if we want to affirm that a calculus is a suitable instrument for the characterization of quantum computable functions. For example, a promising calculus such as van Tonder's λ_q (which was the first proposal in this direction, as explained in Section 7) suffers from the lack of formally proved results about its expressive power.

5.1. *The principal features of* Q

Developing Q, we made a number of choices with respect to several aspects of the calculus. In the following paragraphs we will informally describe the principal features of Q, before giving technical details.

5.2. *Configurations*

Computational steps in Q are defined between *configurations.*

A *configuration* is a triple $[\mathcal{Q}, \mathcal{QV}, M]$ that gives a full instantaneous description of the state of a computation in Q: M is a term from a suitable grammar, $\mathcal{Q}$ is a quantum state, $\mathcal{QV}$ is a set of quantum variables (a superset of those appearing in M). Computational rules are of the form $[\mathcal{Q}, \mathcal{QV}, M] \rightarrow [\mathcal{Q}', \mathcal{QV}', M']$; a configuration can evolve classically (only the lambda term M is involved in the reduction and $\mathcal{Q} = \mathcal{Q}'$, $\mathcal{QV} = \mathcal{QV}'$) or nonclassically (the reduction involves also the quantum register and the set of quantum variables).

The notion of configuration is a useful technical instrument in order to model quantum computation forgetting any representation problem.

5.3. *Untyped setting*

We will work in an untyped setting in order to capture the full expressiveness of the calculus. In this way, we are able to prove the 'quantum Turing completeness' of Q, showing the equivalence with the class of finitely generated uniform quantum circuit families (see Appendix A.2, Definition 32).

Nevertheless, a term's construction is not free. In fact, in order to respect quantum features of data, we need to distinguish between linearity and duplicability. A term's generation is controlled by means of well-forming rules.

5.4. *Linearity of the calculus (static): 'bang' operator*

Q is a calculus designed for quantum computable functions. However, it deals with linearity. Linearity corresponds to the constraint that in every term of the shape $\lambda x.M$ there is *exactly one* free occurrence of the variable x in M. Thus, the quantum variables occurring in a redex $\lambda x.M(N)$ are exactly the ones occurring in the reduct $M\{N/x\}$.

A linear term is neither duplicable nor erasable: this way we are able to guarantee that the 'no-cloning and no-erasing' property is satisfied. The Q syntax includes the modal operator '!' (bang), which allows us to distinguish between different syntactical objects which represent, respectively, classical (duplicable) and quantum (non-duplicable) resources. In Q, a term in the form $!N$ cannot contain (references to) quantum data.

5.5. *Linearity of the calculus (dynamic): surface reduction*

The set of well-forming rules captures the separation between the linear and the classical part of the syntax. Moreover, we capture linearity 'dynamically', adopting *surface reduction* (Simpson 2005): in Q it is not possible to reduce under the scope of the bang operator '!'. Let us consider the term $\lambda !xM(!N)$: a correct computation performs the β-reduction obtaining $M\{N/x\}$; a wrong computation reduces the subterm N.

This is not the only way to enforce the no-cloning and no-erasing properties. Other solutions have been proposed in the literature, see e.g. Altenkirch *et al.* (2007), where duplication is modelled by means of sharing.

5.6. *Absence of reduction strategies*

Surface reduction (Simpson 2005) is the only restriction on reductions. We do not define any reduction strategy for Q. The calculus is confluent in a strong sense: normal forms are unique and also strong normalization and weak normalization are equivalent properties of configurations. It is mandatory to stress that full confluence results are strongly related to the absence of intermediate measurement steps. We will see in Section 6, that the presence of an explicit measurement operator imports a probabilistic behaviour into the computation, thus it is not possible to retrace confluence theorems in a standard form, and it is necessary to reformulate the problem in a different way.

5.7. *Implicit measurement at the end of the computations*

In Q it is not possible to classically observe the content of the quantum register. More specifically, the language of terms does not include any measurement operator which, applied to a quantum variable, has the effect of observing the value of the related qubit. Therefore, we start from a measurement-free quantum λ-calculus. This means that *we assume to have a unique measurement at the end of the computation*. This is a theoretical well-founded choice (Aharonov *et al.* 1998; Nielsen and Chuang 2000; Selinger 2004). Of course, measurement is a fundamental (nonunitary) computational step in some important quantum algorithms. It also opens some interesting theoretical questions. For these reasons, in Section 6, we extend the syntax of Q with an explicit measurement constructor.

5.8. *Syntax and well-forming rules*

The language of terms of Q, given in the following grammar, is an extension of the syntax of an untyped λ-calculus with constants.

$$
\begin{array}{lll}
x & ::= x_0, x_1, \ldots & \textit{classical variables} \\
r & ::= r_0, r_1, \ldots & \textit{quantum variables} \\
\pi & ::= x \mid \langle x_1, \ldots, x_n \rangle & \textit{linear patterns} \\
\psi & ::= \pi \mid !x & \textit{patterns} \\
B & ::= 0 \mid 1 & \textit{boolean constants} \\
U & ::= U_0, U_1, \ldots & \textit{unitary operators} \\
C & ::= B \mid U & \textit{constants} \\
M & ::= x \mid r \mid !M \mid C \mid \mathtt{new}(M) \mid M_1 M_2 \mid & \\
 & \quad \langle M_1, \ldots, M_n \rangle \mid \lambda\psi.M & \textit{terms (where } n \geqslant 2\textit{)}
\end{array}
$$

Principal points are the following:

— Variables are partitioned into *classical* and *quantum* variables: the first ones are the usual variables of λ-calculus and they can be bound by a λ-abstraction; quantum variables refer to qubits in the underlying quantum register. Roughly speaking, quantum variables represents names of quantum bits.

— There are two sorts of constants, namely *boolean constants* (0 and 1) and *unitary operators*: the first ones are useful for generating qubits and play no role in classical computations; unitary operators are applied to tuples of quantum variables when performing quantum computation. Then, in our system we take 'built-in' unitary operators, with the watchfulness explained in the following. We associate to each computable operator $\mathbf{U} \in \mathcal{U}$ a symbol U, where $\mathcal{U}$ is a denumerable fixed set of computable unitary operators (see Appendix A.1, Definition 30).

— The term constructor $\mathtt{new}(\cdot)$ creates a new qubit when applied to a boolean constant. For example, $\mathtt{new}(0)$ returns $|0\rangle$.

— The syntax of terms includes a modal operator ! (the 'bang' operator) introduced in term calculi for linear logic (see for example Wadler's syntax (Wadler 1994)). Bang

modality allows us to distinguish between those syntactical objects (λ-terms) that can be duplicated or erased and those that cannot. In this way, we are able to erase and duplicate classical terms, which do not contain references to quantum data. Roughly speaking, a term is duplicable and erasable if and only if it is of the form $!M$ and, moreover, M does not contain quantum variables. This constraint is ensured 'statically' by the well-forming rules in Figure 2.

— The syntax allows pattern abstraction. A pattern is either a classical variable, a tuple of classical variables, or a 'banged' variable, namely an expression of the kind $!x$, where x is a name of a classical variable. In order to allow an abstraction of the kind $\lambda!x.M$, the environment (see below) must be enriched with !-patterns, denoting duplicable or erasable variables. Then, quantum variables cannot be lambda-abstracted: they can be passed as an argument to a linear pattern by a particular kind of β-reduction (see Section 5.10, Figure 3, rule $\mathsf{q}.\beta$).

In the following, we assume to be working modulo variable renaming, i.e. *terms are equivalence classes modulo α-conversion* (Barendregt 1984). Substitution up to α-equivalence is defined in the usual way.

Notation 4. Let us denote with $\mathbf{Q}(M_1,\ldots,M_k)$ the set of quantum variables occurring in $M_1,\ldots,M_k$. In the rest of the paper, a finite subset of quantum variables will be called a *quantum variable set* (*qvs*).

Note 5.1. The expressive power of Q depends on the choice of the set $\mathcal{U}$, the constants which represent unitary operators. If one want, for example, to capture quantum Turing machines in the style of Bernstein and Vazirani (1997), one fix $\mathcal{U}$ to be the set of computable operators (see Appendix A, Definition 30). On the other hand, the expressivity results given in Section 5.14.1 relates Q and quantum circuit families; clearly, those that can be captured by Q terms with computable operators in $\mathcal{U}$ are precisely those (finitely) generated by $\mathcal{U}$ (see Appendix A.2).

5.9. *Well-forming rules*

Since we are working in a untyped setting, term generation is controlled by well-forming rules. Judgements are defined from various notions of environments, that take into account the way the variables are used. In the following, a set of variables $\{x_1,\ldots,x_n\}$ is often written simply as $x_1,\ldots,x_n$. Analogously, the union of two sets of variables X and Y is denoted simply as X,Y.

— A *classical environment* is a (possibly empty) set of classical variables. Classical environments are denoted by Δ (possibly with indexes). Examples of classical environments are x_1,x_2 or x,y,z or the empty set $\varnothing$. Given a classic environment $\Delta = x_1,\ldots,x_n$, $!\Delta$ denotes the set of patterns $!x_1,\ldots,!x_n$.

— A *quantum environment* is a (possibly empty) set of quantum variables. Quantum environments are denoted Θ (possibly indexed). Examples of quantum environments are r_1,r_2,r_3 or the empty set $\varnothing$.

$$\frac{}{!\Delta \vdash C}\ \mathsf{const} \qquad \frac{}{!\Delta, r \vdash r}\ \mathsf{q\text{–}var} \qquad \frac{}{!\Delta, x \vdash x}\ \mathsf{classic\text{-}var} \qquad \frac{}{!\Delta, !x \vdash x}\ \mathsf{der}$$

$$\frac{!\Delta \vdash M}{!\Delta \vdash !M}\ \mathsf{prom} \qquad \frac{\Lambda_1, !\Delta \vdash M \quad \Lambda_2, !\Delta \vdash N}{\Lambda_1, \Lambda_2, !\Delta \vdash MN}\ \mathsf{app} \qquad \frac{\Lambda_1, !\Delta \vdash M_1 \cdots \Lambda_k, !\Delta \vdash M_k}{\Lambda_1, \ldots, \Lambda_k, !\Delta \vdash \langle M_1, \ldots, M_k \rangle}\ \mathsf{tens}$$

$$\frac{\Gamma \vdash M}{\Gamma \vdash \mathtt{new}(M)}\ \mathsf{new} \qquad \frac{\Gamma, x_1, \ldots, x_n \vdash M}{\Gamma \vdash \lambda\langle x_1, \ldots, x_n \rangle.M}\ \mathsf{lam1} \qquad \frac{\Gamma, x \vdash M}{\Gamma \vdash \lambda x.M}\ \mathsf{lam2} \qquad \frac{\Gamma, !x \vdash M}{\Gamma \vdash \lambda !x.M}\ \mathsf{lam3}$$

Fig. 2. Well-forming rules.

— A *linear environment* is a (possibly empty) set, denoted by Λ (possibly indexed), in the form Δ, Θ where Δ is a classical environment and Θ is a quantum environment. The set x_1, x_2, r_1 is an example of a linear environment.
— An *environment*, denoted by Γ (possibly indexed), is a (possibly empty) set in the form $\Lambda, !\Delta$ where each classical variable x occurs at most once (either as $!x$ or as x) in Γ. For example, $x_1, r_1, !x_2$ is an environment, while $x_1, !x_1$ is *not* an environment.
— A *judgement* is an expression $\Gamma \vdash M$, where Γ is an environment and M is a term of Q.

We say that a judgement $\Gamma \vdash M$ is *well-formed* (notation: $\triangleright \Gamma \vdash M$) if it is derivable by means of the *well-forming rules* in Figure 2. The rules app and tens are subject to the constraint that for each $i \neq j$, we have $\Lambda_i \cap \Lambda_j = \varnothing$ (note that Λ_i and Λ_j are linear environments).

Moreover, with $d \triangleright \Gamma \vdash M$ we mean that d is a derivation of the well-formed judgement $\Gamma \vdash M$. If $\Gamma \vdash M$ is well formed, we say also that the term M *is well formed with respect to the environment* Γ. We say that a term M is *well formed* if the judgement $\mathbf{Q}(M) \vdash M$ is well formed. It is possible to prove that if a term M is well formed then all its classical variables are bound.

Notice that the structure of our terms is strongly based on the formulation of linear logic proposed by Wadler (1994). Moreover, it is easy to observe that, by means of well-forming rules,

a duplicable term $!N$ *cannot contain quantum variables.*

This way, we statically guarantee that references to quantum bits can be neither duplicable nor erasable. Dynamically, quantum properties will be assured by means of surface reduction (Section 5.10).

5.10. *Computations*

Reductions in Q are defined between configurations. A configuration is a reformulation of the notion of *program state*, introduced by Selinger (2004).

We formally define configurations as equivalence classes of *preconfigurations*, i.e. triples of the kind $[\mathcal{Q}, \mathcal{QV}, M]$ where:

— M is a term;
— $\mathcal{QV}$ is a finite qvs such that $\mathbf{Q}(M) \subseteq \mathcal{QV}$;
— $\mathcal{Q} \in \mathcal{H}(\mathcal{QV})$.

Let $\theta : \mathcal{QV} \to \mathcal{QV}'$ be a bijective function from a (nonempty) finite set of quantum variables $\mathcal{QV}$ to another set of quantum variables $\mathcal{QV}'$. Then we can extend θ to any term whose quantum variables are included in $\mathcal{QV}$: $\theta(M)$ will be identical to M, except on quantum variables, which are changed according to θ itself. Observe that $\mathbf{Q}(\theta(M)) \subseteq \mathcal{QV}'$. Similarly, θ can be extended to a function from $\mathcal{H}(\mathcal{QV})$ to $\mathcal{H}(\mathcal{QV}')$ in the obvious way.

Definition 6 (configurations). Two preconfigurations $[\mathcal{Q}, \mathcal{QV}, M]$ and $[\mathcal{Q}', \mathcal{QV}', M']$ are equivalent iff there is a bijection $\theta : \mathcal{QV} \to \mathcal{QV}'$ such that $\mathcal{Q}' = \theta(\mathcal{Q})$ and $M' = \theta(M)$. If a preconfiguration C is equivalent to D, then we will write $C \equiv D$. The relation $\equiv$ is an equivalence relation. A *configuration* is an equivalence class of preconfigurations modulo the relation $\equiv$. Let **Conf** be the set of configurations.

The way configurations are defined, namely quotienting preconfigurations over a relation $\equiv$, is reminiscent of usual α-conversion in lambda terms.

5.11. *Reduction rules*

A computation in Q is a (possibly infinite) sequence of the form $C_0 = [\mathcal{QV}_0, \mathcal{QV}_0, M_0] \to_\alpha [\mathcal{QV}_1, \mathcal{QV}_1, M_1] \to_\alpha \ldots \to_\alpha [\mathcal{QV}_k, \mathcal{QV}_k, M_k] \to_\alpha \ldots$ (where C_0 is the initial configuration and α ranges over a finite set of names of reduction rules).

A computational step, indeed, possibly involves each component of the configuration, performing a reduction on the lambda term and eventually acting on quantum data.

Computations are formally introduced in Definition 10.

Reduction rules are in Figure 3. Let $\mathscr{L} = \{\mathsf{Uq}, \mathsf{new}, \mathsf{l}.\beta, \mathsf{q}.\beta, \mathsf{c}.\beta, \mathsf{l.cm}, \mathsf{r.cm}\}$. The set $\mathscr{L}$ will be ranged over by α, β, γ. For each $\alpha \in \mathscr{L}$, we can define a reduction relation $\to_\alpha \subseteq \mathcal{C} \times \mathcal{C}$ by means of the rules in Figure 3.

For any subset $\mathscr{S}$ of $\mathscr{L}$, we can build a relation $\to_{\mathscr{S}}$ by just taking the union over $\alpha \in \mathscr{S}$ of $\to_\alpha$. In particular, $\to$ will denote $\to_{\mathscr{L}}$. The usual notation for the transitive and reflexive closures will be used. In particular, $\to^*$ will denote the transitive and reflexive closure of $\to$.

In the following we will also deal with some particular subsets of $\mathscr{L}$:

Definition 7 (subsets of $\mathscr{L}$).
— $\mathscr{Q} = \{\mathsf{Uq}, \mathsf{q}.\beta\}$ is the set of quantum rules;
— $n\mathscr{C} = \mathscr{Q} \cup \{\mathsf{new}\} = \{\mathsf{Uq}, \mathsf{q}.\beta, \mathsf{new}\}$ is the set of non-classical rules (including quantum reductions and new reductions);
— $\mathscr{C} = \mathscr{L} - n\mathscr{C} = \{\mathsf{l}.\beta, \mathsf{c}.\beta, \mathsf{l.cm}, \mathsf{r.cm}\}$ is the set of classical rules;
— $\mathscr{K} = \{\mathsf{l.cm}, \mathsf{r.cm}\}$ and $\mathscr{N} = \mathscr{L} - \mathscr{K} = \{\mathsf{Uq}, \mathsf{new}, \mathsf{l}.\beta, \mathsf{q}.\beta, \mathsf{c}.\beta\}$ denote the sets of commutative and non-commutative rules, respectively.

Let us explain the meaning of computational rules. By means of reductions, configurations can evolve classically and nonclassically.

β–reductions

$$[\mathcal{Q}, \mathcal{QV}, (\lambda x.M)N] \to_{\mathsf{l}.\beta} [\mathcal{Q}, \mathcal{QV}, M\{N/x\}] \quad \mathsf{l}.\beta$$

$$[\mathcal{Q}, \mathcal{QV}, (\lambda\langle x_1, \ldots, x_n\rangle.M)\langle r_1, \ldots, r_n\rangle] \to_{\mathsf{q}.\beta} [\mathcal{Q}, \mathcal{QV}, M\{r_1/x_1, \ldots, r_n/x_n\}] \quad \mathsf{q}.\beta$$

$$[\mathcal{Q}, \mathcal{QV}, (\lambda !x.M)!N] \to_{\mathsf{c}.\beta} [\mathcal{Q}, \mathcal{QV}, M\{N/x\}] \quad \mathsf{c}.\beta$$

Unitary transformation of a quantum register

$$[\mathcal{Q}, \mathcal{QV}, U\langle r_{i_1}, ..., r_{i_n}\rangle] \to_{\mathsf{Uq}} [\mathbf{U}_{\langle\langle r_{i_1}, ..., r_{i_n}\rangle\rangle}\mathcal{Q}, \mathcal{QV}, \langle r_{i_1}, ..., r_{i_n}\rangle] \quad \mathsf{Uq}$$

Creation of a new qubit and quantum variable

$$[\mathcal{Q}, \mathcal{QV}, \mathtt{new}(c)] \to_{\mathsf{new}} [\mathcal{Q} \otimes |r \mapsto c\rangle, \mathcal{QV} \cup \{r\}, r] \quad \mathsf{new} \quad (r \text{ is fresh}, c \in \{0, 1\})$$

Commutative reductions

$$[\mathcal{Q}, \mathcal{QV}, L((\lambda\pi.M)N)] \to_{\mathsf{l.cm}} [\mathcal{Q}, \mathcal{QV}, (\lambda\pi.LM)N] \quad \mathsf{l.cm}$$

$$[\mathcal{Q}, \mathcal{QV}, ((\lambda\pi.M)N)L] \to_{\mathsf{r.cm}} [\mathcal{Q}, \mathcal{QV}, (\lambda\pi.ML)N] \quad \mathsf{r.cm}$$

where L is free for π

Context closure

$$\frac{[\mathcal{Q}, \mathcal{QV}, M_i] \to_\alpha [\mathcal{Q}', \mathcal{QV}', M_i']}{[\mathcal{Q}, \mathcal{QV}, \langle M_1, \ldots, M_i, \ldots, M_k\rangle] \to_\alpha [\mathcal{Q}', \mathcal{QV}', \langle M_1, \ldots, M_i', \ldots, M_k\rangle]}\ \mathsf{t}_i$$

$$\frac{[\mathcal{Q}, \mathcal{QV}, N] \to_\alpha [\mathcal{Q}', \mathcal{QV}', N']}{[\mathcal{Q}, \mathcal{QV}, MN] \to_\alpha [\mathcal{Q}', \mathcal{QV}', MN']}\ \mathsf{r.a} \qquad \frac{[\mathcal{Q}, \mathcal{QV}, M] \to_\alpha [\mathcal{Q}', \mathcal{QV}', M']}{[\mathcal{Q}, \mathcal{QV}, MN] \to_\alpha [\mathcal{Q}', \mathcal{QV}', M'N]}\ \mathsf{l.a}$$

$$\frac{[\mathcal{Q}, \mathcal{QV}, M] \to_\alpha [\mathcal{Q}', \mathcal{QV}', M']}{[\mathcal{Q}, \mathcal{QV}, \mathtt{new}(M)] \to_\alpha [\mathcal{Q}', \mathcal{QV}', \mathtt{new}(M')]}\ \mathsf{in.new}$$

$$\frac{[\mathcal{Q}, \mathcal{QV}, M] \to_\alpha [\mathcal{Q}', \mathcal{QV}', M']}{[\mathcal{Q}, \mathcal{QV}, (\lambda !x.M)] \to_\alpha [\mathcal{Q}', \mathcal{QV}', (\lambda !x.M')]}\ \mathsf{in}.\lambda_1 \qquad \frac{[\mathcal{Q}, \mathcal{QV}, M] \to_\alpha [\mathcal{Q}', \mathcal{QV}', M']}{[\mathcal{Q}, \mathcal{QV}, (\lambda\pi.M)] \to_\alpha [\mathcal{Q}', \mathcal{QV}', (\lambda\pi.M')]}\ \mathsf{in}.\lambda_2$$

Fig. 3. Reduction rules.

Let us consider the reduction schema $[\mathcal{Q}, \mathcal{QV}, M] \to_\alpha [\mathcal{Q}', \mathcal{QV}', M']$.

— $\alpha \in \mathscr{C}$: we have a *classical evolution*. A reduction step from M is performed, obtaining M' as reduct; $\mathcal{Q}$ and $\mathcal{QV}$ will not be modified, thus $\mathcal{Q} = \mathcal{Q}'$ and $\mathcal{QV} = \mathcal{QV}'$. No quantum variable has been created, and no unitary transformation has been applied to the quantum register. The only significant component of the computational step is the λ–term M.

— $\alpha \in n\mathscr{C}$: we have a *non-classical evolution*. This kind of reduction step induces an interaction between the term M and the quantum register. This class of reductions (potentially) modifies the set $\mathcal{QV}$ and the quantum register $\mathcal{Q}$ in three possible way:

1. The redex is a *quantum redex*, i.e. a subterm of the kind $(\lambda\langle x_1,\ldots,x_n\rangle.L)\langle r_1,\ldots,r_n\rangle$ and $\alpha = \mathsf{q}.\beta$. In this case the computational step does not modify the quantum register and the qvs ($\mathcal{Q} = \mathcal{Q}'$ and $\mathcal{QV} = \mathcal{QV}'$), but the reduction 'links' the subterm L to quantum data in $\mathcal{Q}$ by means of quantum variable names $\langle r_1,\ldots,r_n\rangle$.
2. $\alpha = \mathsf{new}$, then the *creation of a new quantum bit* occurs by reducing a term of the shape $\mathtt{new}(c)$ (where c is a classical bit). Such a reduction creates a *new quantum variable name* r (a fresh variable): as a consequence, $\mathcal{Q}' = \mathcal{Q} \otimes |c\rangle$ and $\mathcal{QV}' = \mathcal{QV} \cup \{r\}$. The new quantum variable name is a kind of pointer to the newly created qubit. In the lambda term M' the subterm $\mathtt{new}(c)$ of M is replaced by the quantum variable name r.
3. The redex is a term of the shape $U\langle r_1,\ldots,r_n\rangle$, where U is the name of a unitary operator and $r_1,\ldots,r_n$ are quantum variables. In this case $\alpha = \mathsf{Uq}$ and the reduction corresponds to the application of a *unitary transformation to the quantum register*. No modification in $\mathcal{QV}$ are performed, thus $\mathcal{QV}' = \mathcal{QV}$, because no change to quantum variable names are expected; in the lambda term M the subterm $U\langle r_1,\ldots,r_n\rangle$ will be replaced by $\langle r_1,\ldots,r_n\rangle$, and $\mathcal{Q}'$ is the normalized vector resulting after the application of U on qubits $\langle r_1,\ldots,r_n\rangle$ in $\mathcal{Q}$.

Let us give some further words about the set $\mathscr{K}$, where we assume that the term L is free for π: it should appear redundant, but it plays an important role in Q. Commutative rules permit to commute subterms in order to prevent quantum reductions that can block classical ones. Moreover, commutative rules will be a useful technical tool in the proof of standardization theorem (Section 5.13.1, Theorem 6).

As previously seen, the well-forming rules ensure statically that any term of the form $!M$ cannot contain quantum variables. In order to preserve this property under reduction,

reductions cannot be performed under the scope of a bang.

This constraint is named in the literature as *surface reduction* (Simpson 2005) and preserves, dynamically, no-cloning and no-erasing properties on non-duplicable syntactical objects. The following example shows what happens if a reduction occurs in the scope of a bang.

Example 5 (a wrong computation in Q). Let us consider the following well-formed configuration: $[1, \varnothing, (\lambda!x.cnot\langle x, x\rangle)!(\mathtt{new}(1))]$. Observe that the subterm $!(\mathtt{new}(1))$ is a duplicable term because it does not yet contain references to quantum data. The following is a correct computation (we do not reduce in the scope of the bang and we perform firstly the $\mathsf{c}.\beta$-reduction):

$$\begin{aligned}
[1, \varnothing, (\lambda!x.cnot\langle x, x\rangle)!(\mathtt{new}(1))] &\to_{\mathsf{c}.\beta} [1, \varnothing, cnot\langle \mathtt{new}(1), \mathtt{new}(1)\rangle] \\
&\xrightarrow{2}_{\mathsf{new}} [|p \mapsto 1\rangle \otimes |q \mapsto 1\rangle, \{p, q\}, cnot\langle p, q\rangle)] \\
&\to_{\mathsf{Uq}} [|p \mapsto 1\rangle \otimes |q \mapsto 0\rangle, \{p, q\}, \langle p, q\rangle].
\end{aligned}$$

However, if we permitted to reduce under the scope of the bang (namely reducing the subterm $\mathtt{new}(1)$ before executing the $\mathsf{c}.\beta$-reduction), we would obtain the computation:

$$\begin{aligned}[1, \varnothing, (\lambda!x.cnot\langle x, x\rangle)!(\mathtt{new}(1))] &\to_{\mathtt{new}} [|p \mapsto 1\rangle, \{p\}, (\lambda!x.cnot\langle x, x\rangle)!(p)] \\ &\to_{\mathsf{q}.\beta} [|p \mapsto 1\rangle, \{p\}, cnot\langle p, p\rangle)].\end{aligned}$$

Notice that we have duplicated the quantum variable p, creating a *double reference* to the same qubit. As a consequence we could apply a binary unitary transform (cnot) to a single qubit (the one referenced by p), which is not compatible with the basic principles of quantum computing.

Let us remark that the *relation* $\to_\alpha$ *is neither a call-by-value nor a call-by-name strategy*. In Q the only limitation is that we forbid reductions under the scope of a '!', and nevertheless, confluence holds in a very strong sense (see Section 5.12.2). This is in contrast with the calculus developed in Selinger and Valiron (2006), where a call-by-value strategy is indeed necessary, even if we do not take into account the non-deterministic effects of the measurement operator (see Section 7.2).

Some definitions on configurations (very reminiscent of equivalent definitions on lambda terms) are in order now. The notion of a well-formed judgement can be extended to configurations in a natural way:

Definition 8 (well-formed configuration). A configuration $[\mathcal{Q}, \mathcal{QV}, M]$ is said to be *well formed* iff there is an environment Γ such that $\Gamma \vdash M$ is well formed.

In the following, by *configuration* we mean *well-formed configuration.*

We define now *normal forms* and *computations.*

Definition 9 (normal form). A configuration $C \equiv [\mathcal{Q}, \mathcal{QV}, M]$ is said to be in *normal form* iff there is no D such that $C \to D$. Let us denote by NF the set of configurations in normal form.

Definition 10 (computations). If C_0 is a configuration, a *computation* of length $\varphi \leqslant \omega$ starting with C_0 is a sequence of configurations $\{C_i\}_{i<\varphi}$ such that for all $0 < i < \varphi$, $C_{i-1} \to C_i$ and either $\varphi = \omega$ or $C_{\varphi-1} \in \mathsf{NF}$.

If a computation starts with a configuration $[\mathcal{Q}_0, \mathcal{QV}_0, M_0]$ such that $\mathcal{QV}_0$ is empty (and, therefore, $\mathbf{Q}(M_0)$ is empty itself), then at each step i the set $\mathcal{QV}_i$ coincides with the set $\mathbf{Q}(M_i)$:

Proposition 1. Let $\{[\mathcal{Q}_i, \mathcal{QV}_i, M_i]\}_{i<\varphi}$ be a computation, such that $\mathbf{Q}(M_0) = \varnothing$. Then for every $i < \varphi$ we have $\mathcal{QV}_i = \mathbf{Q}(M_i)$.

Proof. Observe that if $[\mathcal{Q}, \mathbf{Q}(M), M] \to [\mathcal{Q}', \mathcal{QV}', M']$ then by induction on reduction rules we immediately have that $\mathcal{QV}' = \mathbf{Q}(M')$ whenever $\mathcal{QV} = \mathbf{Q}(M)$, and conclude. □

Notation 5. In the rest of the paper, $[\mathcal{Q}, M]$ denotes the configuration $[\mathcal{Q}, \mathbf{Q}(M), M]$.

5.11.1. *Examples of computations in* Q. In the following examples we work with fixed lengths of the inputs. In Q it is possible to deal with infinite encodings: an example can be found in Section 5.14.1, where we will code (infinite) circuit families.

Example 6 (EPR states). We define a lambda term representing a quantum circuit that generates an EPR state. EPR states are entangled quantum states used by Einstein, Podolsky and Rosen in a famous thought experiment on quantum mechanics (1935).

EPR states can be easily obtained by means of **cnot** and Hadamard's unitary operator **H**. The general schema of the term is

$$\mathsf{M} \equiv \lambda\langle x, y\rangle.(cnot\langle Hx, y\rangle)).$$

The term M takes 2-qubits as input and then gives as output an EPR (entangled) state.

We give an example of computation, with $[1, \mathsf{M}\ \langle \mathtt{new}(0), \mathtt{new}(1)\rangle]$ as initial configuration:

$$\begin{aligned}
[1, \mathsf{M}\ \langle \mathtt{new}(0), \mathtt{new}(1)\rangle] &\to^{2}_{\mathsf{new}} [|p \mapsto 0\rangle \otimes |q \mapsto 1\rangle, (\lambda\langle x, y\rangle.(cnot\langle Hx, y\rangle))\langle p, q\rangle] \\
&\to_{\mathsf{q}.\beta} [|p \mapsto 0\rangle \otimes |q \mapsto 1\rangle, (cnot\langle Hp, q\rangle)] \\
&\to_{\mathsf{Uq}} \left[\frac{|p \mapsto 0\rangle + |p \mapsto 1\rangle}{\sqrt{2}} \otimes |q \mapsto 1\rangle, (cnot\langle p, q\rangle)\right] \\
&\to_{\mathsf{Uq}} \left[\frac{|p \mapsto 0, q \mapsto 0\rangle + |p \mapsto 1, q \mapsto 1\rangle}{\sqrt{2}}, \langle p, q\rangle\right].
\end{aligned}$$

After some reduction steps, two quantum variables p and q appear in the term and the quantum register is modified accordingly. Finally, constants for unitary operators corresponding to **cnot** and **H** are applied to the quantum state. The quantum register

$$\frac{|p \mapsto 0, q \mapsto 0\rangle + |p \mapsto 1, q \mapsto 1\rangle}{\sqrt{2}}$$

is the so-called β_{00} EPR state.

Example 7 (exchange). Consider the following lambda term:

$$\mathsf{L} \equiv \lambda\langle x, y\rangle.(\lambda\langle a, b\rangle.cnot\langle b, a\rangle)((\lambda\langle w, z\rangle.cnot\langle z, w\rangle)(cnot\langle x, y\rangle))$$

L builds a quantum circuit that performs the exchange of a pair of qubits.

$$\begin{aligned}
&[1, \mathsf{L}\ \langle \mathtt{new}(1), \mathtt{new}(0)\rangle] \\
\overset{2}{\to}\ &[|p \mapsto 1\rangle \otimes |q \mapsto 0\rangle, (\lambda\langle x, y\rangle.(\lambda\langle a, b\rangle.cnot\langle b, a\rangle)((\lambda\langle w, z\rangle.cnot\langle z, w\rangle)(cnot\langle x, y\rangle))\langle p, q\rangle] \qquad (1) \\
\to_{\mathsf{q}.\beta}\ &[|p \mapsto 1\rangle \otimes |q \mapsto 0\rangle, (\lambda\langle a, b\rangle.cnot\langle b, a\rangle)((\lambda\langle w, z\rangle.cnot\langle z, w\rangle)cnot\langle p, q\rangle])] \\
\to_{\mathsf{Uq}}\ &[|p \mapsto 1\rangle \otimes |q \mapsto 1\rangle, (\lambda\langle a, b\rangle.cnot\langle b, a\rangle)((\lambda\langle w, z\rangle.cnot\langle z, w\rangle)\langle p, q\rangle)] \\
\to_{\mathsf{q}.\beta}\ &[|p \mapsto 1\rangle \otimes |q \mapsto 1\rangle, (\lambda\langle a, b\rangle.cnot\langle b, a\rangle)cnot\langle q, p\rangle] \\
\to_{\mathsf{Uq}}\ &[|p \mapsto 0\rangle \otimes |q \mapsto 1\rangle, (\lambda\langle a, b\rangle.cnot\langle b, a\rangle)\langle q, p\rangle] \\
\to_{\mathsf{q}.\beta}\ &[|p \mapsto 0\rangle \otimes |q \mapsto 1\rangle, (cnot\langle p, q\rangle)] \\
\to_{\mathsf{Uq}}\ &[|p \mapsto 0\rangle \otimes |q \mapsto 1\rangle, \langle p, q\rangle]. \qquad (2)
\end{aligned}$$

Notice that the values attributed to p and q in the underlying quantum register are exchanged between configurations (1) and (2).

5.12. *Some good classical properties for* Q

5.12.1. *Subject reduction.* Even if Q is type-free, term formation is syntactically controlled by means of well-forming rules. It is therefore necessary to prove that *the class of well-formed terms is closed by reduction.* For Q, a *subject reduction theorem* holds in a suitable form.

A 'standard' property such as subject reduction is of course an expected property of each good defined language. In this case it is also required by the fact that a particular kind of variable, quantum variables, can be created dynamically during the computation. This effect is due to the presence of new reductions $[\mathcal{Q}, \mathcal{QV}, \mathtt{new}(c)] \rightarrow_{\mathsf{new}} [\mathcal{Q} \otimes |r \mapsto c\rangle, \mathcal{QV} \cup \{r\}, r]$ which creates a new qubit from the classical constant c and a quantum variable r.

Thus, the subject reduction theorem is formulated with respect to quantum variables dynamically created in the computational step:

Theorem 2 (subject reduction).
If $d \triangleright \Gamma \vdash M$ and $[\mathcal{Q}, \mathcal{QV}, M] \rightarrow [\mathcal{Q}', \mathcal{QV}', M']$ then $\triangleright\ \Gamma, \mathcal{QV}' - \mathcal{QV} \vdash M'$.

Proof. The proof is by induction on the height of d and by cases on the last rule r of d. In order to prove subject reduction, we need a number of intermediate results. Firstly we have to prove that the weakening rule is admissible in the set of well-forming rules, i.e. we have to prove the following statement:

Weakening For each derivation d, if $d \triangleright \Gamma \vdash M$ and x does not occur in Γ, then $\triangleright \Gamma, !x \vdash M$.

The proof is by induction on the derivation d.

Moreover, as usual, the proof of subject reduction requires suitable *substitution lemmata.* We need three substitution lemmata, according to the different kind of variables we have in the syntax:

Substitution lemma – linear case: if $\triangleright \Lambda_1, !\Delta, x \vdash M$ and $\triangleright \Lambda_2, !\Delta \vdash N$, with $\Lambda_1 \cap \Lambda_2 = \varnothing$, then $\triangleright \Lambda_1, \Lambda_2, !\Delta \vdash M\{N/x\}$.

Substitution lemma – nonlinear case: if $\triangleright \Lambda_1, !\Delta, !x \vdash M$ and $\triangleright !\Delta \vdash !N$, then $\triangleright \Lambda_1, !\Delta \vdash M\{N/x\}$.

Substitution lemma – quantum case: for every nonempty sequence $x_1, \ldots, x_n$ if $\triangleright \Lambda, !\Delta, x_1, \ldots, x_n \vdash M$ and $r_1, \ldots, r_n \notin \Lambda$, then $\triangleright \Lambda, !\Delta, r_1, \ldots, r_n \vdash M\{r_1/x_1, \ldots, r_n/x_n\}$.

All proofs are by induction on the derivation and by cases on the last rule.

The proof of subject reduction proceeds by induction on the derivation of $[\mathcal{Q}, \mathcal{QV}, M] \rightarrow [\mathcal{Q}', \mathcal{QV}', M']$. Let us show some cases:

— r is app and the reduction rule is

$$[\mathcal{Q}, \mathcal{QV}, (\lambda\langle x_1, \ldots, x_n\rangle.P)N] \rightarrow_{\mathsf{l}.\beta} [\mathcal{Q}, \mathcal{QV}, P\{\langle r_1/x_1, \ldots, r_n/x_n\rangle\}] \quad \mathsf{q}.\beta$$

(the application generates a redex). Suppose we have the following derivation d:

$$\dfrac{\dfrac{\begin{array}{c}d_1\\ \vdots\\ \Lambda_1, !\Delta, x_1, \ldots, x_n \vdash P\end{array}}{\Lambda_1, !\Delta \vdash \lambda\langle x_1, \ldots, x_n\rangle.P}\,\mathsf{lam1} \qquad \begin{array}{c}d_2\\ \vdots\\ !\Delta, r_1, \ldots, r_n \vdash \langle r_1, \ldots, r_n\rangle\end{array}}{\Lambda_1, \Lambda_2, !\Delta \vdash (\lambda\langle x_1, \ldots, x_n\rangle.P)N}\,\mathsf{app}$$

Let us consider the reduction
$[\mathcal{Q}, \mathcal{QV}, (\lambda\langle x_1, \ldots, x_n\rangle.P)\langle r_1, \ldots, r_n\rangle] \to_{\mathsf{q}.\beta} [\mathcal{Q}, \mathcal{QV}, P\{r_1/x_1, \ldots, r_n/x_n\}]$. We note that the reduction does not modify the $\mathcal{QV}$ set, so we just have to apply substitution lemma – quantum case to d_1 and $d_2 : \triangleright\Lambda_1, !\Delta \vdash P\{r_1/x_1, \ldots, r_n/x_n\}$.

— r is new

$$\dfrac{!\Delta \vdash c}{!\Delta \vdash \mathtt{new}(c)}\,\mathsf{new}$$

We have the following reduction rule:

$$[\mathcal{Q}, \mathcal{QV}, \mathtt{new}(c)] \to [\mathcal{Q} \otimes |p \mapsto c\rangle, \mathcal{QV} \cup \{p\}, p].$$

By means of

$$\dfrac{}{!\Delta, p \vdash p}\,\mathsf{q-var}$$

we obtain the result.

— r is app and the reduction rule is

$$[\mathcal{Q}, \mathcal{QV}, L((\lambda\langle x_1, \ldots, x_n\rangle.P)N)] \to_{\mathsf{l.cm}} [\mathcal{Q}, \mathcal{QV}, (\lambda\langle x_1, \ldots, x_n\rangle.LP)N] \quad \mathsf{l.cm}$$

Note that the reduction rule does not modify $\mathcal{Q}$ and $\mathcal{QV}$. So, from derivation:

$$\dfrac{\begin{array}{c}d_1\\ \vdots\\ \Lambda_1, !\Delta \vdash L\end{array} \qquad \dfrac{\dfrac{\begin{array}{c}d_2\\ \vdots\\ \Lambda_2', !\Delta, x_1, \ldots, x_n \vdash P\end{array}}{\Lambda_2', !\Delta \vdash \lambda\langle x_1, \ldots, x_n\rangle.P}\,\mathsf{lam1} \qquad \begin{array}{c}d_3\\ \vdots\\ \Lambda_2'', !\Delta \vdash N\end{array}}{\Lambda_2, !\Delta \vdash (\lambda\langle x_1, \ldots, x_n\rangle.P)N}\,\mathsf{app}}{\Lambda_1, \Lambda_2, !\Delta \vdash L((\lambda\langle x_1, \ldots, x_n\rangle.P)N)}\,\mathsf{app}$$

we exhibit a derivation of $\Lambda_1, \Lambda_2, !\Delta \vdash (\lambda\langle x_1, \ldots, x_n\rangle.LP)N$:

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}d_1\\ \vdots\\ \Lambda_1, !\Delta \vdash L\end{array} \qquad \begin{array}{c}d_2\\ \vdots\\ \Lambda_2', !\Delta, x_1, \ldots, x_n \vdash P\end{array}}{\Lambda_1, \Lambda_2', !\Delta, x_1, \ldots, x_n \vdash LP}\,\mathsf{app}}{\Lambda_1, \Lambda_2', !\Delta \vdash \lambda\langle x_1, \ldots, x_n\rangle.LP}\,\mathsf{lam1} \qquad \begin{array}{c}d_3\\ \vdots\\ \Lambda_2'', !\Delta \vdash N\end{array}}{\Lambda_1, \Lambda_2, !\Delta \vdash (\lambda\langle x_1, \ldots, x_n\rangle.LP)N}\,\mathsf{app}$$

Other cases are similar. For full details see Zorzi (2009). □

As a consequence of subject reduction, the set of well-formed configurations is closed under reduction:

Corollary 1. If M is well formed and $[\mathcal{Q}, \mathcal{QV}, M] \xrightarrow{*} [\mathcal{Q}', \mathcal{QV}', M']$ then $[\mathcal{Q}', \mathcal{QV}', M']$ is well formed.

Another immediate corollary (provable by induction) is the following 'weakening property':

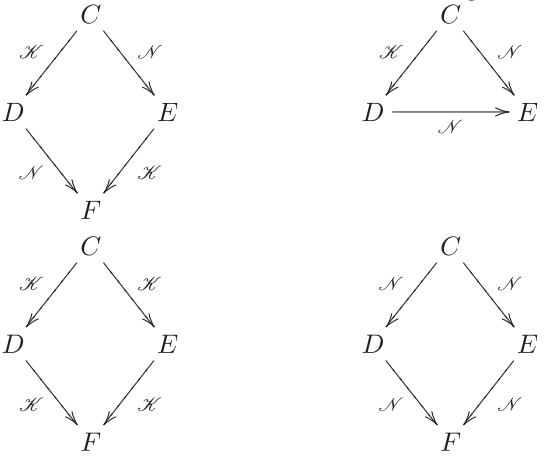
Corollary 2. If $\triangleright\, \Gamma \vdash M$ and $[\mathcal{Q}, \mathcal{QV}, M] \xrightarrow{*} [\mathcal{Q}', \mathcal{QV}', M']$ then $\triangleright\ \Gamma, \mathcal{QV}' - \mathcal{QV} \vdash M$.

5.12.2. *Confluence.* In Q, it is possible to show that strong normalization and weak normalization are equivalent properties. This is due to the fact that confluence holds in a very strong sense, thanks to surface reduction, which implies that in Q it is not possible to erase diverging terms (Simpson 2005).

To prove many-step confluence, an 'imperfect' version of one-step confluence is proved.

One-step confluence cannot hold in its standard formulation because of the presence of commutative rules. For example, starting from the configuration $[\mathcal{Q}, \mathcal{QV}, (\lambda\pi.M)((\lambda x.N)L)]$, then both $[\mathcal{Q}, \mathcal{QV}, (\lambda\pi.M)((\lambda x.N)L)] \to_{\mathcal{N}} [\mathcal{Q}, \mathcal{QV}, (\lambda\pi.M)(N\{L/x\})]$ and $[\mathcal{Q}, \mathcal{QV}, (\lambda\pi.M)((\lambda x.N)L)] \to_{\mathcal{K}} [\mathcal{Q}, \mathcal{QV}, (\lambda x.(\lambda\pi.M)N)L] \to_{\mathcal{N}} [\mathcal{Q}, \mathcal{QV}, (\lambda\pi.M)(N\{L/x\})]$ are legal reduction steps.

Graphically, in presence of commutative rules, the following cases can occur:



This situation is represented in our version of one-step confluence:

Proposition 2 (one-step confluence). Let C, D, E be configurations with $C \to_\alpha D$, $C \to_\beta E$ and $D \neq E$. Then:

1. If $\alpha \in \mathcal{K}$ and $\beta \in \mathcal{K}$, then there is F with $D \to_{\mathcal{K}} F$ and $E \to_{\mathcal{K}} F$.
2. If $\alpha \in \mathcal{N}$ and $\beta \in \mathcal{N}$, then there is F with $D \to_{\mathcal{N}} F$ and $E \to_{\mathcal{N}} F$.
3. If $\alpha \in \mathcal{K}$ and $\beta \in \mathcal{N}$, then either $D \to_{\mathcal{N}} E$ or there is F with $D \to_{\mathcal{N}} F$ and $E \to_{\mathcal{K}} F$.

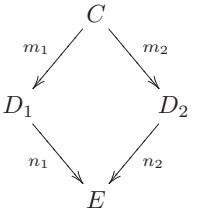
Proof.
Let $C = [\mathcal{Q}, \mathcal{QV}, M]$. By induction on the lambda term M.

□

Thanks to Proposition 2 it is possible to prove the general statement about reduction sequences of arbitrary length:

Theorem 3 (confluence). Let C, D_1, D_2 be configurations with $C \xrightarrow{m_1} D_1$ and $C \xrightarrow{m_2} D_2$. Then, there is a configuration E with $D_1 \xrightarrow{n_1} E$ and $D_2 \xrightarrow{n_2} E$ with $n_1 \leqslant m_2$, $n_2 \leqslant m_1$ and $n_1 + m_1 = n_2 + m_2$.

Graphically:



Proof. By induction on the length of the reduction sequence, and by cases depending on the outcome of Proposition 2. The proof is quite long, for full details see Zorzi (2009) and Dal Lago *et al.* (2010). □

By means of confluence results, it is possible to prove the uniqueness of normal forms in Q.

Theorem 4 (uniqueness of normal forms). Any configuration C has at most one normal form.

Our very strong notion of confluence implies that:

strong and weak normalization are equivalent properties of configurations

as stated in the following theorem:

Theorem 5. A configuration C is strongly normalizing iff C is weakly normalizing.

To prove Theorem 5, we exploit another feature of Q: in a Q computation it is not possible to build an infinite sequence of commuting reductions. This is Lemma 1:

Lemma 1. The relation $\rightarrow_{\mathscr{K}}$ is strongly normalizing. In other words, there cannot be any infinite sequence $C_1 \rightarrow_{\mathscr{K}} C_2 \rightarrow_{\mathscr{K}} C_3 \rightarrow_{\mathscr{K}} \ldots$.

Proof. Define the size $|M|$ of a term M as the number of symbols in it. Moreover, define the abstraction size $|M|_\lambda$ of M as the sum over all subterms of M in the form $\lambda\pi.N$, of $|N|$. Clearly $|M|_\lambda \leqslant |M|^2$. Moreover, if $[\mathcal{Q}, \mathcal{QV}, M] \rightarrow_{\mathscr{K}} [\mathcal{Q}, \mathcal{QV}, N]$, then $|N| = |M|$ but $|N|_\lambda > |M|_\lambda$. This concludes the proof. □

We are able now to prove Theorem 5:

Proof of Theorem 5. Weak normalization implies strong normalization. Suppose, by way of contradiction, that C is weakly normalizing but not strongly normalizing. This implies there is a configuration D in normal form and sequence of $m \in \mathbb{N}$ reduction steps from C to D. Since C is not strongly normalizing, there is also an infinite sequence $C \equiv C_1, C_2, C_3, \ldots$ with $C_1 \to C_2 \to C_3 \to \ldots$ From this infinite sequence, we can extract a sequence of $m+1$ reduction steps due to Lemma 1. Applying Proposition 3, we get a configuration F and a sequence of one reduction step from D to F. However, such a sequence cannot exist, because D is normal. □

This conclude the 'classical analysis' of the calculus Q. In the following section, we will state and prove new interesting quantum properties.

5.13. *Some desirable quantum properties for* Q

In the previous section, we explained two classical properties such as subject reduction and confluence. These results confirm that the classical part of Q, (i.e. the control), is well designed. However, Q is developed for quantum computing: we describe now two important quantum features of Q. Firstly, a standardization theorem is stated and proved, showing that computational steps in Q can be performed in a particular order, in which all quantum reductions can be postponed. The standardization theorem strengthens the common idea that a universal quantum computer should consist of a classical device 'setting up' a quantum circuit that is then fed with an input.

Secondly, Q is shown to be at least as expressive as the quantum circuit families model.

5.13.1. *Standardization theorem: quantum computations in a standard form.* Standard forms of computations are an interesting topic in many computational models. For example, in the probabilistic model, it was proved that a computation on a Turing machine can be standardized into a computation separating stochastic and deterministic parts.

In Q a similar result holds: from each computation, an equivalent computation in standard form can be extracted. Informally, in a *standard computation*, computational steps are performed in the following order: classical reductions, new reductions and finally quantum reductions.

Remember the subsets of $\mathscr{L}$ of Definition 7.
Let $C \to_\alpha D$ and let M be the relevant redex in C; if $\alpha \in \mathscr{Q}$ the redex M is called *quantum*, if $\alpha \in \mathscr{C}$ the redex M is called *classical.*

Different relevant redexes characterize different kinds of configurations:

Definition 11 (classes of configurations).

— A configuration C is called *nonclassical* if $\alpha \in n\mathscr{C}$ whenever $C \to_\alpha D$. Let NCL be the set of non-classical configurations.
— A configuration C is called *essentially quantum* if $\alpha \in \mathscr{Q}$ whenever $C \to_\alpha D$. Let EQT be the set of essentially quantum configurations.

For example, $C = [\mathcal{Q}, \mathcal{QV}, \mathtt{new}(1)] \in \mathsf{NCL}$ and $C = [\mathcal{Q}, \mathcal{QV}, \lambda.\langle x_1, \ldots, x_k\rangle.\langle x_1, \ldots, x_k\rangle (\langle r_1, \ldots, r_k\rangle)] \in \mathsf{EQT}$.

We are able now to define standard computations:

Definition 12 (CNQ computation). A CNQ computation starting with a configuration C is a computation $\{C_i\}_{i<\varphi}$ such that $C_0 \equiv C$, $\varphi \leqslant \omega$ and:

1. for every $1 < i+1 < \varphi$, if $C_{i-1} \rightarrow_{n\mathscr{C}} C_i$ then $C_i \rightarrow_{n\mathscr{C}} C_{i+1}$;
2. for every $1 < i+1 < \varphi$, if $C_{i-1} \rightarrow_{\mathscr{Q}} C_i$ then $C_i \rightarrow_{\mathscr{Q}} C_{i+1}$.

A CNQ computation is built on three different and ordered stages:

— first phase of the computation – classical reductions: this phase builds a lambda term representing a quantum circuit, without any interaction with the underlined quantum register;
— second phase of the computation – new reductions: this phase prepares the input, i.e the quantum register without introducing any superposition;
— third phase of the computation – quantum reductions: in this phase the proper quantum steps are performed, and unitary operators are applied to qubits (possibly introducing superposition). Essentially, the circuit built in the first phase is applied to the quantum register built in the second phase.

From an abstract perspective, we can consider a particular class of terms, called *quantum relevant terms* (see Definition 14), as quantum circuit generators; this will be an important technical ingredient in the proof of the expressive equivalence between Q and the quantum circuit families model.

Theorem 6 (standardization). For every computation $\{C_i\}_{i<\varphi}$ such that $\varphi \in \mathbb{N}$ there is a CNQ computation $\{D_i\}_{i<\xi}$ such that $C_0 \equiv D_0$ and $C_{\varphi-1} \equiv D_{\xi-1}$.

In order to prove Theorem 6 we need some intermediate results. Informally, we need to prove that the class NCL of non-classical configurations is closed under new reduction, while the class EQT of essentially quantum configurations is closed under quantum reduction. This means that we need to prove that a new redex cannot generate a classical redex, and that a quantum redex cannot generate a new redex.

Lemma 2.

i. If $C \in \mathsf{NCL}$ and $C \rightarrow_{\mathsf{new}} D$ then $D \in \mathsf{NCL}$.
ii. If $C \in \mathsf{EQT}$ and $C \rightarrow_{\mathscr{Q}} D$ then $D \in \mathsf{EQT}$.

Proof. i and ii follow from rules' inspection. One has to define a notion of context (a term with one hole) and proceeds by induction on the structure of the context. For a complete and detailed proof see Zorzi (2009) and Dal Lago *et al.* (2010). □

By means of Lemma 2 we can prove the standardization theorem.

Proof of standardization theorem
We build a CNQ computation in three steps:

1. Let us start to reduce $D_0 \equiv C_0$ by using $\mathscr{C}$ reductions as much as possible. By Theorem 5 we must obtain a finite reduction sequence $D_0 \rightarrow_{\mathscr{C}} \ldots \rightarrow_{\mathscr{C}} D_k$ s.t. $0 \leqslant k$ and no $\mathscr{C}$ reductions are applicable to D_k.
2. Reduce D_k by using new reductions as much as possible. By Theorem 5 we must obtain a finite reduction sequence $D_k \rightarrow_{\mathsf{new}} \ldots \rightarrow_{\mathsf{new}} D_j$ s.t. $k \leqslant j$ and no new reductions are applicable to D_j. Note that by Lemma 2 such reduction steps cannot generate classical redexes and in particular no classical redex can appear in D_j.
3. Reduce D_j by using $\mathscr{Q}$ reductions as much as possible. By Theorem 5 we must obtain a finite reduction sequence $D_j \rightarrow_{\mathscr{Q}} \ldots \rightarrow_{\mathscr{Q}} D_m$ such that $j \leqslant m$ and no $\mathscr{Q}$ reductions are applicable to D_m. Note that by Lemma 2 such reduction steps cannot generate $\mathscr{C}$ redexes or new redexes and in particular neither $\mathscr{C}$ nor new reductions are applicable to D_m. Therefore D_m *is in normal form.*

The reduction sequence $\{D_i\}_{i<m+1}$ is such that $D_0 \rightarrow_{\mathscr{C}} \ldots \rightarrow_{\mathscr{C}} D_k \rightarrow_{\mathsf{new}} \ldots \rightarrow_{\mathsf{new}} D_j \rightarrow_{\mathscr{Q}} \ldots \rightarrow_{\mathscr{Q}} D_m$ is a CNQ computation. By Theorem 4 we observe that $C_{\varphi-1} \equiv D_m$, which implies the thesis.

□

The standardization theorem explains the structure of measurement-free quantum computations, confirming the idea of a composite architecture device-server (i.e. the advisability of quantum data-classical control approach). Moreover, it plays an important role in the proof of the equivalence of Q with quantum circuit families: the possibility to perform quantum steps after the classical ones will be extensively used in Theorem 9.

The following is another example of a Q-computation (that is also in standard form):

Example 8 (Deutsch's algorithm). Deutsch's algorithm is the first quantum algorithm to have been defined. It allows one to compute a global property of a function by combining results from two components of a superposition (Deutsch 1985). We refer here to the *Deutsch's algorithm* as presented in Nielsen and Chuang (2000). We propose an encoding in Q's syntax.

Let $\mathbf{W}_f$ be the unitary transformation such that $\mathbf{W}_f|c_1c_2\rangle = |c_1, c_2 \oplus f(c_1)\rangle$ (for any given boolean function f and $c_1, c_2 \in \{0,1\}$), and let $\mathbf{H}$ be the Hadamard transform.

The general quantum circuit that implements Deutsch's algorithm is represented by the following lambda term:

$$\mathsf{D} \equiv \lambda\langle x, y\rangle.((\lambda\langle w, z\rangle.\langle Hw, z\rangle)(W_f\langle Hx, Hy\rangle)).$$

Deutsch's algorithm makes use of *quantum parallelism* and *interference* (Section 4) in order to determine whether f is a constant function by means of a single evaluation of $f(x)$.

In order to perform such a task, we first evaluate the normal form of $[1, \mathsf{D}\langle \mathtt{new}(0), \mathtt{new}(1)\rangle]$.

$$
\begin{aligned}
& [1, \mathsf{D}\langle \mathtt{new}(0), \mathtt{new}(1)\rangle] \\
\to^{2}_{\mathsf{new}} \; & [|p \mapsto 0\rangle \otimes |q \mapsto 1\rangle, (\lambda\langle x, y\rangle.(\lambda\langle w, z\rangle.\langle Hw, z\rangle)(W_f\langle Hx, Hy\rangle))\langle p, q\rangle] \\
\to_{\mathsf{q},\beta} \; & [|p \mapsto 0\rangle \otimes |q \mapsto 1\rangle, (\lambda\langle w, z\rangle.\langle Hw, z\rangle)(W_f\langle Hp, Hq\rangle)] \\
\to_{\mathsf{Uq}} \; & \left[\frac{|p \mapsto 0\rangle + |p \mapsto 1\rangle}{\sqrt{2}} \otimes |q \mapsto 1\rangle, (\lambda\langle w, z\rangle.\langle Hw, z\rangle)(W_f\langle p, Hq\rangle)\right] \\
\to_{\mathsf{Uq}} \; & \left[\frac{|p \mapsto 0\rangle + |p \mapsto 1\rangle}{\sqrt{2}} \otimes \frac{|q \mapsto 0\rangle - |q \mapsto 1\rangle}{\sqrt{2}}, (\lambda\langle w, z\rangle.\langle Hw, z\rangle)(W_f\langle p, q\rangle)\right] \\
= \; & \left[\frac{|p \mapsto 0, q \mapsto 0\rangle}{2} - \frac{|p \mapsto 0, q \mapsto 1\rangle}{2} + \frac{|p \mapsto 1, q \mapsto 0\rangle}{2} + \frac{|p \mapsto 1, q \mapsto 1\rangle}{2}, (\lambda\langle w, z\rangle.\langle Hw, z\rangle)(W_f\langle p, q\rangle)\right] \\
\to_{\mathsf{Uq}} \; & \left[\frac{|p \mapsto 0, q \mapsto 0 \oplus f(0)\rangle}{2} - \frac{|p \mapsto 0, q \mapsto 1 \oplus f(0)\rangle}{2} + \frac{|p \mapsto 1, q \mapsto 0 \oplus f(1)\rangle}{2} + \frac{|p \mapsto 1, q \mapsto 1 \oplus f(1)\rangle}{2},\right. \\
& \left.(\lambda\langle w, z\rangle.\langle Hw, z\rangle)\langle p, q\rangle)\right] \\
\to_{\mathsf{q},\beta} \; & \left[\frac{|p \mapsto 0, q \mapsto 0 \oplus f(0)\rangle}{2} - \frac{|p \mapsto 0, q \mapsto 1 \oplus f(0)\rangle}{2} + \frac{|p \mapsto 1, q \mapsto 0 \oplus f(1)\rangle}{2} + \frac{|p \mapsto 1, q \mapsto 1 \oplus f(1)\rangle}{2},\right. \\
& \left.\langle Hp, q\rangle\right] \\
\to_{\mathsf{Uq}} \; & \left[\frac{|p \mapsto 0\rangle + |p \mapsto 1\rangle}{\sqrt{2}} \otimes \frac{|q \mapsto 0 \oplus f(0)\rangle}{2} - \frac{|p \mapsto 0\rangle + |p \mapsto 1\rangle}{\sqrt{2}} \otimes \frac{|q \mapsto 1 \oplus f(0)\rangle}{2} + \right. \\
& \left. \frac{|p \mapsto 0\rangle - |p \mapsto 1\rangle}{\sqrt{2}} \otimes \frac{|q \mapsto 0 \oplus f(1)\rangle}{2} + \frac{|p \mapsto 0\rangle - |p \mapsto 1\rangle}{\sqrt{2}} \otimes \frac{|q \mapsto 1 \oplus f(1)\rangle}{2}, \langle p, q\rangle\right]
\end{aligned}
$$

We have two cases:

— f is a constant function; i.e. $f(0) \oplus f(1) = 0$.
In this case the normal form may be rewritten as (by means of simple algebraic manipulations):

$$[(-1)^{f(0)}|p \mapsto 0\rangle \otimes \frac{|q \mapsto 0\rangle - |q \mapsto 1\rangle}{\sqrt{2}}, \langle p, q\rangle].$$

— f is not a constant function; i.e. $f(0) \oplus f(1) = 1$.
In this case the normal form may be rewritten as:

$$[(-1)^{f(0)}|p \mapsto 1\rangle \otimes \frac{|q \mapsto 0\rangle - |q \mapsto 1\rangle}{\sqrt{2}}, \langle p, q\rangle].$$

If we measure (by means of a final external apparatus) the first qubit p of the term $\langle p, q\rangle$ in the normal form configuration, we obtain 0 if f is constant and 1 otherwise.

It is essential to stress again that quantum parallelism alone is not enough: a quantum algorithm also has the ability to extract information about more than one of the value $f(x)$ from a superposition state at the same time, as Deutsch's algorithm done.

Note that the computation is in *standard form*, i.e. all new reductions come before all quantum reductions (in this example any classical reduction does not occur).

5.14. *Infinite computations in* Q

What about infinite computations? Of course, in Q it is possible to represent diverging lambda terms, and so *non-terminating computations* occur. From a quantum computation point of view, in the absence of an explicit measurement operator non-terminating computations are not particularly interesting, because there is no final measurable

quantum state, and consequently the transformations of the quantum register are inaccessible.

The extension of standardization to the infinite case makes this observation explicit. First of all, observe that we cannot have an infinite sequence of $n\mathscr{C}$ reductions, as the following lemma shows:

Lemma 3. The relation $\rightarrow_{n\mathscr{C}}$ is strongly normalizing (i.e. there cannot be any infinite sequence $C_1 \rightarrow_{n\mathscr{C}} C_2 \rightarrow_{n\mathscr{C}} C_3 \rightarrow_{n\mathscr{C}} \ldots$).

Proof. Define the size $|M|$ of a term M as the number of symbols in it, observe that if $[\mathcal{Q}, \mathcal{QV}, M] \rightarrow_{n\mathscr{C}} [\mathcal{Q}, \mathcal{QV}, N]$ then $|N| < |M|$ and conclude. □

As a consequence of Lemma 3 only the first, purely classical part of a CNQ computation can diverge:

Proposition 3. Any infinite CNQ computation only includes classical reduction steps.

The following is a simple example of a non-normalizing term in Q (and then of an infinite reduction): given $M = \lambda!x.(x!x)$, clearly $M(!M) \rightarrow M(!M)$.

We can easily state and prove the standardization theorem also for infinite computations:

Theorem 7 (standardization for infinite computations). For every non-terminating computation $\{C_i\}_{i<\omega}$ there is a CNQ computation $\{D_i\}_{i<\omega}$ such that $C_0 \equiv D_0$.

Proof. We build the CNQ computation in the following way: start to reduce $D_0 \equiv C_0$ by using $\mathscr{C}$ reductions as much as possible. This procedure cannot end, otherwise we would contradict Lemma 3 and Theorem 5. □

Infinite computations are quite irrelevant in the measurement-free case; on the contrary, they play a fascinating role in the presence of an explicit measurement operator (see Section 6).

5.14.1. *Expressive power: equivalence with quantum circuit families* In this section, we analyse the expressive power of Q, showing that it is 'quantum Turing complete'. We will prove that the expressive power of Q is equivalent to a particular class of quantum circuit families, and consequently (via the results of Nishimura and Ozawa (2009)) we obtain the equivalence with the model of quantum Turing machines (as defined by Bernstein and Vazirani (1997)).
Noticeably,

one single lambda term in Q represents *an entire family of quantum circuits.*

Two important considerations are now in order.

A lambda term is a finite object (we build it by a finite number of applications of well-forming rules), and in particular it contains only a finite number of constants for unitary transformations. As a consequence, we will restrict our attention to the class of *finitely generated* quantum circuit families (Definition 32, Appendix A.2).

Moreover, in order to encode a quantum circuit family by means of a single lambda term, we will exploit the classical content of Q. Whereas, in the previous examples we

provided finite encodings, the encoding of quantum circuit families is an *infinite encoding*, as a quantum circuit family defines an infinite class of functions. The pure *linear* part of the calculus (it can easily be obtained forgetting modalities and with little adjustments to the well-forming rules) is not enough: it permits us to encode only *one single circuit*, i.e. a particular element of the family which represents the function for one particular input size.

Before showing some aspects of the encoding, it is mandatory to say a few words about the classical content of Q and the relationship with the standard untyped λ-calculus. The untyped λ-calculus can be embedded in Q, but this does not mean that the encoding of quantum circuit families is trivial. First of all, the translation into Q has to be done carefully, because β-reduction in the lambda calculus does not correspond to surface reduction in a calculus like Q: it is easy to build a non-normalizable classical lambda term M such that its (call-by-name) translation $\overline{M}$ is clearly a normal form in Q. Moreover, surface reduction shares a stricter relationship with weak head reduction: in this case if M rewrites to N by weak head reduction, then $\overline{M}$ rewrites to $\overline{N}$ in Q, but the converse is not true. Thus, the embedding into Q it is not immediate.

On the other hand, it is well known that the lambda calculus is Turing complete for any reasonable encoding and this still holds for Q, even if the notion of reduction is stricter. But showing 'classical' Turing completeness for Q (i.e. to prove that Q has at least the expressive power of a Turing machine) does not imply, for free, the encoding of quantum circuit families. This only implies that it is possible to compute the *code* D_n of the nth circuit C_n of any quantum circuit family from input n, i.e. the 'Gödel's number' of C_n. Since we want to evaluate C_n inside Q, we need to prove that the correspondence $D_n \mapsto C_n$ is itself representable in Q and since the way quantum circuits are represented and evaluated in Q has nothing to do with the encoding of the calculus itself, this is *not* a consequence of the alleged (classical) Turing completeness of Q.

In this section, we will show that each finitely generated quantum circuit family can be captured by a subclass of Q terms, called *quantum relevant* terms (Definition 14). In particular, we will prove that the class of functions captured by quantum relevant terms is exactly the class computed by quantum circuit families.

Informally, a quantum relevant term is a Q term which produces a list of quantum variables when applied to a list of classical bits, something like the quantum version of functions on natural numbers in classical recursion theory. It is clear that the way in which we encode natural numbers and data structures is a crucial point. Moreover, we are dealing also with quantum data which are, in general, nonerasable and non-duplicable. Thus, a numbers of constraints have to be added to the encoding, in order to respect quantum features. In this paper, we will give only the idea of the encoding. All details can be found in Dal Lago *et al.* (2009) and Zorzi (2009).

We encode data, classical and quantum, using some variations on Scott's numerals (Wadsworth 1980). In the case of quantum data, a strongly linear discipline is enforced through a slightly different encoding. What is crucial from a computational point of view is the way a quantum relevant term can possibly modify the underlying quantum register. The encoding of data structures is long but quite standard. We summarize the essential definitions about natural numbers and lists in the following paragraphs.

5.14.2. *Natural numbers.* In the λ-calculus Q *natural numbers are encoded nonlinearly*. Any natural number is duplicable by construction, since it has the shape $!M$ for some M. We have:

$$\begin{aligned} \lceil 0 \rceil &= !\lambda!x.\lambda!y.y \\ \forall n \quad \lceil n+1 \rceil &= !\lambda!x.\lambda!y.x\lceil n \rceil \end{aligned}$$

In the following, for each natural number n we denote $\lceil n \rceil$ as its encoding in Q. As usual, encodings of basic constructors such as successor and predecessor are provided.

Among other constructors, recursion is of course an essential ingredient. Thus, it is easy to define a lambda term rec such that, for each $M \in \mathsf{Q}$, $\mathsf{rec}!M \equiv M!(\mathsf{rec}!M)$.

Structural recursion $\mathsf{rec}^{\mathsf{nat}}$ on natural numbers is definable by means of rec.

5.14.3. *Lists.* Whereas, natural numbers are duplicable syntactical objects, in Q *lists are encoded linearly*.

Given any sequence $M_1, \ldots, M_n$ of terms, we can build a term $[M_1, \ldots, M_n]$ encoding the sequence as follows, by induction on n:

$$\begin{aligned} [] &= \lambda!x.\lambda!y.y; \\ [M, M_1 \ldots, M_n] &= \lambda!x.\lambda!y.xM[M_1, \ldots, M_n]. \end{aligned}$$

The occurrences of M and $[M_1, \ldots, M_n]$ which are part of $[M, M_1, \ldots, M_n]$ do not lie in the scope of any bang operator.

It is routine to encode standard operations on list such as construction, destruction, selection, extraction. Iteration is available on lists, too.

Representable functions in Q are defined as follows:

Definition 13. A (partial) function $f : \mathbb{N}^n \to \mathbb{N}$ is *representable* iff there is a term M_f such that:

— Whenever $M_f\lceil m_1 \rceil \ldots \lceil m_n \rceil$ has a normal form N (with respect to $\overset{*}{\to}_{\mathscr{C}}$), then $N \equiv \lceil m \rceil$ for some natural number m.
— $M_f\lceil m_1 \rceil \ldots \lceil m_n \rceil \overset{*}{\to}_{\mathscr{C}} \lceil m \rceil$ iff $f(m_1, \ldots, m_n)$ is defined and equal to m.

As already mentioned, Q reduction relation is quite different from the standard λ-calculus reduction. In Dal Lago *et al.* (2009) and Zorzi (2009), it is possible to find the full proof of the following proposition, which ensures the equivalence between representable and partially recursive functions:

Proposition 4. The class of representable functions in Q coincides with the class of partial recursive functions (on natural numbers).

We now formally define the class of quantum relevant terms.

In the following, given any subset $\mathscr{S}$ of $\mathscr{L}$, the expression $C \Downarrow_{\mathscr{S}} D$ means that $C \overset{*}{\to}_{\mathscr{S}} D$ and D is in normal form with respect to the relation $\to_{\mathscr{S}}$. $C \Downarrow D$ stands for $C \Downarrow_{\mathscr{L}} D$.

Confluence and equivalence between weakly normalizing and strongly normalizing configurations authorize the following definition:

Definition 14 (quantum relevant terms). A term M is called *quantum relevant* (shortly, *qrel*) if it is well formed and for each list $![!c_1, ..., !c_n]$ ($c_i \in \{0,1\}$) there is a quantum register $\mathcal{Q}$ and a natural number m such that $[1, \varnothing, M![!c_1, ..., !c_n]] \Downarrow [\mathcal{Q}, \{r_1, \ldots, r_m\}, [r_1, \ldots, r_m]]$.

The class of functions captured by quantum relevant terms coincides with the class of functions which can be computed by finitely generated quantum circuit families.

The situation is summarized by the following diagram

$$\begin{array}{ccc} [\emptyset, M] & \cdots\cdots\succ & [\emptyset, M\lceil n\rceil] \rightarrow_{\mathscr{L}} [\mathcal{Q}, [r_i, \ldots, r_k]] \\ \uparrow Th.\ 8 \;\big|\; \downarrow Th\ 9 & & \Big\updownarrow \\ (K_i)_{i<\omega} & \cdots\cdots\succ & K_{h(\lceil n\rceil)}(\lceil n\rceil) = \mathcal{Q} \end{array}$$

where M is a quantum relevant term and $(K_i)_i$ is a finitely generated uniform quantum circuit family. We sketch here both directions of the equivalence. Detailed proofs are given in Dal Lago *et al.* (2009) and Zorzi (2009).

We start showing that each finite family of quantum circuit can be 'represented' in Q, i.e that Q is at least as computationally strong as finitely generated uniform quantum circuit families (Definition 32).

Theorem 8. For every finitely generated uniform family of quantum circuits (f, g, h) there is a quantum relevant term $M_{f,g,h}$ such that for each $c_1, \ldots, c_n$ ($c_i \in \{0,1\}$), the following two conditions are equivalent:

— $[1, \varnothing, M_{f,g,h}![!c_1, ..., !c_n]] \Downarrow [\mathcal{Q}, \{r_1, \ldots, r_m\}, [r_1, \ldots, r_m]]$;
— $m = f(n)$ and $\mathcal{Q} = \Phi_{f,g,h}(c_1, \ldots, c_n)$.

Proof. We give the principal ideas of the proof, which is essentially an encoding work. Since any recursive function can be represented in Q, we can assume that f, g and h are representable whenever (f, g, h) is a uniform family of circuits.

Recall that:

— $f : \mathbb{N} \to \mathbb{N}$ computes the input dimension.
— $h : \mathbb{N} \to \mathbb{N}$ returns the index of the suitable circuit into the family with respect to the input dimension.

An essential ingredient is the possibility to represent, in the calculus, elementary permutations on terms. The nth elementary permutation of m elements (where $1 \leqslant n < m$) is the function which maps n to $n+1$, $n+1$ to n and any other elements in the interval $1, \ldots, m$ to itself. Any (finite) permutation can be effectively decomposed into a product of elementary permutations. In particular, it is possible to define a term M_{el} which *computes the $n+1$th elementary permutation on lists*: for every list $[N_1, \ldots, N_m]$ with $m > n$, $M_{el}(\lceil n\rceil)$ $[N_1, \ldots, N_m] \rightarrow^*_{\mathscr{C}} [N_1, \ldots, N_{n-1}, N_{n+1}, N_n, N_{n+2}, \ldots, N_m]$.

Now, we are able to prove that any finitely generated family of circuits can be represented in Q. Suppose that $\{\mathbf{K}_i\}_{i\in\mathbb{N}}$ is an effective enumeration of quantum circuits (Appendix A.2) and assume it is based on an elementary set of unitary operators.

Suppose that for every $i \in \mathbb{N}$, the circuit $\mathbf{K}_i$ is

$$\mathbf{U}^i_1, r^{i,1}_1, \ldots, r^{i,p(i,1)}_1, \ldots, \mathbf{U}^i_{k(i)}, r^{i,1}_{k(i)}, \ldots, r^{i,p(i,k(i))}_{k(i)}$$

where $p : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ and $k : \mathbb{N} \to \mathbb{N}$ are computable functions. Since (f, g, h) is finitely generated, there is a finite family of gates $\mathcal{G} = \{\mathbf{U}_1, \ldots, \mathbf{U}_b\}$ such that for every $i \in \mathbb{N}$ the gates $\mathbf{U}^{h(i)}_1, \ldots, \mathbf{U}^{h(i)}_{k(i)}$ are all from $\mathcal{G}$. Let $ar(1), \ldots, ar(b)$ be the arities of $\mathbf{U}_1, \ldots, \mathbf{U}_b$. Since the enumeration $\{\mathbf{K}_i\}_{i \in \mathbb{N}}$ is effective, we can assume the existence of a recursive function $u : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that $u(i, j) = x$ iff $\mathbf{U}^{h(i)}_j$ is $\mathbf{U}_x$. Moreover, we know that for every $i \in \mathbb{N}$ and for every $1 \leqslant j \leqslant k(h(i))$, the variables

$$r^{h(i),1}_j, \ldots, r^{h(i),p(h(i),k(h(i)))}_j$$

are distinct and in $\{r_1, \ldots, r_{f(h(i))}\}$. So, there are permutations π^i_j of $\{1, \ldots, f(h(i))\}$ such that $\pi^i_j(x) = y$ iff $r^{h(i),x}_j = r_y$ for every $1 \leqslant x \leqslant p(h(i), k(h(i)))$. Let ρ^i_j be the inverse of π^i_j. Clearly, both π^i_j and ρ^i_j can be effectively computed from i and j by means of a suitable decomposition into elementary permutations. As a consequence, the following functions are partial recursive (in the 'classical' sense):

— A function $r : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ which, given (i, j) returns the number of elementary permutations of $\{1, \ldots, f(h(i))\}$ in which π^i_j can be decomposed.
— A function $q : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that $q(i, j, x) = y$ iff the xth elementary permutation of $\{1, \ldots, f(h(i))\}$ in which π^i_j can be decomposed is the yth elementary permutation.
— A function $s : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ which, given (i, j) returns the number of elementary permutations of $\{1, \ldots, f(h(i))\}$ in which ρ^i_j can be decomposed.
— A function $t : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that $t(i, j, x) = y$ iff the xth elementary permutation of $\{1, \ldots, f(h(i))\}$ in which ρ^i_j can be decomposed is the yth elementary permutation.

The term representing the quantum circuit family is built out of the following subterms:

— M_{init} that, given a list L of boolean constants and a natural number $\lceil n \rceil$, computes the input list for $\mathbf{K}_{h(n)}$ from L.
— M_{circ}, that, given a natural number $\lceil n \rceil$ builds a term computing the unitary transformations involved in $\mathbf{K}_{h(n)}$ acting on lists of quantum variables with length $f(n)$.
— M_{length}, which applied to a list $[!N_1, \ldots, !N_m]$ computes the length (the representation $\lceil m \rceil$ of the number of the elements).

Now, the term $M_{f,g,h}$ is:

$$\lambda !x.(M_{circ}(M_{length}x))(M_{init}\, !x(M_{length}x)).$$

For the full encoding of $M_{f,g,h}$ see, Dal Lago *et al.* (2009) and Zorzi (2009). □

We now state the converse of Theorem 8, completing the proof of the equivalence with quantum circuit families. In the following, we will use the compact notation $[\mathcal{Q}, M]$ for configurations (Note 5, Section 5.10).

Theorem 9. For each qrel M there is a uniform quantum circuit family (f, g, h) such that for each list $c_1, \ldots, c_n$ the following two conditions are equivalent:

— $[1, M![!c_1, \ldots, !c_n]] \Downarrow [\mathcal{Q}, [r_1, \ldots, r_m]]$
— $m = f(n)$ and $\mathcal{Q} = \Phi_{f,g,h}(c_1, \ldots, c_n)$.

The proof of Theorem 9 is quite intuitive. Noticeably, here the standardization theorem plays a central role. In fact, we can assume that all non-quantum reduction steps can be done before any quantum reduction step. Thus, it is possible to work on quantum relevant terms in separate stages, building the quantum circuit family in two phases.

Firstly, we consider reduction steps in the set $n\mathscr{Q} = \mathscr{L} - \mathscr{Q}$ and we show that from a quantum relevant term it is possible to extract, regardless of the input vector, an essentially quantum term. Let M be a qrel term, let $![!c_1, \ldots, !c_n], ![!d_1, \ldots, !d_n]$ be two lists of bits (with the same length) and suppose that $[1, M![!c_1, \ldots, !c_n]] \Downarrow_{n\mathscr{Q}} [\mathcal{Q}, N]$ (notice that N cannot contain any boolean constant, since M is assumed to be a qrel). By applying exactly the same computation steps that lead from $[1, M![!c_1, \ldots, !c_n]]$ to $[\mathcal{Q}, N]$, we can prove that $[1, M![!d_1, \ldots, !d_n]] \Downarrow_{n\mathscr{Q}} [\mathcal{Q}', N]$, where $\mathcal{Q}$ and $\mathcal{Q}'$ live in the same Hilbert space $\mathcal{H}(\mathbf{Q}(N))$ and are both elements of the computational basis. Moreover, any computational step leading from $[1, M![!c_1, \ldots, !c_n]]$ to $[\mathcal{Q}, N]$ is effective, i.e. it is intuitively computable (in the classical sense). Therefore, by the Church–Turing's thesis we obtain the following:

Proposition 5. For each qrel M there exists a term N and two total computable functions $f : \mathbb{N} \to \mathbb{N}$ and $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that for every $n \in \mathbb{N}$ and for every $c_1, \ldots, c_n$, $[1, M![!c_1, \ldots, !c_n] \Downarrow_{n\mathscr{Q}} [|r_1 \mapsto c_{g(n,1)}, \ldots, r_{f(n)} \mapsto c_{g(n,f(n))}\rangle, N]$, where we conventionally set $c_0 \equiv 0$ and $c_{n+1} \equiv 1$.

Note that, at this stage, we have built the quantum circuit from the description function with respect to the input and we have also built the input, i.e. the quantum register.

Let us now consider computations from essentially quantum terms.

Let $[\mathcal{Q}, M] \in \mathsf{EQT}$ and let us suppose that $[\mathcal{Q}, M] \Downarrow_{\mathscr{Q}} [\mathcal{Q}', [r_1, \ldots, r_m]]$. Then $\mathcal{Q}$ and $\mathcal{Q}'$ live in the same Hilbert space $\mathcal{H}(\mathbf{Q}(M)) = \mathcal{H}(\mathbf{Q}([r_1, \ldots, r_m])) = \mathcal{H}(\{r_1, \ldots, r_m\})$. The sequence of reductions in this computation allows us to build in an effective way a unitary transformation $\mathbf{U}$ such that $\mathcal{Q}' = \mathbf{U}_{\langle r_1, \ldots, r_m \rangle}(\mathcal{Q})$. This is captured by the following:

Proposition 6. Let M be a term only containing quantum redexes. Then, there is a circuit $\mathbf{K}$ such that $\mathcal{Q}' = U_{\mathbf{K}}(\mathcal{Q})$ whenever $[\mathcal{Q}, M] \Downarrow_{\mathscr{Q}} [\mathcal{Q}', M']$. Moreover, $\mathbf{K}$ is generated by gates appearing in M. Furthermore $\mathbf{K}$ can be effectively computed from M.

The proof of Theorem 9 follows as a direct consequence of Propositions 5 and 6.

6. Adding measurement operators: theoretical problems and technical instruments

If a quantum state does not interact with the ambient (i.e. with something able to perform a measurement) its evolution is deterministic: in fact, as explained in Sections 2 and 3, it persists in a linear, unitary evolution by means of unitary operators (Postulate III).

In Q, data and control are separated: on the one hand, quantum data live in the quantum world, and all quantum register transformations are unitary and invertible; on the other hand, the classical part of the calculus, the control, enjoys good computational properties. In Q it is not possible to classically observe the content of the quantum

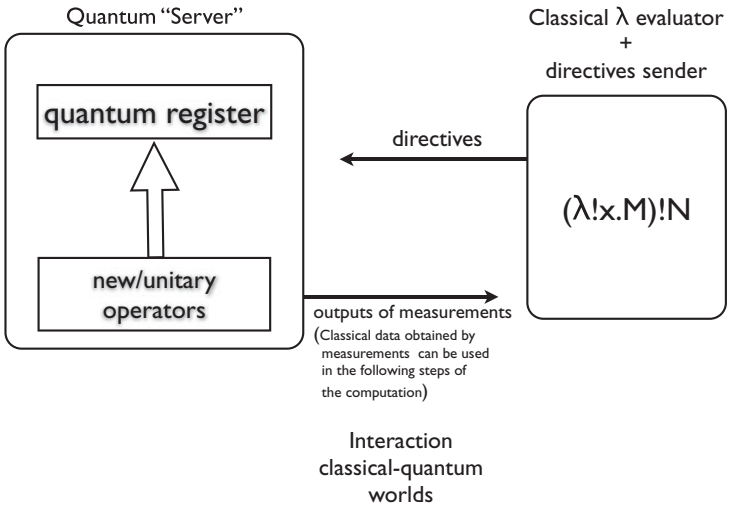


Fig. 4. Quantum client-server architecture with feedback.

register: the language of terms does not include any measurement operator that, applied to a quantum variable, has the effect of observing the value of the related qubit. This is in contrast, for example, with Selinger and Valiron's λ_{sv} (where such a measurement operator is indeed part of the language of terms) and with other calculi for quantum computation (like the measurement calculus (Danos *et al.* 2007)), where the possibility of observing is even more central (see Section 7). A measurement-free calculus is good for foundational investigations, and the choice of performing a unique, final measurement at the end of the computation does not modify the expressive power or the theoretical generality of the calculus. In Q we are potentially able to encode any quantum computable functions, but we may not be able to 'reformulate' algorithms that provide intermediate measurement steps (even if we know that the same algorithms can be rewritten in terms of a unique final measurement). In an arbitrary quantum algorithm, quantum data can be observed: this interaction with the quantum state gives a set of *observables*, i.e. a probabilistic distribution of classical data which can be used in the following steps of the computation.

An explicit measurement constructor becomes an essential ingredient if we want to encode the process of iterated/intermediate measurements of algorithms like Shor's one.

The extension of Q with a measurement operator $\mathtt{meas}(\cdot)$ (in the style of λ_{sv}) is not problematic from a linguistic point of view. The architecture in Figure 1 can be updated with a feedback from the quantum server to the client, which uses the observed data as classical constants (see Figure 4).

The problem is rather to prove that the extended calculus again enjoys good quantum and computational properties.

6.1. *The measurement problem: from physics to computer science*

The so-called *measurement problem* is one of the most fascinatingly tricky points of quantum mechanics. Different approaches to it give different interpretations of the theory, leaving some questions still unanswered.

In quantum computing Postulate V induces less foundational choices, but the possibility to observe quantum data changes the features of the computational framework in a sensible way.

An explicit measurement operator in the syntax allows an observation at an intermediate step of the computation. In quantum calculi the intended meaning of a measurement is to observe the status of a possibly superposed quantum bit: this observation gives as output a classical bit and the two possible outcomes 0 and 1 can be observed with two probabilities summing to 1. The output of a measurement can be successively used in the following step of the computation as a classical datum. For example, let us consider the following configuration (written in the extended syntax of Q^*, Section 6.2):

$$C = [1, \varnothing, (\lambda !x.(\mathsf{if}\ x\ \mathsf{then}\ 0\ \mathsf{else}\ 1))(\mathtt{meas}(H(\mathsf{new}(0))))]$$

From C it is possible to build two different computations, depending on the output measurement of the quantum state $\frac{1}{2}(|0\rangle + |1\rangle)$ created by the redex $\mathtt{meas}(H(\mathsf{new}(0)))$: the expression $\mathtt{meas}(H(\mathsf{new}(0)))$ can evaluate (with equal probability) to 0 or to 1. The first case yields the normal form $C_1 \equiv [p \mapsto 1, \{p\}, 0]$ with probability $1/2$, and the second case yields the normal form $C_2 \equiv [p \mapsto 0, \{p\}, 1]$ with probability $1/2$.

Thus a measurement redex breaks confluence, importing a *probabilistic behaviour* into the computations. In this situation, a 'classical' notion of confluence seems to be irremediably lost.

Let us change our point of view. Taking the configuration C of the previous example, we can say that it rewrites deterministically to the probability distribution of configurations $\{p_1 : C_1, p_2 : C_2\}$, with $p_1 = p_2 = \frac{1}{2}$. A computation involving measurement should be seen as a rewriting between distributions of configurations, which takes into account all the possible results of measurements steps. Under this point of view, an interesting question is the following: is it possible to preserve equivalence between different strategies? In other words, if divergence coming from a *single* measurement redex is irrecoverable, is it possible to avoid divergence coming from *different* redexes?

In this second case, the question has a positive answer. The proof of this confluence result, which involves probabilistic instruments, is the main result about Q^*. In Section 6.5, the *strong confluence result* (the equivalence between different strategies) will be stated and proved for *mixed states*[†], distributions of Q^* configurations. It is easy to extend in a natural way the notion of reduction between configurations to reduction between mixed

[†] This name is 'imported' from physics. When the knowledge about a quantum system is complete, the wave function $|\phi\rangle$ fully describes the system, as seen in Postulate I. When the knowledge of the system is not complete, another mathematical instrument is used, the so-called *density matrix* $\rho = \sum_i p_i |\phi_i\rangle$. The density matrix describes a mixture of pure quantum states i.e. a set, $\{p_i : |\phi\rangle\}$ of pure states with linked probabilities. The system described by the density matrix can be described with probability p_i by the vector $|\phi_i\rangle$. For a full treatment of this topic, see for example Isham (1995).

states. Reduction on mixed states is a non-probabilistic relation, and in this way quantum computation with intermediate measurements can be described into a standard rewriting system with a standard notion of confluence.

We will not reason directly with mixed states. We proceed first by studying a new notion of computation, called *probabilistic computation*, which extends the usual notion of computation as a sequence of reduction steps. A probabilistic computation is a possibly infinite tree, in which the binary choice corresponds to the two different outcomes of a measurement. The set of leaves of a probabilistic computation is consequently a probabilistic distribution of configurations.

It is possible to reformulate the confluence theorem in the following way: *for every configuration C and for every configuration in normal form D, there is a fixed real number p such that the probability of observing D when reducing C is always p, independently of the reduction strategy* (this is essentially the main result of Section 6.4).

Note that our result does not come only from the fact that distributions of possible outcomes are considered instead of single results: confluence is a feature of Q^* and it is not an obvious property. For example the quantum lambda calculus defined in Selinger and Valiron (2006) does not enjoy such a property: call-by-value and call-by-name strategies give different distributions of outcomes (see Section 7, where we will provide an overview on this calculus and a comparative example).

6.2. *Syntax, well-forming rules and computations in Q^**

The syntax of Q^* extends the syntax of Q with two new constructs $\mathsf{if}\,(\cdot)\,\mathsf{then}\,(\cdot)\,\mathsf{else}\,(\cdot)$[†] and $\mathtt{meas}(\cdot)$:

$$
\begin{array}{lll}
x & ::= x_0, x_1, \ldots & \textit{classical variables} \\
r & ::= r_0, r_1, \ldots & \textit{quantum variables} \\
\pi & ::= x \mid \langle x_1, \ldots, x_n \rangle & \textit{linear patterns} \\
\psi & ::= \pi \mid !x & \textit{patterns} \\
B & ::= 0 \mid 1 & \textit{boolean constants} \\
U & ::= U_0, U_1, \ldots & \textit{unitary operators} \\
C & ::= B \mid U & \textit{constants} \\
M & ::= x \mid r \mid !M \mid C \mid \mathtt{new}(M) \mid M_1 M_2 \mid & \\
 & \quad \mathtt{meas}(M) \mid \mathsf{if}\ N\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2 \mid & \\
 & \quad \langle M_1, \ldots, M_n \rangle \mid \lambda\psi.M \mid & \textit{terms (where } n \geqslant 2\textit{)}
\end{array}
$$

The set of well-forming rules of Q (Figure 2, Section 5.8) is extended with the following two rules:

$$
\frac{\Gamma \vdash M}{\Gamma \vdash \mathtt{meas}(M)}\ \mathsf{meas}
\qquad
\frac{\Lambda \vdash N \quad !\Delta \vdash M_1 \quad !\Delta \vdash M_2}{\Lambda, !\Delta \vdash \mathsf{if}\ N\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2}\ \mathsf{if}
$$

[†] The $\mathsf{if}\,(\cdot)\,\mathsf{then}\,(\cdot)\,\mathsf{else}\,(\cdot)$ constructor can be thought of as syntactic sugar, as, of course, can the boolean constants 0 and 1.

The main important novelty is of course the presence of the explicit construct for measurement, $\mathsf{meas}(\cdot)$, which, informally, takes a (name of) a quantum bit as an argument and returns a classical bit. The measurement operator has an effect on the quantum register. In Q^*, in fact, three kinds of quantum register transformations can occur: 'new operations' (which add qubits to the current state) as in Q; 'unitary transformation' as in Q; and the 'single qubit destructive measurement operator'.

The contraction of a measurement redex as $\mathsf{meas}(q)$ (see reduction rule meas_r in Figure 5, below) corresponds, at the level of a quantum register, to the application of a suitably defined function which maps a normalized vector in a certain Hilbert space into another normalized vector in a smaller dimension Hilbert space. More precisely, for each qvs $\mathcal{QV}$ and for each quantum variable $r \in \mathcal{QV}$, we assume to have two, measurement based, linear transformations of quantum registers: $\mathcal{M}_{r,0}, \mathcal{M}_{r,1} : \mathcal{H}(\mathcal{QV}) \to \mathcal{H}(\mathcal{QV} - \{r\})$ (see Definition 5, Section 2.4). Given a quantum register $\mathcal{Q} \in \mathcal{H}(\mathcal{QV})$, the measurement of the qubit named r in $\mathcal{Q}$ gives the outcome c (with $c \in \{0,1\}$) with probability $p_c = \langle \mathcal{Q}|\mathcal{M}_{r,c}{}^{\dagger}\mathcal{M}_{r,c}|\mathcal{Q}\rangle$ and produces the new quantum register $\frac{\mathcal{M}_{r,c}\mathcal{Q}}{\sqrt{p_c}}$. This probabilistic behaviour also involves the control (i.e. the third component of the configuration, the lambda term). In fact, the reduction relation $\to_\alpha$ (as for Q, defined between configurations), has to be parameterized by a probability p.

Let $\mathscr{L}^* = \{\mathsf{Uq}, \mathsf{new}, \mathsf{l}.\beta, \mathsf{q}.\beta, \mathsf{c}.\beta, \mathsf{l.cm}, \mathsf{r.cm}, \mathsf{if}_1, \mathsf{if}_2, \mathsf{meas}_r\}$. For every $\alpha \in \mathscr{L}^*$ and for every $p \in \mathbb{R}_{[0,1]}$, we define a relation $\to_\alpha^p \subseteq \mathbf{Conf} \times \mathbf{Conf}$ by the set of reductions in Figure 5. The notation $C \to_\alpha D$ stands for $C \to_\alpha^1 D$.

We will still adopt *surface reduction*, as for Q: reduction is not allowed in the scope of any ! operator.

Furthermore, *we forbid reduction in N and P in the term* $\mathsf{if}\ M\ \mathsf{then}\ N\ \mathsf{else}\ P$.

We distinguish again three particular subsets of $\mathscr{L}^*$:

Definition 15 (subsets of $\mathscr{L}^*$).

— $\mathscr{K} = \{\mathsf{l.cm}, \mathsf{r.cm}\}$
— $\mathscr{N} = \mathscr{L}^* - (\mathscr{K} \cup \{\mathsf{meas}_r\})$
— $n\mathscr{M} = \mathscr{L}^* - \{\mathsf{meas}_r\}$.

In the following, we write $M \to_\alpha N$ meaning that there are $\mathcal{Q}$, $\mathcal{QV}$, $\mathcal{R}$ and $\mathcal{RV}$ such that $[\mathcal{Q}, \mathcal{QV}, M] \to_\alpha [\mathcal{R}, \mathcal{RV}, N]$. Similarly for the notation $M \to_{\mathscr{S}} N$ where $\mathscr{S}$ is a subset of $\mathscr{L}^*$.

In Q (Section 5) infinite computations do not play an important role, because the (classical) divergent phase of the computation does not permit the access to quantum reduction steps. In presence of explicit measurement the situation is quite different.

In Q^*, an infinite computation can tend to a configuration which is essentially different from the configurations in the computation itself.

In other words, an infinite numbers of computational steps can be necessary in order to reach a probabilistic distribution of finite configurations. The following examples provides an explanation of this phenomenon.

$$[\mathcal{Q}, \mathcal{QV}, (\lambda x.M)N] \to^1_{\mathsf{l}.\beta} [\mathcal{Q}, \mathcal{QV}, M\{N/x\}] \qquad [\mathcal{Q}, \mathcal{QV}, (\lambda !x.M)!N] \to^1_{\mathsf{c}.\beta} [\mathcal{Q}, \mathcal{QV}, M\{N/x\}]$$

$$[\mathcal{Q}, \mathcal{QV}, (\lambda\langle x_1, \ldots, x_n\rangle.M)\langle r_1, \ldots, r_n\rangle] \to^1_{\mathsf{q}.\beta} [\mathcal{Q}, \mathcal{QV}, M\{r_1/x_1, \ldots, r_n/x_n\}]$$

$$[\mathcal{Q}, \mathcal{QV}, \mathsf{if}\ 1\ \mathsf{then}\ M\ \mathsf{else}\ N] \to^1_{\mathsf{if}_1} [\mathcal{Q}, \mathcal{QV}, M]$$

$$[\mathcal{Q}, \mathcal{QV}, \mathsf{if}\ 0\ \mathsf{then}\ M\ \mathsf{else}\ N] \to^1_{\mathsf{if}_2} [\mathcal{Q}, \mathcal{QV}, N]$$

$$[\mathcal{Q}, \mathcal{QV}, U\langle r_{i_1}, \ldots, r_{i_n}\rangle] \to^1_{\mathsf{Uq}} [\mathbf{U}_{\langle\langle r_{i_1}, \ldots, r_{i_n}\rangle\rangle}\mathcal{Q}, \mathcal{QV}, \langle r_{i_1}, \ldots, r_{i_n}\rangle]$$

$$[\mathcal{Q}, \mathcal{QV}, \mathtt{meas}(r)] \to^{p_c}_{\mathsf{meas}_r} [\mathcal{M}_{r,c}(\mathcal{Q}), \mathcal{QV} - \{r\}, !c] \qquad (c \in \{0,1\} \text{ and } p_c = \langle\mathcal{Q}|\mathsf{m}_{r,c}{}^\dagger\mathsf{m}_{r,c}|\mathcal{Q}\rangle \in \mathbb{R}_{[0,1]})$$

$$[\mathcal{Q}, \mathcal{QV}, \mathtt{new}(c)] \to^1_{\mathsf{new}} [\mathcal{Q} \otimes |r \mapsto c\rangle, \mathcal{QV} \cup \{r\}, r] \qquad (r \text{ is fresh})$$

$$[\mathcal{Q}, \mathcal{QV}, L((\lambda\pi.M)N)] \to^1_{\mathsf{l.cm}} [\mathcal{Q}, \mathcal{QV}, (\lambda\pi.LM)N]$$

$$[\mathcal{Q}, \mathcal{QV}, ((\lambda\pi.M)N)L] \to^1_{\mathsf{r.cm}} [\mathcal{Q}, \mathcal{QV}, (\lambda\pi.ML)N]$$

$$\frac{[\mathcal{Q}, \mathcal{QV}, M] \to^p_\alpha [\mathcal{R}, \mathcal{RV}, N]}{[\mathcal{Q}, \mathcal{QV}, , \langle M_1, \ldots, M, \ldots, M_k\rangle] \to^p_\alpha [\mathcal{R}, \mathcal{RV}, \langle M_1, \ldots, N, \ldots, M_k\rangle]}\ \mathsf{t}_i$$

$$\frac{[\mathcal{Q}, \mathcal{QV}, N] \to^p_\alpha [\mathcal{R}, \mathcal{RV}, P]}{[\mathcal{Q}, \mathcal{QV}, MN] \to^p_\alpha [\mathcal{R}, \mathcal{RV}, MP]}\ \mathsf{r.a}$$

$$\frac{[\mathcal{Q}, \mathcal{QV}, M] \to^p_\alpha [\mathcal{R}, \mathcal{RV}, P]}{[\mathcal{Q}, \mathcal{QV}, MN] \to^p_\alpha [\mathcal{R}, \mathcal{RV}, PN]}\ \mathsf{l.a}$$

$$\frac{[\mathcal{Q}, \mathcal{QV}, M] \to^p_\alpha [\mathcal{R}, \mathcal{RV}, N]}{[\mathcal{Q}, \mathcal{QV}, \mathtt{new}(M)] \to^p_\alpha [\mathcal{R}, \mathcal{RV}, \mathtt{new}(N)]}\ \mathsf{in.new}$$

$$\frac{[\mathcal{Q}, \mathcal{QV}, M] \to^p_\alpha [\mathcal{R}, \mathcal{RV}, N]}{[\mathcal{Q}, \mathcal{QV}, \mathtt{meas}(M)] \to^p_\alpha [\mathcal{R}, \mathcal{RV}, \mathtt{meas}(N)]}\ \mathsf{in.meas}$$

$$\frac{[\mathcal{Q}, \mathcal{QV}, M] \to^p_\alpha [\mathcal{R}, \mathcal{RV}, N]}{[\mathcal{Q}, \mathcal{QV}, \mathsf{if}\ M\ \mathsf{then}\ L\ \mathsf{else}\ P] \to^p_\alpha [\mathcal{R}, \mathcal{RV}, \mathsf{if}\ N\ \mathsf{then}\ L\ \mathsf{else}\ P]}\ \mathsf{in.if}$$

$$\frac{[\mathcal{Q}, \mathcal{QV}, M] \to^p_\alpha [\mathcal{R}, \mathcal{RV}, N]}{[\mathcal{Q}, \mathcal{QV}, (\lambda !x.M)] \to^p_\alpha [\mathcal{R}, \mathcal{RV}, (\lambda !x.N)]}\ \mathsf{in}.\lambda_1 \qquad \frac{[\mathcal{Q}, \mathcal{QV}, M] \to^p_\alpha [\mathcal{R}, \mathcal{RV}, N]}{[\mathcal{Q}, \mathcal{QV}, (\lambda\pi.M)] \to^p_\alpha [\mathcal{R}, \mathcal{RV}, (\lambda\pi.N)]}\ \mathsf{in}.\lambda_2$$

Fig. 5. Reduction Rules for Q^*.

Example 9 (infinite computation in Q^*).
Let us consider the configuration

$$E = [1, \varnothing, (\mathsf{Y}!(\lambda !f.\lambda !x.\mathsf{if}\ x\ \mathsf{then}\ 0\ \mathsf{else}\ f(\mathtt{meas}(H(\mathtt{new}(0))))))(\mathtt{meas}(H(\mathtt{new}(0))))]$$

where Y is a fix point operator. Note that after a finite number of reduction steps C rewrites to a distribution in the form $\{\sum_{1<i\leqslant n}\frac{1}{2^i} : [1, \varnothing, 0], 1 - \sum_{1<i\leqslant n}\frac{1}{2^i} : D\}$, and only after an infinite number of reduction steps the distribution $\{1 : [1, \varnothing, 0]\}$ is reached.

As a consequence, the study of infinite computations is significant in this setting.

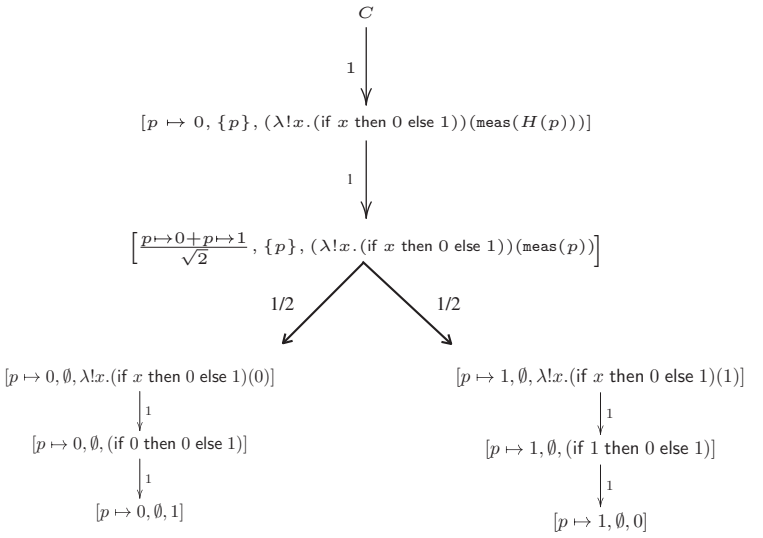


Fig. 6. Example of probabilistic computation.

6.3. *A probabilistic notion of computation*

We introduce here the notion of *probabilistic computation*, which permits one to deal with distributions of possible results and represents the main technical ingredient to retrace confluence in presence of measurements.

A probabilistic computation is a notion more general than a sequence of reductions and less general than the reduction tree. Informally, probabilistic computations are trees in which each node has at most two sons, and in which the binary choice is exactly represented by the two different outcomes (0 or 1) of a qubit measurement. For example, let us take the configuration $C = [1, \varnothing, (\lambda!x.(\mathsf{if}\ x\ \mathsf{then}\ 0\ \mathsf{else}\ 1))(\mathtt{meas}(H(\mathsf{new}(0))))]$ of Section 6.1.

In Figure 6, a probabilistic computation (where we labelled arrows with probabilities) with root C is given.

We will represent computations as (possibly) infinite trees. A (possibly) infinite tree T will be an $(n+1)$-tuple $[R, T_1, \ldots, T_n]$, where $0 \leqslant n \leqslant 2$, R is the *root* of T and $T_1, \ldots, T_n$ are its *immediate subtrees.*

Definition 16 (probabilistic computation).

i. A nonempty set of (possibly) infinite trees $\mathscr{S}$ is said to be a *set of probabilistic computations* if $P \in \mathscr{S}$ iff (exactly) one of the following three conditions holds:

1. $P = [C]$ and $C \in \mathbf{Conf}$.
2. $P = [C, R]$, where $C \in \mathbf{Conf}$, $R \in \mathscr{S}$ has root D and $C \to_{n.\mathscr{M}} D$
3. $P = [(p, q, C), R, Q]$, where $C \in \mathbf{Conf}$, $R \in \mathscr{S}$ has root D, $Q \in \mathscr{S}$ has root E and $C \to^{p_c}_{meas_r} D$, $C \to^{q_{\overline{c}}}_{meas_r} E$ with $p_c, q_{\overline{c}} \in \mathbb{R}_{[0,1]}$, $c = 0, 1$ and $\overline{c}$ is the complement of c;

ii. The set of all probabilistic computations is the largest set $\mathscr{P}$ of probabilistic computations with respect to set inclusion.

iii. The set of finite probabilistic computations is the smallest set $\mathscr{F}$ of probabilistic computations with respect to set inclusion.

In the point i-3 of the definition above, we mean that the configuration D and E are respectively obtained from the outcome 0 or 1 of the measurement of the qubit r. Without loss of generality, we can assume for example that the left subcomputation R (with root D) comes from the observation of 0 and the right subcomputation Q (with root E) comes from the observation of 1. In the following, we will sometimes omit the subscripts c and $\overline{c}$ in the reductions $\rightarrow^{p_c}_{meas_r}$ and $\rightarrow^{q_{\overline{c}}}_{meas_r}$.

Note that the sets $\mathscr{P}$ and $\mathscr{F}$ of Definition 16 exist because of the Knapster–Tarski theorem.

With a little abuse of language, sometimes we will say that the root of $P = [(p, q, C), R, Q]$ is C, diverging from the above definition without any danger of ambiguity. We define now maximal probabilistic computations, the 'lifting' of the normal form notion in this setting.

Definition 17 (maximal probabilistic computation). A probabilistic computation P is *maximal* if for every leaf C in P, $C \in \mathsf{NF}$.

We can give definitions and proofs over *finite* probabilistic computations i.e. over $\mathscr{F}$ by ordinary induction. The same is not true for arbitrary probabilistic computations, since $\mathscr{P}$ is not a well-founded set.

Definition 18 (sub-computation). Let $P \in \mathscr{P}$ be a probabilistic computation. A finite probabilistic computation $R \in \mathscr{F}$ is a *sub-computation* of P, written $R \sqsubseteq P$ iff one of the following conditions is satisfied:

— $R = [C]$ and the root of P is C.
— $R = [C, Q]$, $P = [C, S]$, and $Q \sqsubseteq S$.
— $R = [(p, q, C), Q, S]$, $P = [(p, q, C), U, V]$, $Q \sqsubseteq U$ and $S \sqsubseteq V$.

Since the outcomes of a probabilistic computation P are given by the configurations which appear as leaves of P, it should be useful, for example, to know the probability of observing a normal form, or to count the number of times a certain normal form occurs.

Thus, we state now some 'quantitative properties' of probabilistic computations. Let $\delta : \mathbf{Conf} \rightarrow \{0, 1\}$ be a function defined as $\delta(C) = 0$ if the quantum register of C is 0 and $\delta(C) = 1$ otherwise.

Definition 19. For every *finite* probabilistic computation P and every $C \in \mathsf{NF}$ we define $\mathcal{P}(P, C) \in \mathbb{R}_{[0,1]}$ by induction on the structure of P:

— $\mathcal{P}([C], C) = \delta(C)$;
— $\mathcal{P}([C], D) = 0$ whenever $C \neq D$;
— $\mathcal{P}([C, P], D) = \mathcal{P}(P, D)$;
— $\mathcal{P}([(p, q, C), P, R], D) = p\mathcal{P}(P, D) + q\mathcal{P}(R, D)$;

Informally, $\mathcal{P}(P, C)$ is the probability of observing C as a leaf in P. Similarly, we define $\mathcal{N}(P, C)$,which represents the number of times C appears as a leaf in P:

Definition 20. For every *finite* probabilistic computation P and every $C \in \mathsf{NF}$, we define $\mathcal{N}(P, C) \leqslant \aleph_0$ by induction on the structure of P:

— $\mathcal{N}([C], C) = 1$;
— $\mathcal{N}([C], D) = 0$ whenever $C \neq D$;
— $\mathcal{N}([C, P], D) = \mathcal{N}(P, D)$;.
— $\mathcal{N}([(p, q, C), P, R], D) = \mathcal{N}(P, D) + \mathcal{N}(R, D)$.

In the following we will also refer to $\mathcal{P}(P)$, the probability of observing *any* configuration (in normal form) as a leaf in P, and to $\mathcal{N}(P)$, the number of times *any* configuration appears as a leaf in P. These definitions can be easily obtained from Definitions 19 and 20.

It is possible to extend quantitative properties to *arbitrary*, possibly infinite, probabilistic computations. We exploit the fact that $\mathbb{R}_{[0,1]}$ and $\mathbb{N} \cup \{\aleph_0\}$ are complete lattices (with respect to standard orderings), and we extend the above notions to the case of arbitrary probabilistic computations, by taking the least upper bound over all finite subcomputations. Thus, we have the following definition:

Definition 21. If $P \in \mathscr{P}$ and $C \in \mathsf{NF}$, then:

— $\mathcal{P}(P, C) = \sup_{R \sqsubseteq P} \mathcal{P}(R, C)$,
— $\mathcal{N}(P, C) = \sup_{R \sqsubseteq P} \mathcal{N}(R, C)$,
— $\mathcal{P}(P) = \sup_{R \sqsubseteq P} \mathcal{P}(R)$,
— $\mathcal{N}(P) = \sup_{R \sqsubseteq P} \mathcal{N}(R)$.

Notice that, since $C \in \mathsf{NF}$, for finite P the two notions coincide.

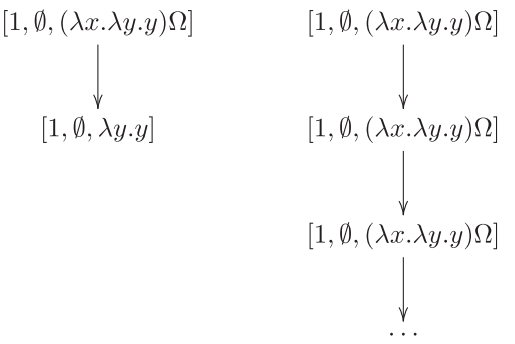
We are now ready to state and prove our confluence result.

6.4. *Strong confluence for* Q^*

In this section, we will prove the main result about Q^*. The confluence result can be informally described in the following way: *any two maximal probabilistic computations P and R with the same root have exactly the same quantitative and qualitative behaviour*, that is to say, the following equations hold for every $C \in \mathsf{NF}$:

— $\mathcal{P}(P, C) = \mathcal{P}(R, C)$: the probability of observing C is independent from the adopted strategy (then all strategies are equivalent).
— $\mathcal{N}(P, C) = \mathcal{N}(R, C)$ and $\mathcal{N}(P) = \mathcal{N}(R)$: the number of (not necessarily distinct) leaves in any probabilistic computation does not depend on the strategy
— $\mathcal{P}(P) = \mathcal{P}(R)$: the probability of converging is not affected by the underlying strategy.

We define our confluence result as 'strong' because equalities like the ones above do *not* even hold for the ordinary λ-calculus. It is easy to show a counterexample. Let us take the lambda term $(\lambda x.\lambda y.y)\Omega$. It is the root of the following two (linear) maximal computations:



The first has one leaf $[1, \varnothing, \lambda y.y]$ and the second one has no leaves at all.

The proof of the strong confluence theorem (Theorem 10) is quite long. We omit non-interesting auxiliary results, but we will dwell on some lemmata which present non-standard proof-theoretical contents or are essential to introduce the theorem. For full details see Zorzi (2009).

As for Q, Q^* does not enjoy the diamond property in a classical sense, because of the presence of commutative reduction rules. The *quasi-one-step confluence* we state below retraces exactly Proposition 2 (Section 5.12.2). In presence of measurements, we also have some 'good' cases induced by measurement (cases 3, 5 and 6 of Proposition 7); unfortunately, during the computations, the case $C \to_{\mathsf{meas}_r} D$ and $C \to_{\mathsf{meas}_r} E$ also occurs, and it cannot be solved.

Proposition 7 (quasi-one-step confluence). Let C, D, E be configurations with $C \to^{p}_{\alpha} D$, $C \to^{s}_{\beta} E$. Then

1. If $\alpha \in \mathscr{K}$ and $\beta \in \mathscr{K}$, then either $D = E$ or there is F with $D \to_{\mathscr{K}} F$ and $E \to_{\mathscr{K}} F$.
2. If $\alpha \in \mathscr{K}$ and $\beta \in \mathscr{N}$, then either $D \to_{\mathscr{N}} E$ or there is F with $D \to_{\mathscr{N}} F$ and $E \to_{\mathscr{K}} F$.
3. If $\alpha \in \mathscr{K}$ and $\beta = \mathsf{meas}_r$, then there is F with $D \to^{s}_{\mathsf{meas}_r} F$ and $E \to_{\mathscr{K}} F$.
4. If $\alpha \in \mathscr{N}$ and $\beta \in \mathscr{N}$, then either $D = E$ or there is F with $D \to_{\mathscr{N}} F$ and $E \to_{\mathscr{N}} F$.
5. If $\alpha \in \mathscr{N}$ and $\beta = \mathsf{meas}_r$, then there is F with $D \to^{s}_{\mathsf{meas}_r} F$ and $E \to_{\mathscr{K}} F$.
6. If $\alpha = \mathsf{meas}_r$ and $\beta = \mathsf{meas}_q$ $(r \neq q)$, then there are $t, u \in \mathbb{R}_{[0,1]}$ and a F such that $pt = su$, $D \to^{t}_{\mathsf{meas}_q} F$ and $E \to^{u}_{\mathsf{meas}_r} F$.

Proof. By induction on M. □

Note that, as for Q, the strong normalization property for $\to_{\mathscr{K}}$ (Lemma 1) still holds.

The proof of the strong confluence theorem (Theorem 10) is essentially based on quasi-one-step confluence and on a suitable version of the *strip lemma* (Barendregt 1984); to state the strip-lemma, we need to compare quantitatively probabilistic computations.

Some definitions are in order now.

We define the *branch degree* $\mathsf{B}(P)$ of a *finite* probabilistic computation P by induction on the structure of P:

— $\mathsf{B}([C]) = 1$.
— $\mathsf{B}([C, P]) = \mathsf{B}(P)$.
— $\mathsf{B}([(p, q, C), P, R]) = \mathsf{B}(P) + \mathsf{B}(R)$.

Please observe that $\mathsf{B}(P) \geqslant 1$ for every P.

We also define the *weight* $\mathsf{W}(P)$ of a *finite* probabilistic computation P by induction on the structure of P:

— $\mathsf{W}([C]) = 0$.
— Let D be the root of P. If $C \to_{\mathscr{K}} D$, then $\mathsf{W}([C,P]) = \mathsf{W}(P)$, otherwise $\mathsf{W}([C,P]) = \mathsf{B}(P) + \mathsf{W}(P)$.
— $\mathsf{W}([(p,q,C),P,R]) = \mathsf{B}(P) + \mathsf{B}(R) + \mathsf{W}(P) + \mathsf{W}(R)$.

The following is a probabilistic variation on the classical *strip lemma* of the λ-calculus. We detail some cases of the proof, which is quite nonstandard.

Lemma 4 (probabilistic strip lemma). Let P be a finite probabilistic computation with root C and positive weight $\mathsf{W}(P)$.

— If $C \to_{\mathscr{N}} D$, then there is R with root D such that $\mathsf{W}(R) < \mathsf{W}(P)$, $\mathsf{B}(R) \leqslant \mathsf{B}(P)$ and for every $E \in \mathsf{NF}$, it holds that $\mathcal{P}(R,E) \geqslant \mathcal{P}(P,E)$, $\mathcal{N}(R,E) \geqslant \mathcal{N}(P,E)$, $\mathcal{P}(R) \geqslant \mathcal{P}(P)$ and $\mathcal{N}(R) \geqslant \mathcal{N}(P)$.
— If $C \to_{\mathscr{K}} D$, then there is R with root D such that $\mathsf{W}(R) \leqslant \mathsf{W}(P)$, $\mathsf{B}(R) \leqslant \mathsf{B}(P)$ and for every $E \in \mathsf{NF}$, it holds that $\mathcal{P}(R,E) \geqslant \mathcal{P}(P,E)$, $\mathcal{N}(R,E) \geqslant \mathcal{N}(P,E)$, $\mathcal{P}(R) \geqslant \mathcal{P}(P)$ and $\mathcal{N}(R) \geqslant \mathcal{N}(P)$.
— If $C \to^{q}_{\mathsf{meas}_r} D$ and $C \to^{p}_{\mathsf{meas}_r} E$, then there are R and Q with roots D and E such that $\mathsf{W}(R) < \mathsf{W}(P)$, $\mathsf{W}(Q) < \mathsf{W}(P)$, $\mathsf{B}(R) \leqslant \mathsf{B}(P)$, $\mathsf{B}(Q) \leqslant \mathsf{B}(P)$ and for every $E \in \mathsf{NF}$, it holds that $q\mathcal{P}(R,E) + p\mathcal{P}(Q,E) \geqslant \mathcal{P}(P,E)$, $\mathcal{N}(R,E) + \mathcal{N}(Q,E) \geqslant \mathcal{N}(P,E)$, $q\mathcal{P}(R) + p\mathcal{P}(Q) \geqslant \mathcal{P}(P)$ and $\mathcal{N}(R) + \mathcal{N}(Q) \geqslant \mathcal{N}(P)$.

Proof. By induction on the structure of P:

— P cannot simply be $[C]$, because $\mathsf{W}(P) \geqslant 1$.
— If $P = [C,S]$, where S has root F and $C \to_{\mathscr{N}} F$, then:
  – Suppose $C \to_{\mathscr{N}} D$. If $D = F$, then the required R is simply S. Otherwise, by Proposition 7 (quasi-one-step-confluence), there is G such that $D \to_{\mathscr{N}} G$ and $F \to_{\mathscr{N}} G$. Now, if S is simply $[F]$, then the required probabilistic computation is simply $[D]$, because neither F nor D are in normal form and, moreover, $\mathsf{W}([D]) = 0 < 1 = \mathsf{W}(P)$. If, on the other hand, S has positive weight we can apply the IH to it, obtaining a probabilistic computation T with root G such that $\mathsf{W}(T) < \mathsf{W}(S)$, $\mathsf{B}(T) \leqslant \mathsf{B}(S)$, $\mathcal{P}(T,H) \geqslant \mathcal{P}(S,H)$ and $\mathcal{N}(T,H) \geqslant \mathcal{N}(S,H)$ for every $H \in \mathsf{NF}$. Then, the required probabilistic computation is $[D,T]$, since

$$\begin{aligned}
\mathsf{W}([D,T]) &= \mathsf{B}(T) + \mathsf{W}(T) < \mathsf{B}(T) + \mathsf{W}(S) \\
&\leqslant \mathsf{B}(S) + \mathsf{W}(S) = \mathsf{W}(P); \\
\mathcal{P}([D,T],H) &= \mathcal{P}(T,H) \geqslant \mathcal{P}(S,H) \\
&= \mathcal{P}(P,H); \\
\mathcal{N}([D,T],H) &= \mathcal{N}(T,H) \geqslant \mathcal{N}(S,H) \\
&= \mathcal{N}(P,H).
\end{aligned}$$

  – Suppose $C \to_{\mathscr{K}} D$. By Proposition 7 one of the following two cases applies:

- There is G such that $D \to_{\mathscr{N}} G$ and $F \to_{\mathscr{K}} G$ Now, if S is simply $[F]$, then the required probabilistic computation is simply $[D,[G]]$, because $\mathsf{W}([D,[G]]) = 1 = \mathsf{W}(P)$. If, on the other hand, S has positive weight we can apply the IH to it, obtaining a probabilistic computation T with root G such that $\mathsf{W}(T) \leqslant \mathsf{W}(S)$, $\mathsf{B}(T) \leqslant \mathsf{B}(S)$ and $\mathcal{P}(T,H) \geqslant \mathcal{P}(T,H)$ for every $H \in \mathsf{NF}$. Then, the required probabilistic computation is $[D,T]$, since

$$\begin{aligned}
\mathsf{W}([D,T]) &= \mathsf{B}(T) + \mathsf{W}(T) \leqslant \mathsf{B}(T) + \mathsf{W}(S) \\
&\leqslant \mathsf{B}(S) + \mathsf{W}(S) = \mathsf{W}(P) \\
\mathcal{P}([D,T],H) &= \mathcal{P}(T,H) \geqslant \mathcal{P}(S,H) \\
&= \mathcal{P}(P,H); \\
\mathcal{N}([D,T],H) &= \mathcal{N}(T,H) \geqslant \mathcal{N}(S,H) \\
&= \mathcal{N}(P,H).
\end{aligned}$$

- $D \to_{\mathscr{N}} F$. The required probabilistic computation is simply $[D,S]$. Indeed:

$$\mathsf{W}([D,S]) = \mathsf{B}(S) + \mathsf{W}(S) = \mathsf{W}([C,S]) = \mathsf{W}([P]).$$

– Suppose $C \to^{q}_{\mathsf{meas}_r} D$ and $C \to^{p}_{\mathsf{meas}_r} E$. By Proposition 7, there are G and H such that $D \to_{\mathscr{N}} G$, $E \to_{\mathscr{N}} H$, $F \to^{q}_{\mathsf{meas}_r} G$, $F \to^{p}_{\mathsf{meas}_r} H$. Now, if S is simply F, then the required probabilistic computations are simply $[D]$ and $[E]$, because neither F nor D nor E are in normal form and, moreover, $\mathsf{W}([D]) = \mathsf{W}([E]) = 0 < 1 = \mathsf{W}(P)$. If, on the other hand, S has positive weight we can apply the IH to it, obtaining probabilistic computations T and U with roots G and H such that $\mathsf{W}(T) < \mathsf{W}(S)$, $\mathsf{W}(U) < \mathsf{W}(S)$, $\mathsf{B}(T) \leqslant \mathsf{B}(S)$, $\mathsf{B}(U) \leqslant \mathsf{B}(S)$, $q\mathcal{P}(T,H) + (p)\mathcal{P}(U,H) \geqslant \mathcal{P}(S,H)$ and $\mathcal{N}(T,H) + \mathcal{N}(U,H) \geqslant \mathcal{N}(S,H)$ for every $H \in \mathsf{NF}$. Then, the required probabilistic computations are $[D,T]$ and $[E,U]$, since

$$\begin{aligned}
\mathsf{W}([D,T]) &= \mathsf{B}(T) + \mathsf{W}(T) < \mathsf{B}(T) + \mathsf{W}(S) \\
&\leqslant \mathsf{B}(S) + \mathsf{W}(S) = \mathsf{W}(P); \\
\mathsf{W}([E,U]) &= \mathsf{B}(U) + \mathsf{W}(U) < \mathsf{B}(U) + \mathsf{W}(S) \\
&\leqslant \mathsf{B}(S) + \mathsf{W}(S) = \mathsf{W}(P).
\end{aligned}$$

Moreover, for every $H \in \mathsf{NF}$

$$\begin{aligned}
q\mathcal{P}([D,T],H) + p\mathcal{P}([E,U],H) &= q\mathcal{P}(T,H) + p\mathcal{P}(U,H) \\
&\geqslant \mathcal{P}(S,H) = \mathcal{P}(P,H) \\
\mathcal{N}([D,T],H) + \mathcal{N}([E,U],H) &= \mathcal{N}(T,H) + \mathcal{N}(U,H) \\
&\geqslant \mathcal{N}(S,H) = \mathcal{N}(P,H).
\end{aligned}$$

— The other cases are similar. □

Proposition 8 follows from the probabilistic strip lemma. This is a *simulation result*: if P and R are maximal and they have the same root, then P can simulate R (and vice-versa).

Proposition 8. For every maximal probabilistic computation P and for every finite probabilistic computation R such that P and R have the same root, there is a finite sub-computation Q of P such that for every $C \in \mathsf{NF}$, $\mathcal{P}(Q,C) \geqslant \mathcal{P}(R,C)$ and $\mathcal{N}(Q,C) \geqslant \mathcal{N}(R,C)$. Moreover, $\mathcal{P}(Q) \geqslant \mathcal{P}(R)$ and $\mathcal{N}(Q) \geqslant \mathcal{N}(R)$.

Proof. Given any probabilistic computation S, its $\mathscr{K}$-degree n_S is the number of consecutive commutative rules you find descending S, starting at the root. By Lemma 1 (Section 5.12.2), this is a good definition. The proof goes by induction on $(\mathsf{W}(R), n_R)$, ordered lexicographically:

— If $\mathsf{W}(R) = 0$, then R is just $[D]$ for some configuration D. Then, $Q = R$ and all the required conditions hold.

— If $\mathsf{W}(R) > 0$, then we distinguish three cases, depending on the shape of P:

– If $P = [D,S]$, E is the root of S and $D \to_{\mathscr{N}} E$, then, by Lemma 4, there is a probabilistic computation T with root E such that $\mathsf{W}(T) < \mathsf{W}(R)$ and $\mathcal{P}(T,C) \geqslant \mathcal{P}(R,C)$ for every $C \in \mathsf{NF}$. By the inductive hypothesis applied to S and T, there is a sub-probabilistic computation U of S such that $\mathcal{P}(U,C) \geqslant \mathcal{P}(T,C)$ and $\mathcal{N}(U,C) \geqslant \mathcal{N}(T,C)$ for every $C \in \mathsf{NF}$. Now, consider the probabilistic computation $[D,U]$. This is clearly a sub-probabilistic computation of P. Moreover, for every $C \in \mathsf{NF}$:

$$\begin{aligned} \mathcal{P}([D,U],C) &= \mathcal{P}(U,C) \\ &\geqslant \mathcal{P}(T,C) \geqslant \mathcal{P}(R,C) \\ \mathcal{N}([D,U],C) &= \mathcal{N}(U,C) \\ &\geqslant \mathcal{N}(T,C) \geqslant \mathcal{N}(R,C). \end{aligned}$$

– If $P = [D,S]$, E is the root of S and $D \to_{\mathscr{K}} E$, then, by Lemma 4, there is a probabilistic computation T with root E such that $\mathsf{W}(T) \leqslant \mathsf{W}(R)$ and $\mathcal{P}(T,C) \geqslant \mathcal{P}(R,C)$ for every $C \in \mathsf{NF}$. Now, observe that we can apply the inductive hypothesis to S and T, because $\mathsf{W}(T) \leqslant \mathsf{W}(R)$ and $n_S < n_P$. So, there is a sub-probabilistic computation U of S such that $\mathcal{P}(U,C) \geqslant \mathcal{P}(T,C)$ and $\mathcal{N}(U,C) \geqslant \mathcal{N}(T,C)$ for every $C \in \mathsf{NF}$. Now, consider the probabilistic computation $[D,U]$. This is clearly a sub-probabilistic computation of P. Moreover, for every $C \in \mathsf{NF}$:

$$\begin{aligned} \mathcal{P}([D,U],C) &= \mathcal{P}(U,C) \\ &\geqslant \mathcal{P}(T,C) \geqslant \mathcal{P}(R,C) \\ \mathcal{N}([D,U],C) &= \mathcal{N}(U,C) \\ &\geqslant \mathcal{N}(T,C) \geqslant \mathcal{N}(R,C). \end{aligned}$$

– $P = [(p,q,D), S_1, S_2]$, E_1 is the root of S_1 and E_2 is the root of S_2, then, by Lemma 4, there are probabilistic computations T_1 and T_2 with root E_1 and E_2 such that $\mathsf{W}(T_1), \mathsf{W}(T_2) < \mathsf{W}(R)$ and $p\mathcal{P}(T_1,C) + q\mathcal{P}(T_2,C) \geqslant \mathcal{P}(R,C)$ for every $C \in \mathsf{NF}$. By the inductive hypothesis applied to S_1 and T_1 (to S_2 and T_2, respectively), there is a sub-probabilistic computation U_1 of S_1 (a sub-probabilistic computation U_2 of S_2, respectively) such that $\mathcal{P}(U_1,C) \geqslant \mathcal{P}(T_1,C)$ for every

$C \in \mathsf{NF}$ ($\mathcal{P}(U_2, C) \geqslant \mathcal{P}(T_2, C)$ for every $C \in \mathsf{NF}$, respectively). Now, consider the probabilistic computation $[(p, q, D), U_1, U_2]$. This is clearly a sub-probabilistic computation of P. Moreover, for every $C \in \mathsf{NF}$:

$$\begin{aligned}\mathcal{P}([(p, q, D, U_1, U_2], C) &= p\mathcal{P}(U_1, C) + q\mathcal{P}(U_2, C) \\ &\geqslant p\mathcal{P}(T_1, C) + q\mathcal{P}(T_2, C) \geqslant \mathcal{P}(R, C).\end{aligned}$$

This concludes the proof. □

We are now able to prove our main result:

Theorem 10 (strong confluence). For every maximal probabilistic computations P and R such that P and R have the same root, and for every $C \in \mathsf{NF}$, $\mathcal{P}(P, C) = \mathcal{P}(R, C)$ and $\mathcal{N}(P, C) = \mathcal{N}(R, C)$. Moreover, $\mathcal{P}(P) = \mathcal{P}(R)$ and $\mathcal{N}(P) = \mathcal{N}(R)$.

Proof. Let $C \in \mathsf{NF}$ be any configuration in normal form. By definition:

$$\mathcal{P}(P, C) = \sup_{Q \sqsubset P}\{\mathcal{P}(Q, C)\} \qquad \mathcal{P}(R, C) = \sup_{S \sqsubset R}\{\mathcal{P}(S, C)\}.$$

Now, consider the two sets $A = \{\mathcal{P}(Q, C)\}_{Q \sqsubset P}$ and $B = \{\mathcal{P}(S, C)\}_{S \sqsubset R}$. We claim the two sets have the same upper bounds. Indeed, if $x \in \mathbb{R}$ is an upper bound on A and $S \sqsubset R$, by Proposition 8 there is $Q \sqsubset P$ such that $\mathcal{P}(Q, C) \geqslant \mathcal{P}(S, C)$, and so $x \geqslant \mathcal{P}(S, C)$. As a consequence, x is an upper bound on B. Symmetrically, if x is an upper bound on B, it is an upper bound on A. Since A and B have the same upper bounds, they have the same least upper bound, and $\mathcal{P}(P, C) = \mathcal{P}(R, C)$. The other claims can be proved in exactly the same way. This concludes the proof. □

6.5. *Computing with mixed states*

In the previous section we proved the strong confluence theorem for probabilistic computations. This result can be reformulated on *mixed states*, informally introduced in Section 6.1.

More formally, a mixed state is a finite support distribution of configurations:

Definition 22 (mixed state). A mixed state is a function $\mathcal{M} : \mathbf{Conf} \to \mathbb{R}_{[0,1]}$ such that there is a finite set $S \subseteq \mathbf{Conf}$ with $\mathcal{M}(C) = 0$ except when $C \in S$ and, moreover, $\sum_{C \in S} \mathcal{M}(C) = 1$. **Mix** is the set of mixed states.

A mixed state $\mathcal{M}$ will be denoted with the linear notation $\{p_1 : C_1, \ldots, p_k : C_k\}$ or as $\{p_i : C_i\}_{1 \leqslant i \leqslant k}$, where p_i is the probability $\mathcal{M}(C_i)$ associated with the configuration C_i, and where $\{C_1, \ldots, C_k\}$ is the set S from the above definition.

Remember the discussion in Section 6.1: the reduction between mixed states, as a *deterministic* reduction between distributions, permits us to avoid the probabilistic behaviour of computations, and retrace results proved in the previous section in a standard, deterministic rewriting system.

The reduction $\to_\alpha^p$ between configurations can be extended to mixed states in the following way:

Definition 23 (mixed-reduction). The reduction relation $\Longmapsto$ between mixed states is defined in the following way: $\{p_1 : C_1, \ldots, p_m : C_m\} \Longmapsto \mathcal{M}$ *iff* there exist m mixed states $\mathcal{M}_1 = \{q_1^i : D_1^i\}_{1 \leqslant i \in n_1}, \ldots, \mathcal{M}_m = \{q_m^i : D_m^i\}_{1 \leqslant i \leqslant n_m}$ such that:

1. For every $i \in [1, m]$, it holds that $1 \leqslant n_i \leqslant 2$;
2. If $n_i = 1$, then either C_i is in normal form and $C_i = D_i^1$ or $C_i \to_{n,\mathscr{M}} D_i^1$;
3. If $n_i = 2$, then $C_i \to^p_{\mathsf{meas}_r} D_i^1$, $C_i \to^q_{\mathsf{meas}_r} D_i^2$, $p, q \in \mathbb{R}_{[0,1]}$, and $q_i^1 = p, q_k^2 = q$;
4. $\forall D \in \mathbf{Conf}.\ \mathcal{M}(D) = \sum_{i=1}^m p_i \cdot \mathcal{M}_i(D)$.

Given the reduction relation $\Longmapsto$, the corresponding notion of computation (that we call *mixed computation*, in order to emphasize that mixed states play the role of configurations) is completely standard.

We again need to deal with quantitative properties of computations:

Definition 24 (probability of observing a configuration in a mixed state). Given a mixed state $\mathcal{M}$ and a configuration $C \in \mathsf{NF}$, the *probability* of observing C in $\mathcal{M}$ is defined as $\mathcal{M}(C)$ and is denoted as $\mathcal{P}(\mathcal{M}, C)$.

Observe that if $\mathcal{M} \Longmapsto \mathscr{M}'$ and $C \in \mathsf{NF}$, then $\mathcal{P}(\mathcal{M}, C) \leqslant \mathcal{P}(\mathscr{M}', C)$. If $\{\mathcal{M}_i\}_{i<\varphi}$ is a mixed computation, then

$$\sup_{i<\varphi} \mathcal{P}(\mathcal{M}_i, C)$$

always exists, and is denoted as $\mathcal{P}(\{\mathcal{M}_i\}_{i<\varphi}, C)$.

Note 6.1. A maximal mixed computation is always infinite. Indeed, if $\mathcal{M} = \{p_i : C_i\}_{1 \leqslant i \leqslant n}$ and for every $i \in [1, n], C_i \in \mathsf{NF}$, then $\mathcal{M} \Longmapsto \mathcal{M}$.

Proposition 9 is an essential tool for the proof of Theorem 11. Note that equivalence between different probabilistic computations with the same root (Theorem 10) is used.

Proposition 9. Let $\{\mathcal{M}_i\}_{i<\omega}$ be a maximal mixed computation and let $C_1, \ldots, C_n$ be the configurations on which $\mathcal{M}_0$ evaluates to a positive real. Then there are maximal probabilistic computations $P_1, \ldots, P_n$ with roots $C_1, \ldots, C_n$ such that $\sup_{j<\varphi} \mathcal{M}_j(D) = \sum_{i=1}^n (\mathcal{M}_0(C_i)\mathcal{P}(P_i, D))$ for every D.

Proof. Let $\{\mathcal{M}_i\}_{i<\omega}$ be a maximal mixed computation. Observe that $\mathcal{M}_0 \Longmapsto^m \mathcal{M}_m$ for every $m \in \mathbb{N}$. For every $m \in \mathbb{N}$ let $\mathcal{M}_m$ be

$$\{p_1^m : C_1^m, \ldots, p_{n_m}^m : C_{n_m}^m\}.$$

For every m, we can build maximal probabilistic computations $P_1^m, \ldots, P_{n_m}^m$, generatively: assuming $P_1^{m+1}, \ldots, P_{n_{m+1}}^{m+1}$ are the probabilistic computations corresponding to $\{\mathcal{M}_i\}_{m+1 \leqslant i < \omega}$, they can be extended (and possibly merged) into some maximal probabilistic computations $P_1^m, \ldots, P_{n_m}^m$ corresponding to $\{\mathcal{M}_i\}_{m \leqslant i < \omega}$. But we can even define for every $m, k \in \mathbb{N}$ with $m \leqslant k$, some finite probabilistic computations $Q_1^{m,k}, \ldots, Q_{n_m}^{m,k}$ with root $C_1, \ldots, C_{n_m}$ and such

that, for every m, k,

$$\begin{aligned} Q_i^{m,k} &\sqsubseteq P_i^m \\ \mathcal{M}_k(D) &= \sum_{i=1}^{n_m} \left(\mathcal{M}_m(C_i)\mathcal{P}(Q_i^{m,k}, D) \right). \end{aligned}$$

This proceeds by induction on $k - m$. We can easily prove that for every $S \sqsubseteq P_i^m$ there is k such that $S \sqsubseteq Q_i^{m,k}$: this is an induction on S (which is a finite probabilistic computation). But now, for every $D \in \mathsf{NF}$,

$$\begin{aligned} \sup_{j<\omega} \mathcal{M}_j(D) &= \sup_{j<\omega} \sum_{i=1}^{n_0} \left(\mathcal{M}_0(C_i)\mathcal{P}(Q_i^{0,j}, D) \right) \\ &= \sum_{i=1}^{n_0} \left(\mathcal{M}_0(C_i) \sup_{j<\omega} \mathcal{P}(Q_i^{0,j}, D) \right) \\ &= \sum_{i=1}^{n_0} \left(\mathcal{M}_0(C_i)\mathcal{P}(P_i^0, D) \right). \end{aligned}$$

This concludes the proof. □

Theorem 11 says that maximal mixed computations with the same root are equivalent in the following sense: normal form configurations have the same probability of being observed.

Theorem 11. For any two maximal mixed computations $\{\mathcal{M}_i\}_{i<\omega}$ and $\{\mathscr{M}'_i\}_{i<\omega}$ such that $\mathcal{M}_0 = \mathscr{M}'_0$, the following condition holds: for every $C \in \mathsf{NF}$, $\mathcal{P}(\{\mathcal{M}_i\}_{i<\omega}, C) = \mathcal{P}(\{\mathscr{M}'_i\}_{i<\omega}, C)$.

Proof. A consequence of Proposition 9. Sketch: suppose, by way of contradiction and without loosing generality, that $\mathcal{P}(\{\mathcal{M}_i\}_{i<\varphi}, C) < \mathcal{P}(\{\mathscr{M}'_i\}_{i<\psi}, C)$. This means there exists a natural number $n \leqslant \psi$ such that $\mathcal{P}(\{\mathcal{M}_i\}_{i<\varphi}, C) < \mathcal{P}(\{\mathscr{M}'_i\}_{i<n}, C)$. The idea is that you can apply Proposition 8 to the finite probabilistic computations induced by $\{\mathscr{M}'_i\}_{i<n}$ and to the corresponding maximal probabilistic computations induced by $\{\mathcal{M}_i\}_{i<\varphi}$, obtaining a contradiction. □

7. Intermezzo: quantum higher-order languages

In this section, we will recall some contributions in the definition of quantum languages. We will principally focus on functional calculi. In Section 7.4 we will spend few words about other important approaches such as the measurement calculus (Danos *et al.* 2007), providing some bibliographic references for the interested reader.

It is mandatory to say that the first defined quantum λ-calculus, here dubbed λ_q, has been introduced in van Tonder (2004). λ_q is a higher-order, quantum functional language without an explicit measurement operator, whose terms' generation is based on Wadler's formulation of linear logic (as for Q). λ_q includes explicit constants for quantum bits

and quantum gates; it aims to model interesting quantum features such as reversibility (a notion of *history*, is proposed) and superposition between terms is admitted: this means that (differently from Q) the syntax allows to write expressions like $M + N$, where M and N are lambda terms and the symbol $+$ represents a formal sum. As a matter of fact, terms in superposition can only differ in occurrences containing the constants 0 and 1 van Tonder (2004). Consequently, in λ_q *two terms M and N in superposition differ only for qubits values.* An informal proof about the equivalence of λ_q with quantum Turing machines is sketched.

λ_q is a corner stone in the brief history of quantum functional calculi and van Tonder's proposal is a valid platform for several investigations (Arrighi *et al.* 2008; Dal Lago and Zorzi 2013, 2014).

Nevertheless, here we will focus on other calculi, in our opinion more useful to the non-expert reader in order to understand *how* a quantum calculus can be build and used as a programming paradigm. Where appropriate, we will make some comparisons among them and with respect to our Q and Q^*.

7.1. *The languages* λ_{sv} *and* QML,

We propose an overview on two distinct *foundational* proposals: the quantum λ-calculus with classical control defined in Selinger and Valiron (2006), here dubbed λ_{sv}, and the functional language QML (Grattage 2006, 2011).

7.2. *Selinger's* λ_{sv}

After the first attempt to define a quantum higher-order language in two unpublished papers (Maymin 1996, 1997), Selinger rigorously defined a first-order quantum functional language (Selinger 2004). Subsequently the author, in joint work with Valiron (Selinger and Valiron 2006), defined a quantum λ-calculus with classical control, inspired by the QRAM architecture (Lanzagorta and Uhlmann 2009).

The main goal of Selinger and Valiron's work is to give the basis of a *typed* quantum functional language. The idea is to define a language where only data (the qubits) are superposed, and where programs live in a standard classical world. This means, in particular, that it is not necessary to have objects such as λ-terms in superposition (i.e. to have in the syntax expressions of the kind $\sum_i \alpha_i M_i$, where α_i is an amplitude and M_i is a lambda term). The approach is well condensed by the slogan: *'quantum data-classical control'* (see also Section 5). The proposed calculus, here dubbed λ_{sv}, is based on a call-by-value λ-calculus enriched with constants for unitary transformations and an explicit measurement operator which allows the program to observe the value of 1-qubit.

Reductions are defined between *program states*, whose definition is similar to the one of configuration (Section 5.10). Because of the presence of measurement, the authors provide the operational semantics introducing a suitable probabilistic reduction system, in order to define a probabilistic call-by-value procedure for the evaluation.

The type system of λ_{sv} is based on linear logic in order to control over the linearity of the system (linearity is extended to higher types), by distinguishing between duplicable

and non-duplicable resources. Selinger and Valiron develop the following type syntax: $A, B ::= \alpha | X | !A | A \multimap B | \top | A \otimes B$ where α ranges over a set of type constants, X ranges over a countable set of type variables and $\top$ is the linear unit type.

The authors restrict the system to be *affine*, i.e. *contraction structural rule is not allowed*. This choice is justified by the no-cloning property of quantum states. The type system avoids run time errors and is also equipped with subtyping rules, which provide a more refined control of the resources. The calculus enjoys some good properties such as subject reduction and progress, and a strong notion of *safety*. The authors also develop a new interesting quantum type inference algorithm, based on the idea that a linear (quantum) type can be viewed as a decoration of an intuitionistic one.

The following is an example of λ_{sv} well-typed term. The `nw` constant behaves like our new constant.

Example 10. The quantum circuit EPR (see Example 6, Section 5.11.1) is encoded in λ_{sv} as $EPR = \lambda x.CNOT\langle H(\mathsf{nw}(0)), \mathsf{nw}(0)\rangle$ and has type $!(\top \to (\mathsf{qbit} \otimes \mathsf{qbit}))$ where qbit is the base type of qubits.

λ_{sv}'s reductions are performed in a call-by-value setting. This is a central characteristic of the calculus, which makes λ_{sv} different from Q (and principally from Q*, which has an explicit measurement operator), where no strategy is imposed. The following example shows that is essential to choose a precise strategy and that different strategies are not equivalent.

Let us consider, for examples, the following (well typed) term in λ_{sv}:

$$M \equiv (\lambda x.x \oplus x)(\mathsf{ms}(H(\mathtt{nw}(0))))$$

where $\oplus$ is (the encoding of) a XOR operator and ms is the λ_{sv} measurement operator.

Using a call-by-value strategy (dubbed $\to_{\mathsf{cbv}}$), if we start from the program state $[\varnothing, M]$ we obtain the following computation:
$[\varnothing, M] \to_{\mathsf{cbv}} [|0\rangle, (\lambda x.x \oplus x)(\mathsf{ms}(H(p)))] \to_{\mathsf{cbv}} [\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), (\lambda x.x \oplus x)(\mathsf{ms}(p))]$.

The measurement step $\mathsf{ms}(p)$ yields the equiprobable program states $[|0\rangle, (\lambda x.x \oplus x)(0)]$ and $[|1\rangle, (\lambda x.x \oplus x)(1)]$ which reduce, respectively, to $[|0\rangle, (0)]$ and $[|1\rangle, (0)]$. Thus the initial state $[\varnothing, M]$ evaluate to 0 with probability 1.

Using a call-by-name reduction strategy, the state $[\varnothing, M]$ reduces with the same probability $\frac{1}{4}$ to one of the following states: $[|01\rangle, 1]$, $[|01\rangle, 1]$, $[|00\rangle, 0]$ and $[|11\rangle, 0]$. Therefore, the output is 0 or 1 with equal probability $\frac{1}{2}$.

Moreover, as the authors observe, a mixed strategy can produce an hill-formed term. Starting from the same program state $[\varnothing, M]$, let us consider the following computation, where some steps of call-by-value are followed by call-by-name reductions (dubbed $\to_{\mathsf{cbn}}$):

$$[\varnothing, M] \to^*_{\mathsf{cbv}} \left[\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), (\lambda x.x \oplus x)(\mathsf{ms}(p))\right] \to_{\mathsf{cbn}} \left[\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), (\mathsf{ms}(p)) \oplus (\mathsf{ms}(p))\right] \ldots$$

The last state $[\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), (\mathsf{ms}(p)) \oplus (\mathsf{ms}(p))]$ is not a legal program state, since the qubit name p has been duplicated.

In Selinger and Valiron (2009), the authors provide a categorical semantics for λ_{sv}. As they declare in Selinger and Valiron (2009), the construction of a concrete model

for higher-order quantum computations is still a partially open, interesting problem. For some recent investigations on semantic models, see Gay and Makie (2009). See also Hashuo and Hoshino (2011), where the authors define a particle-style geometry of interaction (Abramsky *et al.* 2002; Haghverdi and Scott 2002) for λ_{sv}.

7.3. *Altenkirch et Grattage's QML*

Another noticeable contribution in the definition of quantum functional calculi is QML, a typed quantum language for finite quantum computations. QML has been developed in Altenkirch and Grattage (2005), Grattage (2006), Altenkirch *et al.* (2007) and Grattage (2011). Here, we mainly refer to Grattage (2006), Altenkirch *et al.* (2007) and Grattage (2011). QML addresses programming issues rather than foundational questions and permits to encode quantum algorithms in a simple way (more precisely, a single term is able to encode a quantum circuit, i.e. a single term captures an algorithm of type $Q^k \multimap Q^k$, where Q is the type of the qubits for a given $k \in \mathbb{N}$). The syntax allows to build expressions such as αt, where α is an amplitude and t is a term, or expressions like $t + u$ where the symbol '+' represents the superposition of terms. However, superposition is controlled by a restrictive notion of 'orthogonality' between terms, defined by means of a suitable notion of inner product between judgements (Altenkirch *et al.* 2007). Intuitively, $t \perp u$ holds when t and u are 'distinguishable in some way' (Grattage 2011): in other words, one can derive the judgement $\mathsf{true} \perp \mathsf{false}$ from $t \perp u$ by means of orthogonality rules. Orthogonality judgements are inferred automatically by static analysis of QML's terms (Grattage 2011).

Quantum superposition can be combined with the conditional construct. The syntax includes both the usual if then else and the quantum conditional if° then else ; the quantum conditional if° then else is only allowed when the values in the branches are orthogonal. As an example, the following term represents the encoding of the Hadamard gate in QML:

$$\mathsf{had}\ x = \mathsf{if}^{\circ} x\ \mathsf{then}\ \left(\frac{1}{\sqrt{2}}\mathsf{false} + \left(-\frac{1}{\sqrt{2}}\right)\mathsf{true}\right)\ \mathsf{else}\ \left(\frac{1}{\sqrt{2}}\mathsf{false} + \frac{1}{\sqrt{2}}\mathsf{true}\right).$$

Comparing it with λ_{sv}, QML shows a different approach to the (central) problem of copy and discard quantum data. In QML it is possible to write expressions like $\mathsf{let}\ x = (\mathsf{false} + \mathsf{true})\mathsf{in}\ \langle x, x\rangle$ which (apparently) violates no-cloning properties, since the variable x is duplicate in the let expression.

This expression does not actually clone quantum data (this is guaranteed by the formal semantics (Altenkirch *et al.* 2007)); on the contrary, it shares one copy of the quantum datum between different references.

This idea is captured by the formulation of the type system, where no restrictions are imposed on the use of the structural rule of contraction. QML's type system is based on strict (multiplicative) linear logic, i.e. linear logic with contraction, but *without implicit weakening*. This makes it different from λ_{sv} type system, which is affine (implicit contraction, which potentially introduce syntactical violation of the no-cloning property, is not allowed).

Example 11 (CNOT in QML). In the following, Q is the type of quantum bits and $\otimes$ is the linear logic tensor between linear types.

Let $qnot\ x = \mathsf{if}^{\circ} x\ \mathsf{then}\ \mathsf{false}\ \mathsf{else}\ \mathsf{true}$ be the encoding of not gate.
qnot has type $Q \multimap Q$. Let us define $cnot : Q \multimap Q \multimap Q \otimes Q$ as

$$cnot\ c\ x = \mathsf{if}^{\circ} c\ \mathsf{then}\ \langle \mathsf{true}, qnot\ x \rangle\ \mathsf{else}\ \langle \mathsf{false}, x \rangle.$$

In QML,, measurement is modelled by means of conditional operator. For more details, see Grattage (2006).

For QML, some semantics have been proposed. QML's operational and denotational semantics are developed in terms of quantum circuits and superoperators (Preskill 2006) respectively. An equational theory for the measurement-free fragment of QML has been proposed in Altenkirch *et al.* (2007). Moreover, Grattage defined a compiler (developed in Haskell) for the language (Grattage 2011). In an ongoing project, this perspective is currently developed and extended. For a complete overview on this topic, see AA.VV. (2013).

7.4. *Other approaches*

In Arrighi and Dowek (2008), the linear algebraic λ-calculus $\mathcal{L}$ is proposed. $\mathcal{L}$ is an untyped calculus, where linear combinations of lambda terms are admitted (as e.g. in the algebraic λ-calculus of Vaux (2006, 2009)) and linearity is enforced by means of suitable reduction rules. The authors claim the language suitable for a quite easy 'specialization' into a quantum functional language. Others references about this approach are (Arrighi *et al.* 2012a,b).

Starting from the seminal paper (Nielsen 2003), some measurement based calculi have been defined. In particular, the so-called *measurement calculus* (Danos *et al.* 2007) has been developed as an efficient rewriting system for measurement based quantum computation. In Danos *et al.* (2007), the authors defined a calculus of local equations for 1-qubit one-way quantum computing. Roughly speaking, the idea is that a computation is built out of three basic commands, *entanglement*, *measurement* and *local correction*. The authors define a suitable syntax for these primitives, which permits the description of patterns, i.e. sequences of basic commands with qubits as input–output. By pattern composition, it is possible to implement quantum gates and quantum protocols. Moreover, a standardization theorem, which has some important consequences, is stated and proved. Measurement calculus has subsequently been extended and further developed in several papers (see, for example, (Danos *et al.* 2009)).

8. Taming quantum complexity classes

One of the strong motivations behind quantum computing is computational complexity: as showed in Shor (1994, 1997), quantum computers seem to be able to efficiently solve classical hard problems, thanks to the potential speed up in the execution time induced by superposition phenomena.

Quantum complexity theory has been developed since the 90s: several quantum classes have been defined and several results have been proved. Since quantum models are very complex, it seems to be interesting to characterize quantum complexity classes by *machine-free* and *resource-independent* systems, following the so-called *implicit computational complexity* (ICC) approach. The ICC approach involves a number of different methodologies which include, among others, proof-theoretical methods (logical systems, types systems), and bounded versions of arithmetic. A trend in linear logic community is to design type system inspired to controlled-complexity versions of linear logic, such as bounded linear logic (Girard *et al.* 1992), light linear logic (Girard 1998) and soft linear logic (Lafont 2004). Some examples of ICC-oriented calculi are: (Baillot and Mogbil 2004), a lambda calculus, sound and complete for polynomial time computations; (Gaboardi *et al.* 2013), where a characterization of the complexity class PSPACE is given; (Baillot *et al.* 2011), in which optimal reduction is studied by means of elementary and light affine logic inspired type systems. See Baillot *et al.* (2009) and related bibliographic references for an overview on some recent ICC trends.

In this section, we will see some results about a significant fragment of Q, called SQ. SQ is a measurement-free polytime quantum λ-calculus inspired by soft linear logic. We will prove that the construction of lambda terms may be controlled in order to capture a class of terms for which the size and the normalization procedure are polynomially bounded. SQ has been developed in Dal Lago *et al.* (2010), and up to our knowledge it is the only quantum polytime calculus that has appeared in the literature.

8.1. *The polytime quantum λ-calculus SQ: syntax and well-forming rules*

SQ has the same syntax as Q; the only difference is that we restrict the constants representing unitary operators to the subclass $\mathcal{U}$ of *polytime computable operators* on $\ell^2\{0,1\}^n$ (see Definition 30). Then, for each operator $\mathbf{U} \in \mathcal{U}$, there exists a symbol U in the grammar which represents it. This restriction is justified by the observations made in Appendix A.

All assumptions made for Q still hold. SQ is a 'refinement' of Q: in particular, we have to control the use of resources, in order to manipulate the intrinsic complexity of the system. The well-forming rules of SQ are strong enough to guarantee polystep termination of computations (Section 8.4).

Note that we are still defining an untyped language: in fact, the structure of untyped terms is itself sufficient to enforce properties such as soundness and completeness w.r.t. polynomial time, even in the absence of types. The non-quantum fragment of SQ is very similar (essentially equivalent) to the language of terms of Baillot and Mogbil's soft λ-calculus (Baillot and Mogbil 2004).

In SQ, resources are strongly monitored. As a consequence, we need to refine the notions of environment Γ and judgement $\Gamma \vdash M$. There are three kinds of variables: classical (x), banged (!x) and sharped (# x). Intuitively a !-variable represents a syntactical object that has to occur zero or once in a term, whereas a #-variable can occur in the term as many times as we want but at least once. Judgements and environments are defined taking into account the way the variables are used:

— A *classical environment* is a (possibly empty) set (denoted by Δ, possibly indexed) of classical variables. With $!\Delta$ we denote the set $!x_1, \ldots, !x_n$ whenever Δ is $x_1, \ldots, x_n$. Analogously, with $\#\Delta$, we denote the environment $\#x_1, \ldots, \#x_n$ whenever Δ is $x_1, \ldots, x_n$. If Δ is empty, then $!\Delta$ and $\#\Delta$ are empty. Notice that if Δ is a nonempty classical environment, both $\#\Delta$ and $!\Delta$ are *not* classical environments.

— A *quantum environment* is a (possibly empty) set (denoted by Θ, possibly indexed) of quantum variables.

— A *linear environment* is a (possibly empty) set (denoted by Λ, possibly indexed) Δ, Θ of classical and quantum variables.

— A *non-contractible environment* is a (possibly empty) set (denoted by Ψ, possibly indexed) $\Lambda, !\Delta$ where each variable name occurs at most once.

— An *environment* (denoted by Γ, eventually indexed) is a (possibly empty) set $\Psi, \#\Delta$ where each variable name occurs at most once.

In all the above definitions, we are implicitly assuming that the same (quantum or classical) variable name cannot appear more than once in an environment, e.g. $x, !y, \#z$ is a correct environment, while $x, !x$ is not.

Judgements are defined in a standard way: a *judgment* is an expression $\Gamma \vdash M$, where Γ is an environment and M is a term.

Notation 6. If $\Gamma_1, \ldots, \Gamma_n$ are (not necessarily pairwise distinct) environments, $\Gamma_1 \cup \ldots \cup \Gamma_n$ denotes the environment obtained by means of the standard set-union of $\Gamma_1, \ldots, \Gamma_n$.

The role of the underlying environment in well-formed judgements can be explained as follows. If $\Gamma, x \vdash M$ is well formed, then x appears free *exactly* once in M and, moreover, the only free occurrence of x does not lie in the scope of any ! construct. On the other hand, if $\Gamma, \#x \vdash M$ is well formed, then x appears free *at least* once in M and every free occurrence of x does not lie in the scope of any ! construct. Finally, if $\Gamma, !x \vdash M$ is well formed, then x appears *at most* once in M.

We say that a judgement $\Gamma \vdash M$ is *well formed* (*notation:* $\triangleright \Gamma \vdash M$), if it is derivable by means of the *well-forming rules* in Figure 7. In the app and tens rules, we relies on the assumption that the environment in the conclusion is well formed (multiple occurrences of the same variable name are not allowed).

If d is a derivation of the well-formed judgement $\Gamma \vdash M$, we write $d \triangleright \Gamma \vdash M$. If $\Gamma \vdash M$ is *well formed* we say that *M is well formed with respect to the environment* Γ, or, simply, that *M is well formed*.

Notice that the full-promotion rules prom (reminiscent of Scott's rule for modal logic) forces the parallel introduction of the operator '!' on the left and on the right side of the sequent: this induces a controlled nesting of '!'. Notice also that in arrow introduction rules abs_3 and abs_4 the distinction between banged and sharped variables disappears.

As for Q, it is easy to prove that if a term M is well formed all its classical variables are bound.

$$\frac{}{!\Delta \vdash C}\ \mathsf{const} \qquad \frac{}{!\Delta, r \vdash r}\ \mathsf{qvar} \qquad \frac{}{!\Delta, x \vdash x}\ \mathsf{cvar}$$

$$\frac{}{!\Delta, \#x \vdash x}\ \mathsf{der}_1 \qquad \frac{}{!\Delta, !x \vdash x}\ \mathsf{der}_2$$

$$\frac{\Psi_1, \#\Delta_1 \vdash M_1 \quad \Psi_2, \#\Delta_2 \vdash M_2}{\Psi_1, \Psi_2, \#\Delta_1 \cup \#\Delta_2 \vdash M_1 M_2}\ \mathsf{app}$$

$$\frac{\Psi_1, \#\Delta_1 \vdash M_1 \qquad \cdots \qquad \Psi_k, \#\Delta_k \vdash M_k}{\Psi_1, \ldots, \Psi_k, \#\Delta_1 \cup \#\Delta_2 \cup \cdots \cup \#\Delta_k \vdash \langle M_1, \ldots, M_k \rangle}\ \mathsf{tens}$$

$$\frac{\Delta_1 \vdash M}{!\Delta_2, !\Delta_1 \vdash !M}\ \mathsf{prom} \qquad \frac{\Gamma \vdash M}{\Gamma \vdash \mathtt{new}(M)}\ \mathsf{new} \qquad \frac{\Gamma, x_1, \ldots, x_n \vdash M}{\Gamma \vdash \lambda\langle x_1, \ldots, x_n \rangle.M}\ \mathsf{abs}_1$$

$$\frac{\Gamma, x \vdash M}{\Gamma \vdash \lambda x.M}\ \mathsf{abs}_2 \qquad \frac{\Gamma, \#x \vdash M}{\Gamma \vdash \lambda !x.M}\ \mathsf{abs}_3 \qquad \frac{\Gamma, !x \vdash M}{\Gamma \vdash \lambda !x.M}\ \mathsf{abs}_4$$

Fig. 7. Well-forming rules.

8.2. *Computations and operational properties of* SQ

The notions of configurations and reductions are exactly the same as Q. Even if the well-forming rules of SQ are different from those of Q, the well-formed terms of SQ are also well formed with respect to Q.

Therefore we adopt for SQ the same reduction rules as Q, given in Figure 3 (Section 5.10).

Good operational properties hold for SQ. They partially come from equivalent results on Q with some significant differences.

— SQ *enjoys the subject reduction property*, and the subject reduction theorem has to been reformulated in the following way:

Theorem 12 (subject reduction for SQ). If $\triangleright\Lambda, !\Delta_1, \#\Delta_2 \vdash M_1$ and $[\mathcal{Q}_1, \mathcal{QV}_1, M_1] \to [\mathcal{Q}_2, \mathcal{QV}_2, M_2]$ then there are environments Δ_3, Δ_4, such that $\Delta_1 = \Delta_3, \Delta_4$ and $\triangleright\Lambda, !\Delta_3, \#\Delta_4 \cup \#\Delta_2, \mathcal{QV}_2 - \mathcal{QV}_1 \vdash M_2$. Moreover, $\mathcal{QV}_2 - \mathcal{QV}_1 = \mathbf{Q}(M_2) - \mathbf{Q}(M_1)$.

We omit the long proof, but it is mandatory to say that, differently from the proof of Theorem 2, it is quite complex. The main point is the presence of the patterns $!x$ and $\#x$, which are both mapped into the term $\lambda !x$ by means of the $\to_!$ and $\to_\#$ rules. See Zorzi (2009) and Dal Lago *et al.* (2010) for full details.

— SQ *enjoys confluence* exactly as for Q. Confluence can be proved directly (as done for Q), but it can also be derived as a consequence of Q's confluence (Theorem 3), exploiting the fact that the set of SQ terms is a subset of Q terms.

— Finally, *standardization* also still holds: from each soft quantum computation an equivalent computation can be built, in which classical reductions are performed before new reductions and new reductions precede all purely quantum steps. The standardization result (Theorem 6, Section 5.13.1) is useful for the proof of soundness in Section 8.4.

Note 8.1. In order to simplify the treatment in Sections 8.3 and 8.4, we will consider reduction between terms rather than between configurations. If $[\mathcal{Q}, \mathcal{QV}, M] \rightarrow_\alpha [\mathcal{R}, \mathcal{RV}, N]$, then we will simply write $M \rightarrow_\alpha N$. This is sensible, since N only depends on M (and does not depend on $\mathcal{Q}$ or $\mathcal{QV}$).

8.3. *Encoding data structures and representing decision problems*

In this section, we define how to encode data structures in SQ and how to represent decision problems. For full details see Zorzi (2009) and Dal Lago *et al.* (2010).

We start with natural numbers. Unfortunately, we cannot use the same encoding adopted in Section 5, because now we have to control the duplication of resources.

Instead of using Scott's encodings (like in Q), we will use here a major variation on Church-style encoding: in particular, in SQ we will have several different lambda terms representing the same natural number.

Given a natural number $n \in \mathbb{N}$ and a term M, the class of *n-banged forms of* M is defined by induction on n:

— The only 0-banged form of M is M itself;

— If N is a n-banged form of M, any term $!L$ where $L \rightarrow_{\mathcal{N}}^{*} N$ is an $n+1$-banged form of M.

Let $!^n M$ denote $\underbrace{!(!\ldots(!}_{n \text{ times}} M)\ldots)$. Notice that $!^n M$ is an n-banged form of M for every $n \in \mathbb{N}$ and for every term M.

Given natural numbers $n, m \in \mathbb{N}$, a term M is said to *n-represent* the natural number m iff for every n-banged form L of N

$$ML \rightarrow_{\mathcal{N}}^{*} \lambda z.\underbrace{N(N(N(\ldots(N}_{m \text{ times}} z)\ldots))).$$

A term M is said to (n,k)-*represent* a function $f : \mathbb{N} \rightarrow \mathbb{N}$ iff for every natural number $m \in \mathbb{N}$, for every term N which 1-represents m, and for every n-banged form L of N

$$ML \rightarrow_{\mathcal{N}}^{*} P$$

where P k-represents $f(m)$.

For every natural number $m \in \mathbb{N}$, let $\lceil m \rceil$ be the term

$$\lambda !x.\lambda y.\underbrace{x(x(x(\ldots(x}_{m \text{ times}} y)\ldots))).$$

For every m, $\lceil m \rceil$ 1-represents the natural number m.

For every natural number $m \in \mathbb{N}$ and every positive natural number $n \in \mathbb{N}$, let $\lceil m \rceil^n$ be the term defined by induction on n:

$$\lceil m \rceil^0 = \lceil m \rceil;$$
$$\lceil m \rceil^{n+1} = \lambda !x.\lceil m \rceil^n x.$$

For every n, m, $\lceil m \rceil^n$ can be proved to $n+1$-represent the natural number m.

SQ can compute any polynomial, in a strong sense.

Proposition 10. For any polynomial with natural coefficients $p : \mathbb{N} \to \mathbb{N}$ of degree n, there is a term M_p that $(2n+1, 2n+1)$-represents p.

Proof. The proof exploits the fact that any polynomial can be written as a *Horner's polynomial*, which is either:

— The constant polynomial $x \mapsto k$, where $k \in \mathbb{N}$ does not depend on x.
— Or the polynomial $x \mapsto k + x \cdot p(x)$, where $k \in \mathbb{N}$ does not depend on x and $p : \mathbb{N} \to \mathbb{N}$ is itself a Horner's polynomial.

So, proving that the thesis holds for Horner's polynomials suffices. The proof is by induction, following the recursion schema. □

We are also interested in representing strings. Given any string $s = b_1 \dots b_n \in \Sigma^*$ (where Σ is a finite alphabet), the term $\lceil s \rceil^\Sigma$ is the following:

$$\lambda !x_{a_1}. \dots .\lambda !x_{a_m}.\lambda !y.\lambda z.yx_{b_1}(yx_{b_2}(yx_{b_3}(\dots(yx_{b_n}z)\dots))),$$

where $\Sigma = \{a_1, \dots, a_m\}$. Consider the term

$$\mathsf{strtonat}_\Sigma = \lambda x.\lambda !y.\lambda z.x \underbrace{!!(\lambda w.w)\dots!!(\lambda w.w)}_{m \text{ times}}!(\lambda !w.\lambda r.yr)z.$$

As can be easily shown, $\mathsf{strtonat}_\Sigma \lceil b_1 \dots b_n \rceil^\Sigma$ rewrites to a term 1-representing n:

$$\begin{aligned}
\mathsf{strtonat}_\Sigma \lceil b_1 \dots b_n \rceil^\Sigma !L &\to_{\mathcal{N}}^* \lambda z.\lceil b_1 \dots b_n \rceil^\Sigma \underbrace{!!(\lambda w.w)\dots!!(\lambda w.w)}_{m \text{ times}}!(\lambda !w.\lambda r.Lr)z \\
&\to_{\mathcal{N}}^* \lambda z.(\lambda !w.\lambda r.Lr)!(\lambda w.w)((\lambda !w.\lambda r.Lr)!(\lambda w.w)((\lambda !w.\lambda r.Lr)!(\lambda w.w) \\
&\qquad (\dots((\lambda !w.\lambda r.Lr)!(\lambda w.w)z)\dots))) \\
&\to_{\mathcal{N}}^* (\lambda z.\underbrace{L(L(L(\dots(L}_{n \text{ times}}z)\dots))).
\end{aligned}$$

Lists are the obvious generalization of strings where an infinite amount of constructors are needed. Given a sequence $M_1, \dots, M_n$ of terms (with no free variable in common), we can build a term $[M_1, \dots, M_n]$ encoding the sequence as follows, by induction on n:

$$[] = \lambda !x.\lambda !y.y;$$
$$[M, M_1 \dots, M_n] = \lambda !x.\lambda !y.xM[M_1, \dots, M_n].$$

This way we can construct and destruct lists in a principled way: the terms cons and sel can be built as follows:

$$\mathsf{cons} = \lambda z.\lambda w.\lambda !x.\lambda !y.xzw;$$
$$\mathsf{sel} = \lambda x.\lambda y.\lambda z.xyz.$$

They behave as follows on lists:

$$\mathsf{cons} M[M_1,\ldots,M_n] \to^*_{\mathscr{N}} [M,M_1,\ldots,M_n];$$
$$\mathsf{sel}[]!N!L \to^*_{\mathscr{N}} L;$$
$$\mathsf{sel}[M,M_1,\ldots,M_n]!N!L \to^*_{\mathscr{N}} NM[M_1,\ldots,M_n].$$

By exploiting cons and sel, we can build more advanced constructors and destructors: for every natural number n there are the terms append_n and $\mathsf{extract}_n$ behaving as follows:

$$\mathsf{append}_n[N_1,\ldots,N_m]M_1,\ldots,M_n \to^*_{\mathscr{N}} [M_1,\ldots,M_n,N_1,\ldots,N_m];$$
$$\forall m \leqslant n.\ \mathsf{extract}_n M[N_1,\ldots,N_m] \to^*_{\mathscr{N}} M[]N_m N_{m-1}\ldots N_1;$$
$$\forall m \geqslant n.\ \mathsf{extract}_n M[N_1,\ldots N_m] \to^*_{\mathscr{N}} M[N_{n+1}\ldots N_m]N_n N_{n-1}\ldots N_1.$$

append_n terms can be built by induction on n:

$$\mathsf{append}_0 = \lambda x.x;$$
$$\mathsf{append}_{n+1} = \lambda x.\lambda y_1.\ldots.\lambda y_{n+1}.\mathsf{cons}\ y_{n+1}(\mathsf{append}_n x y_1\ldots y_n).$$

Similarly, $\mathsf{extract}_n$ terms can be built inductively:

$$\mathsf{extract}_0 = \lambda x.\lambda y.xy;$$
$$\mathsf{extract}_{n+1} = \lambda x.\lambda y.(\mathsf{sel} y!(\lambda z.\lambda w.\lambda v.\mathsf{extract}_n vwz)!(\lambda z.[]))x.$$

Indeed

$$\forall m.\ \mathsf{extract}_0 M[N_1,\ldots N_m] \to^*_{\mathscr{N}} M[N_1,\ldots,N_m];$$
$$\forall n.\ \mathsf{extract}_{n+1} M[] \to^*_{\mathscr{N}} M[];$$
$$\forall m < n.\ \mathsf{extract}_{n+1} M[N,N_1\ldots N_m] \to^*_{\mathscr{N}} \mathsf{extract}_n M[N_1,\ldots,N_m]N \to^*_{\mathscr{N}} M[]N_m\ldots N_1 N;$$
$$\forall m \geqslant n.\mathsf{extract}_{n+1} M[N,N_1\ldots N_m] \to^*_{\mathscr{N}} \mathsf{extract}_n M[N_1,\ldots,N_m]N \to^*_{\mathscr{N}} M[N_{n+1}\ldots N_m]N_n\ldots N_1 N.$$

We now need to understand how to represent subsets of $\{0,1\}^*$ in SQ.

A term M *outputs the binary string* $s \in \{0,1\}^*$ *with probability* p *on input* N iff there is $m \geqslant |s|$ such that

$$[1,\varnothing,MN] \xrightarrow{*} [\mathcal{Q},\{q_1,\ldots,q_m\},[q_1,\ldots,q_m]]$$

and the probability of observing s when projecting $\mathcal{Q}$ into the subspace $\mathcal{H}(\{q_{|s|+1},\ldots,q_m\})$ is precisely p.

Given $n \in \mathbb{N}$, two binary strings $s, r \in \{0,1\}^k$ and a probability $p \in [0,1]$, a term M is said to (n,s,r,p)-*decide a language* $L \subseteq \{0,1\}^*$ iff the following two conditions hold:

— M outputs the binary string s with probability at least p on input $!^n\lceil t \rceil^{\{0,1\}}$ whenever $t \in L$;
— M outputs the binary string r with probability at least p on input $!^n\lceil t \rceil^{\{0,1\}}$ whenever $t \notin L$.

With the same hypothesis, M is said to be *error-free (with respect to* (n,s,r)*)* iff for every binary string t, the following two conditions hold:

— If M outputs s with positive probability on input $!^n\lceil t \rceil^{\{0,1\}}$, then M outputs r with null probability on the same input;
— Dually, if M outputs r with positive probability on input $!^n\lceil t \rceil^{\{0,1\}}$, then M outputs s with null probability on the same input.

We are now able to define some classes of languages decided by terms in SQ:

Definition 25 (classes ESQ, BSQ and ZSQ). Three classes of languages in the alphabet $\{0,1\}$ are defined below:

1. ESQ is the class of languages which can be $(n,s,r,1)$-decided by a term M of SQ;
2. BSQ is the class of languages which can be (n,s,r,p)-decided by a term M of SQ, where $p > \frac{1}{2}$;
3. ZSQ is the class of languages which can be (n,s,r,p)-decided by an error-free (w.r.t. (n,s,r)) term M of SQ, where $p > \frac{1}{2}$;

The purpose of the following two sections is precisely proving that ESQ, BSQ and ZSQ coincide with the quantum polytime complexity classes EQP, BQP ad ZQP (defined in Appendix A.4), respectively.

8.4. *Polytime soundness*

In this section and in the following one we will show that SQ is sound and complete with respect to *polynomial time quantum Turing machines* as defined by Bernstein and Vazirani (1997). In particular, we want to prove the 'perfect' equivalence of SQ with polynomial quantum Turing machines (Appendices A.3 and A.4).

To prove the soundness theorem is equivalent to showing that all *decision problems* which can be represented in SQ belong to one of the three classes of quantum polytime. We will only give a sketch of the proof, which is long but quite standard for ICC calculi, with the exception that the proof of the main theorem (Theorem 14) exploits a quantum property of SQ computations, i.e. standardization.

Note 8.2. In the following we assume that *all the involved terms are well formed.*

Some definitions are in order now. It will be useful to deal with the number of free occurrences of classical variables in a term. For each term M and for each classical

variable x, we define the number of free occurrences $\mathsf{NFO}(x, M)$ of x in M by induction on M:

$$\begin{aligned}
\mathsf{NFO}(x, x) &= 1 \\
\mathsf{NFO}(x, y) = \mathsf{NFO}(x, r) = \mathsf{NFO}(x, C) &= \mathsf{NFO}(x, \lambda x.M) = \mathsf{NFO}(\lambda !x.M) = 0 \\
\mathsf{NFO}(x, !M) &= \mathsf{NFO}(x, \mathtt{new}(M)) = \mathsf{NFO}(x, M) \\
\mathsf{NFO}(x, \lambda y.M) &= \mathsf{NFO}(x, \lambda !y.M) = \mathsf{NFO}(x, M) \\
\mathsf{NFO}(x, \lambda\langle x_1, \ldots, x_n\rangle.M) &= \mathsf{NFO}(x, M) \\
\mathsf{NFO}(x, MN) &= \mathsf{NFO}(x, M) + \mathsf{NFO}(x, N) \\
\mathsf{NFO}(x, \langle M_1, \ldots, M_n\rangle) &= \sum_1^n \mathsf{NFO}(x, M_i).
\end{aligned}$$

The *size* of a term is defined in a standard way:

Definition 26 (size).

$$\begin{aligned}
|x| = |r| = |C| &= 1 \\
|!N| &= |N| + 1 \\
|\mathsf{new}(P)| &= |P| + 1 \\
|PQ| &= |P| + |Q| + 1 \\
|\langle M_1, \ldots, M_k\rangle| &= |M_1| + \ldots + |M_k| + 1 \\
|\lambda x.N| = |\lambda !x.N| = |\lambda\langle x_1, \ldots, x_k\rangle.N| &= |N| + 1
\end{aligned}$$

It is possible to prove that, for each term M and for each free variable x, the size $|M|$ is an upper bound for $\mathsf{NFO}(x, M)$. Moreover, the following results hold:

Lemma 5. If $M \xrightarrow{n}_{\mathscr{K}} N$, then (i) $|M| = |N|$; (ii) $n \leqslant |M|^2$.

Proof. See Dal Lago *et al.* (2010). □

We also need to define some weights on lambda terms, and successively we need to prove that these weights are 'controllable' during reductions.

Definition 27 (box-depth, duplicability-factor, weights).

1. the *box-depth* $\mathsf{B}(M)$ of M (the maximum number of nested !–terms in M) is defined as

$$\begin{aligned}
\mathsf{B}(x) = \mathsf{B}(r) = \mathsf{B}(C) &= 0 \\
\mathsf{B}(!N) &= \mathsf{B}(N) + 1 \\
\mathsf{B}(\mathsf{new}(N)) &= \mathsf{B}(N) \\
\mathsf{B}(PQ) &= \max\{\mathsf{B}(P), \mathsf{B}(Q)\} \\
\mathsf{B}(\langle M_1, \ldots, M_k\rangle) &= \max\{\mathsf{B}(M_1), \ldots, \mathsf{B}(M_k)\} \\
\mathsf{B}(\lambda x.N) = \mathsf{B}(\lambda !x.N) = \mathsf{B}(\lambda\langle x_1, \ldots, x_k\rangle.N) &= \mathsf{B}(N);
\end{aligned}$$

2. the *duplicability-factor* $\mathsf{D}(M)$ of M (an upper bound on number of occurrences of any variable bound by a λ) is defined as

$$\begin{aligned}
\mathsf{D}(x) = \mathsf{D}(r) = \mathsf{D}(C) &= 1\\
\mathsf{D}(!N) &= \mathsf{D}(N)\\
\mathsf{D}(\mathsf{new}(N)) &= \mathsf{D}(N)\\
\mathsf{D}(PQ) &= \max\{\mathsf{D}(P), \mathsf{D}(Q)\}\\
\mathsf{D}(\langle M_1, \ldots, M_k\rangle) &= \max\{\mathsf{D}(M_1), \ldots, \mathsf{D}(M_k)\}\\
\mathsf{D}(\lambda x.N) = \mathsf{D}(\lambda !x.N) &= \max\{\mathsf{D}(N), \mathsf{NFO}(x, N)\}\\
\mathsf{D}(\lambda\langle x_1, \ldots, x_k\rangle.N) &= \max\{\mathsf{D}(N), \mathsf{NFO}(x_1, N), \ldots, \mathsf{NFO}(x_k, N)\};
\end{aligned}$$

3. the *n-weight* $\mathsf{W}_n(M)$ of M (the weight of a term with respect to n) is defined as

$$\begin{aligned}
\mathsf{W}_n(x) = \mathsf{W}_n(r) = \mathsf{W}_n(C) &= 1\\
\mathsf{W}_n(!N) &= n \cdot \mathsf{W}_n(N) + 1\\
\mathsf{W}_n(\mathsf{new}N) &= \mathsf{W}_n(N) + 1\\
\mathsf{W}_n(PQ) &= \mathsf{W}_n(P) + \mathsf{W}_n(Q) + 1\\
\mathsf{W}_n(\langle M_1, \ldots, M_k\rangle) &= \mathsf{W}_n(M_1) + \ldots + \mathsf{W}_n(M_k) + 1\\
\mathsf{W}_n(\lambda x.N) = \mathsf{W}_n(\lambda !x.N) = \mathsf{W}_n(\lambda\langle x_1, \ldots, x_k\rangle.N) &= \mathsf{W}_n(N) + 1;
\end{aligned}$$

4. the *weight* of a term M is defined as $\mathsf{W}(M) = \mathsf{W}_{\mathsf{D}(M)}(M)$.

Let us spend some words about the (sketch of the) proof of soundness. Soundness is essentially based on three facts we will prove on $\mathsf{W}(\cdot)$:

— $\mathsf{W}(M)$ is an upper bound for $|M|$ (Lemma 8);
— $\mathsf{W}(M)$ is monotone nonincreasing with respect to reduction steps (Lemma 9);
— $\mathsf{W}(M)$ is bounded by a certain polynomial (Lemma 10).

To prove $\mathsf{W}(\cdot)$ bounds, we need some results on duplicability factor, size and free variables.

It is possible to prove, by induction, that the duplicability factor $\mathsf{D}(\cdot)$ is nonincreasing with respect to reduction. Moreover, given a term M, the size of M is an upper bound for the duplicability factor:

Lemma 6. For every term M, $\mathsf{D}(M) \leqslant |M|$.

Proof. By induction on M. See Dal Lago *et al.* (2010). □

Another essential step towards polytime soundness is the controlled growth of free variables during the reduction.

Lemma 7. If $P \to_{\mathscr{L}} Q$ then $\max\{\mathsf{NFO}(x, P), \mathsf{D}(P)\} \geqslant \mathsf{NFO}(x, Q)$.

Proof. The proof proceeds by induction on the derivation of $P \to_{\mathscr{L}} Q$. □

We are now able to prove $\mathsf{W}(M)$ is an upper bound of $|M|$.

Lemma 8. For every term M, $|M| \leqslant \mathsf{W}(M)$.

Proof. By induction on the term M. □

$\mathsf{W}(M)$ is strictly decreasing at any non-commutative reduction step and it is nonincreasing at any commutative reduction step:

Lemma 9.

1. If $M \to_{\mathscr{K}} N$, then $\mathsf{W}(M) \geqslant \mathsf{W}(N)$;
2. if $M \to_{\mathscr{N}} N$, then $\mathsf{W}(M) > \mathsf{W}(N)$.

Proof. (i) The result follows by definition. Inductive steps are performed by means of context closures by induction on the derivation of $\to_{\mathscr{L}}$. (ii) the proof is by cases on the last rule of the derivation $\to_{\mathscr{L}}$ and by means of ad hoc substitution lemmata. □

Finally, we have to prove that $\mathsf{W}(M)$ is bounded by a polynomial $p(|M|)$, where the exponent of p depends on $\mathsf{B}(M)$ (but not on $|M|$). This implies, by Lemma 5, that the number of reduction steps from M to its normal form is polynomially related to $\mathsf{W}(M)$.

Lemma 10. For every term M, $\mathsf{W}(M) \leqslant |M|^{\mathsf{B}(M)+1}$.

Proof. It is possible to prove by induction on M that for every term M and for all positive $n \in \mathbb{N}$, $\mathsf{W}_n(M) \leqslant |M| \cdot n^{\mathsf{B}(M)}$. From this fact and by Lemma 6: $\mathsf{W}(M) = \mathsf{W}_{\mathsf{D}(M)}(M) \leqslant |M| \cdot \mathsf{D}(M)^{\mathsf{B}(M)} \leqslant |M| \cdot |M|^{\mathsf{B}(M)} = |M|^{\mathsf{B}(M)+1}$. □

We have all the technical tools to prove another crucial lemma, which tells us that the size is guaranteed to be polynomially bounded during the reduction.

Lemma 11. If $M \xrightarrow{*} N$, then $|N| \leqslant |M|^{\mathsf{B}(M)+1}$.

Proof. By means of Lemmas 8–10: $|N| \leqslant \mathsf{W}(N) \leqslant \mathsf{W}(M) \leqslant |M|^{\mathsf{B}(M)+1}$. □

The last step towards soundness is the following theorem, which states that each reduction in SQ is polystep.

Theorem 13 (bounds). There is a family of unary polynomials $\{p_n\}_{n\in\mathbb{N}}$ such that for any term M, for any $m \in \mathbb{N}$, if $M \xrightarrow{m} N$ (M reduces to N in m steps) then $m \leqslant p_{\mathsf{B}(M)}(|M|)$ and $|N| \leqslant p_{\mathsf{B}(M)}(|M|)$.

Proof. The suitable polynomials are $p_n(x) = x^{3(n+1)} + 2x^{2(n+1)}$. We need some definitions. Let $\mathbf{K}$ be a finite sequence $M_0, \ldots, M_v$ such that $\forall i \in [1, v]$. $M_{i-1} \to_c M_i$. $f(\mathbf{K}) = M_0$, $l(\mathbf{K}) = M_v$ and $\#\mathbf{K}$ denote respectively the first element, the last element and the length of the reduction sequence $\mathbf{K}$. Let us define the weight of a sequence $\mathbf{K}$ as $\mathsf{W}(\mathbf{K}) = \mathsf{W}(f(\mathbf{K}))$. We write a computation in the form $M = M_0, \ldots, M_m = N$ as a sequence of blocks of commutative steps $\mathbf{K}_0, \ldots, \mathbf{K}_\alpha$ where $M_0 = f(\mathbf{K}_0)$ and $l(\mathbf{K}_{i-1}) \to_{\mathscr{N}} f(\mathbf{K}_i)$ for every $1 \leqslant i \leqslant \alpha$. Note that $\alpha \leqslant |M|^{\mathsf{B}(M)+1}$; indeed, $\mathsf{W}(\mathbf{K}_0) > \cdots > \mathsf{W}(\mathbf{K}_\alpha)$ and

$$\mathsf{W}(\mathbf{K}_0) = \mathsf{W}(f(\mathbf{K}_0)) = \mathsf{W}(M_0) \leqslant |M|^{\mathsf{B}(M)+1}.$$

For every $i \in [0, v]$

$$\#\mathbf{K}_i \leqslant |f(\mathbf{K}_i)|^2 \leqslant (\mathsf{W}(f(\mathbf{K}_i)))^2 \leqslant (\mathsf{W}(M_0))^2 \leqslant |M|^{2(\mathsf{B}(M)+1)}.$$

Finally

$$\begin{aligned} m &\leqslant \#\mathbf{K}_0 + \cdots + \#\mathbf{K}_\alpha + \alpha \\ &\leqslant \underbrace{|M|^{2(\mathsf{B}(M)+1)} + \cdots + |M|^{2(\mathsf{B}(M)+1)}}_{\alpha+1} + |M|^{\mathsf{B}(M)+1} \\ &\leqslant \underbrace{(|M|^{2(\mathsf{B}(M)+1)} + \cdots\cdots + |M|^{2(\mathsf{B}(M)+1)})}_{|M|^{\mathsf{B}(M)+1}+2} \\ &= |M|^{2(\mathsf{B}(M)+1)} \cdot (|M|^{\mathsf{B}(M)+1} + 2) = |M|^{3(\mathsf{B}(M)+1)} + 2|M|^{2(\mathsf{B}(M)+1)} \\ &= p_{\mathsf{B}(M)}(|M|). \end{aligned}$$

Moreover,

$$\begin{aligned} |N| = |f(\mathbf{K}_\alpha)| &\leqslant \mathsf{W}(f(\mathbf{K}_\alpha)) \leqslant \mathsf{W}(M_0) \leqslant |M|^{\mathsf{B}(M)+1} \\ &\leqslant p_{\mathsf{B}(M)}(|M|). \end{aligned}$$

This concludes the proof. □

Note that the strong normalization property of SQ configurations follows as an easy corollary from Theorem 13.

The following is our soundness theorem. In the proof, we will essentially use three ingredients: the standardization theorem, which permits to decompose computations in distinct phases; Theorem 13, which gives our polynomial bounds; and 'perfect' equivalence between the quantum Turing machine and finitely generated uniform quantum circuit families (Nishimura and Ozawa 2009) (see Appendix A.5).

Theorem 14 (polytime soundness). The following inclusions hold: $\mathsf{ESQ} \subseteq \mathsf{EQP}$, $\mathsf{BSQ} \subseteq \mathsf{BQP}$ and $\mathsf{ZSQ} \subseteq \mathsf{ZQP}$.

Proof. Let us consider the first inclusion. Suppose a language $\mathcal{L}$ is in ESQ. This implies that $\mathcal{L}$ can be $(n, s, r, 1)$-decided by a term M. By the standardization theorem, for every $t \in \{0,1\}^*$, there is a CNQ computation $\{C_i^t\}_{1 \leqslant i \leqslant n_t}$ starting at $[1, \varnothing, M!^n\lceil t \rceil^{\{0,1\}}]$. By Theorem 13, n_t is bounded by a polynomial on the length $|t|$ of t. Moreover, the size of any C_i^t (that is to say, the sum of the term in C_i^t and the number of quantum variables in the second component of C_i^t) is itself bounded by a polynomial on $|t|$. Since $\{C_i^t\}_{1 \leqslant i \leqslant n_t}$ is CNQ, any classical reduction step comes before any new-reduction step, which itself comes before any quantum reduction step. As a consequence, there is a polynomial time deterministic Turing machine which, for every t, computes one configuration in $\{C_i^t\}_{i \leqslant n_t}$ which only contains non-classical redexes (if any). But notice that a configuration only containing non-classical redexes is nothing but a concise abstract representation of a quantum circuit, fed with boolean inputs. Moreover, all the quantum circuits produced in this way are finitely generated, i.e. they can only contain the quantum gates (i.e. unitary operators) which appears in M, since $!^n\lceil t \rceil^{\{0,1\}}$ does not contain any unitary operators and reduction does not introduce new unitary operators in the underlying term.

Summing up, the first component $\mathcal{Q}$ of $C^t_{n_t}$ is simply an element of a Hilbert space $\mathcal{H}(\{q_1,\ldots,q_m\})$ (where $[q_1,\ldots,q_m]$ is the third component of $C^t_{n_t}$) obtained by evaluating a finitely generated quantum circuit whose size is polynomially bounded on $|t|$ and whose code can be effectively computed from t in polynomial time. By the results in Nishimura and Ozawa (2009), $L \in \mathsf{EQP}$. The other two inclusions can be handled in the same way. □

8.5. *Polytime completeness*

We prove here the converse of Theorem 14, i.e. that all problems which are computable in polynomial time by some quantum devices can be decided by an SQ term.

The proof of the completeness theorem will essentially be an encoding of polytime quantum Turing machines, although not in a direct way: we will exploit Yao's encoding of quantum Turing machines with finitely generated quantum circuit families (a sketch of the encoding can be found in Appendix A.5) and we will show that SQ is able to simulate Yao's construction. This way, we avoid dealing directly with both quantum Turing machines and classical Turing machines (the latter being an essential ingredient of the definition of a quantum circuit family).

We will recall the principal 'ingredients' of Yao's encoding. A computation of a polytime quantum Turing machine computing in $m = t(n)$ steps (where n is the length of the input) can be simulated by a quantum circuit $L_{t(n)}$. For each m, the circuit L_m is composed by m copies of a circuit K_m: each sub-circuit K_m encodes an instantaneous description of the state, a particular configuration assumed by the Turing machine during the computation. The sub-circuit K_m is built by composing further sub-circuits of kind G_m (a 'switch' circuit) and J_m, decomposable in some single-qubit circuit H. See Figure 8 in Appendix A.5.

The simulation must be uniform, i.e. there must be *a single* term M generating all the possible L_m where m varies over the natural numbers.

The following two propositions show how sub-circuits G_m and J_m can be generated uniformly by an SQ term.

Proposition 11. For every n, there is a term M^n_G which uniformly generates G_m, i.e. such that whenever L n-encodes the natural number m, $M^n_G L \to_c R^m_G$ where R^m_G encodes G_m.

Proof. Consider the following terms:

$$M^n_G = \lambda x.\lambda y.\mathsf{extract}_\eta(\lambda z.\lambda w_1.\ldots.\lambda w_\eta.\mathsf{append}_\eta w_1 \ldots w_\eta(N^n_G xz))y$$
$$N^n_G = \lambda x.x!^n(\lambda y.\lambda z.\mathsf{extract}_{\lambda+2}((L_G y)z))(\lambda y.y)$$
$$L_G = \lambda x.\lambda y.\lambda z_1.\ldots.\lambda z_{\lambda+2}.(\lambda\langle w,q\rangle.\mathsf{append}_{\lambda+2}(xy)z_1\ldots z_\lambda wq)(cnot\langle z_{\lambda+1}, z_{\lambda+2}\rangle)$$

In order to prove the correctness of the encoding, let us define P^m_G for every $m \in \mathbb{N}$ by induction on m as follows:

$$P^0_G = \lambda x.x$$
$$P^{m+1}_G = (\lambda y.\lambda z.(\mathsf{extract}_{\lambda+2}((L_G y)z)))P^m_G$$

Observe that if L n-encodes the natural number m, then
$N_G^n L \to_c L!^n(\lambda y.\lambda z.\mathsf{extract}_{\lambda+2}((L_G y)z))(\lambda y.y) \to_c \underbrace{P(P(P(\ldots(P}_{m \text{ times}}(\lambda x.x))\ldots))) = P_G^m$
where $P = (\lambda y.\lambda z.(\mathsf{extract}_{\lambda+2}((L_G y)z)))$.
By induction, it is possible to prove that for every $m \in \mathbb{N}$:

$$[\mathcal{Q}, \mathcal{QV}, P_G^m[q_1, \ldots, q_{m(\lambda+2)}, \ldots, q_h]] \xrightarrow{*} [\mathcal{R}, \mathcal{QV}, [q_1, \ldots, q_{m(\lambda+2)}, \ldots, q_h]]$$

where $\mathcal{R} = \mathbf{cnot}_{\langle\langle q_{\lambda+1}, q_{\lambda+2}\rangle\rangle}(\mathbf{cnot}_{\langle\langle q_{2\lambda+3}, q_{2\lambda+4}\rangle\rangle}(\ldots(\mathbf{cnot}_{\langle\langle q_{m(\lambda+2)-1}, q_{m(\lambda+2)}\rangle\rangle}(\mathcal{Q}))\ldots))$.
Now, if L n-encodes the natural number m, then

$$\begin{aligned} M_G^n L &\to_c \lambda y.\mathsf{extract}_\eta(\lambda z.\lambda w_1.\ldots.\lambda w_\eta.\mathsf{append}_\eta w_1 \ldots w_\eta(N_G^n L z))y \\ &\to_c \lambda y.\mathsf{extract}_\eta(\lambda z.\lambda w_1.\ldots.\lambda w_\eta.\mathsf{append}_\eta w_1 \ldots w_\eta(P_G^m z))y \end{aligned}$$

which has all the properties we require for R_G^m. This concludes the proof. □

Proposition 12. For every n, there is a term M_J^n which uniformly generates J_m, i.e. such that $M_J^n L \to_c R_J^m$ where R_J^m encodes J_m whenever L n-encodes the natural number m.

Proof. The encoding is

$$\begin{aligned} M_J^n &= \lambda x.x!^n(N_J)(\lambda y.y) \\ N_J &= \lambda x.\lambda y.\mathsf{extract}_{\eta+\lambda+2}(L_J x)y \\ L_J &= \lambda x.\lambda y.\lambda z_1.\ldots.\lambda z_\eta.\lambda w_1.\ldots.\lambda w_{\lambda+2}. \\ &\quad \mathsf{extract}_{\eta+2(\lambda+2)}(P_J w_1 \ldots w_{\lambda+2})(x(\mathsf{append}_\eta y z_1 \ldots z_\eta)) \\ P_J &= \lambda x_1.\ldots.\lambda x_{\lambda+2}.\lambda w.\lambda y_1.\ldots.\lambda y_\eta.\lambda z_1.\ldots.\lambda z_{2(\lambda+2)}.(\lambda\langle q_1.\ldots.\lambda q_{\eta+3(\lambda+2)}\rangle. \\ &\quad \mathsf{append}_{\eta+3(\lambda+2)} w q_1 \ldots q_{\eta+3(\lambda+2)})(H\langle y_1, \ldots, y_\eta, x_1, \ldots, x_{\lambda+2}, z_1, \ldots z_{2(\lambda+2)}\rangle) \end{aligned}$$

The proof of correctness is similar to the one for Proposition 11. □

We now need a formal definition of what is a faithful simulation of a quantum Turing machine by an SQ term.

Given a Hilbert's space $\mathcal{H}$, an element $\mathcal{Q}$ of $\mathcal{H}$ and a condition E defining a subspace of $\mathcal{Q}$, the probability of observing E when globally measuring $\mathcal{Q}$ is denoted as $\mathscr{P}_{\mathcal{Q}}(E)$. For example, if $\mathcal{H} = \mathcal{H}(Q \times \Sigma^\# \times \mathbb{Z})$ is the configuration space of a quantum Turing machine, E could be $state = q$, which means that the current state is $q \in Q$. As another example, if $\mathcal{H}$ is $\mathcal{H}(\mathcal{QV})$, E could be $q_1, \ldots, q_n = s$, which means that the value of the variables $q_1, \ldots, q_n$ is $s \in \{0,1\}^n$.

Given a quantum Turing machine $\mathcal{M} = (Q, \Sigma, \delta)$, we say that a term M *simulates* the machine $\mathcal{M}$ iff there is a natural number n and an injection $\rho : Q \to \{0,1\}^{\lceil \log_2 |Q| \rceil}$ such that for every string $s \in \Sigma^*$ it holds that if $\mathcal{C}$ is the final configuration of $\mathcal{M}$ on input s, then

$$[1, \varnothing, M!^n\lceil s \rceil^\Sigma] \xrightarrow{*} [\mathcal{Q}, \{q_1, \ldots, q_m\}, [q_1, \ldots, q_m]].$$

where for every $q \in Q$

$$\mathscr{P}_{\mathcal{C}}(state = q) = \mathscr{P}_{\mathcal{Q}}(q_1, \ldots, q_{\lceil \log_2 |Q| \rceil} = \rho(q)).$$

Finally, we can state and prove the main result of this section:

Theorem 15. For every polynomial time quantum Turing machine $\mathcal{M} = (Q, \Sigma, \delta)$ there is a term $M_{\mathcal{M}}$ such that $M_{\mathcal{M}}$ simulates the machine $\mathcal{M}$.

Proof. The theorem follows from Propositions 11, 12 and 10. More precisely, the term $M_{\mathcal{M}}$ has the form $\lambda !x.(M^{circ}_{\mathcal{M}} x)(M^{init}_{\mathcal{M}} x)$ where

— $M^{circ}_{\mathcal{M}}$ builds the Yao's circuit, given a string representing the input;
— $M^{init}_{\mathcal{M}}$ builds a list of quantum variables to be fed to the Yao's circuit, given a string representing the input.

Now, suppose $\mathcal{M}$ works in time $p : \mathbb{N} \to \mathbb{N}$, where p is a polynomial of degree k. For every term M and for every natural number $n \in \mathbb{N}$, we define $\{M\}_n$ by induction on n:

$$\{M\}_0 = M$$
$$\{M\}_{n+1} = \lambda !x.!(\{M\}_n x).$$

It is easy to prove that for every M, for every N, for every $n \in \mathbb{N}$ and for every n-banged form L of N, $\{M\}_n L \to^*_{\mathcal{N}} P$ where P is an n-banged form of MN. Now, $M^{circ}_{\mathcal{M}}$ has the following form

$$\lambda !x.(N^{circ}_{\mathcal{M}} x)(L^{circ}_{\mathcal{M}} x)$$

where

$$N^{circ}_{\mathcal{M}} = \lambda x.M_{2p+1}(\{\mathsf{strtonat}_\Sigma\}_{2k+1}(M^{2k+2}_{id} x))$$
$$L^{circ}_{\mathcal{M}} = \lambda x.(\{P^{circ}_{\mathcal{M}}\}_{2k+1} x)$$
$$P^{circ}_{\mathcal{M}} = \lambda !z.\lambda y.(M^{2k+1}_J(M_{2p+1}(\{\mathsf{strtonat}_\Sigma\}_{2k+1} z)))(M^{2k+1}_G(M_{2p+1}(\{\mathsf{strtonat}_\Sigma\}_{2k+1} z)))y$$

M^{2k+1}_G comes from Proposition 11, M^{2k+1}_J comes from Proposition 12 and M_{2p+1} comes from Proposition 10. Now, consider any string $s = b_1 \dots b_n \in \Sigma^*$. First of all:

$$\begin{aligned} N^{circ}_{\mathcal{M}} !^{4k+3}\lceil s \rceil^\Sigma &\to^*_{\mathcal{N}} M_{2p+1}(\{\mathsf{strtonat}_\Sigma\}_{2k+1}(M^{2k+2}_{id} !^{4k+3}\lceil s \rceil^\Sigma)) \\ &\to^*_{\mathcal{N}} M_{2p+1}(\{\mathsf{strtonat}_\Sigma\}_{2k+1} !^{2k+1}\lceil s \rceil^\Sigma) \\ &\to^*_{\mathcal{N}} M_{2p+1} N \end{aligned}$$

where N is a $2k+1$-banged form of $\mathsf{strtonat}_\Sigma\lceil s \rceil^\Sigma$, itself a term which 1-represents the natural number n. As a consequence:

$$M_{2p+1} N \to^*_{\mathcal{N}} L$$

where L $2k+1$-represents the natural number $2p(n)+1$. Now:

$$\begin{aligned} L^{circ}_{\mathcal{M}} !^{4k+2}\lceil s \rceil^\Sigma &\to^*_{\mathcal{N}} \{P^{circ}_{\mathcal{M}}\}_{2k+1} !^{4k+3}\lceil s \rceil^\Sigma \\ &\to^*_{\mathcal{N}} P \end{aligned}$$

where P is a $2k+1$-banged form of $P^{circ}_{\mathcal{M}} !^{2k+2}\lceil s \rceil^\Sigma$. So, we can conclude that $M^{circ}_{\mathcal{M}} !^{4k+4}\lceil s \rceil^\Sigma$ rewrites to a term representing the circuit $\mathbf{L}_n$. $M^{init}_{\mathcal{M}}$ can be built with similar techniques. □

Corollary 3 (polytime completeness). The following inclusions hold:
$\mathsf{EQP} \subseteq \mathsf{ESQ}$, $\mathsf{BQP} \subseteq \mathsf{BSQ}$ and $\mathsf{ZQP} \subseteq \mathsf{ZSQ}$.

From Theorem 14 and Corollary 3, $\mathsf{EQP} = \mathsf{ESQ}$, $\mathsf{BQP} = \mathsf{BSQ}$ and $\mathsf{ZQP} = \mathsf{ZSQ}$. In other words, there is a perfect correspondence between (polynomial time) quantum complexity classes and the classes of languages decidable by SQ terms.

9. Conclusions

In this paper, we introduced an approach to quantum functional calculi. In particular, we studied a family of untyped quantum lambda calculi with classical control and we addressed some foundational questions such as the expressive power (the equivalence with other computational models), the theoretical studies of infinite computations in presence of an explicit measurement operator and the characterization of quantum polynomial time complexity classes.

Other approaches are possible. In particular, when the full expressiveness of the language is not required, one may perform different choices moving, for example, to a typed calculus as Selinger and Valiron (2006) and Grattage (2006). Some examples of (typed) calculi without explicit separation of the quantum register can be found in Delbeque (2011) or in Dal Lago and Zorzi (2014). These calculi represent a good basis for semantical studies. In the former, a game-style semantics is given, lifting the notion of game to the quantum case. In the latter, an operational account in terms of wave style token-machine (Abramsky *et al.* 2002) is proposed. The study of the role of geometry of interaction in quantum computing appears a promising direction of investigation, and it represents for us an ongoing research project.

As a further task, we aim to define the calculus SQ^*, i.e. an intrinsically polytime quantum calculus with explicit measurement operator. We plan to start from a calculus in the style of Delbeque (2011) and Dal Lago and Zorzi (2013).

Acknowledgements

A heartfelt thanks to my co-authors Ugo Dal Lago and Andrea Masini. Thanks also to the anonymous referees for their useful suggestions. I wish to dedicate this paper to Franca.

Appendix A. Quantum Computational Models and Quantum Complexity Classes

In this section, we will give some basic notions about computational models for quantum computing and some related complexity classes. The main quantum models are quantum Turing machine (QTM) (Bernstein and Vazirani 1997; Nishimura and Ozawa 2009) and quantum circuit families (QCF) (Nishimura and Ozawa 2002, 2009; Yao 1993). Equivalence results between the two models have been proved (Nishimura and Ozawa 2009).

A.1. *On computable operators*

After the introduction of quantum computational models, it is mandatory to spend some words about the following tricky point.

If we want to define calculi for quantum computational functions and we want to formalize expressiveness equivalence between different quantum models, we need to restrict the class of unitary operators we consider.

The following definitions on computability are therefore important.

Definition 28 ((polytime) computable real numbers).

1. A real number $x \in \mathbb{R}$ is *computable* iff there is a deterministic Turing machine which on input 1^n computes a binary representation of an integer $m \in \mathbb{Z}$ such that $|m/2^n - x| \leqslant \frac{1}{2^n}$. Let $\tilde{\mathbb{R}}$ be the set of computable real numbers.
2. A real number $x \in \mathbb{R}$ is *polynomial time computable* iff there is a deterministic polytime Turing machine which on input 1^n computes a binary representation of an integer $m \in \mathbb{Z}$ such that $|m/2^n - x| \leqslant \frac{1}{2^n}$. Let $\mathbf{P}\tilde{\mathbb{R}}$ be the set of polynomial time real numbers.

Definition 29 ((polytime) computable complex numbers and vectors).

1. A complex number $z = x + iy$ is *computable* iff $x, y \in \tilde{\mathbb{R}}$. Let $\tilde{\mathbb{C}}$ be the set of computable complex numbers.
2. A complex number $z = x + iy$ is *polynomial time computable* iff $x, y \in \mathbf{P}\tilde{\mathbb{R}}$. Let $\mathbf{P}\tilde{\mathbb{C}}$ be the set of polynomial time computable complex numbers.
3. A normalized vector ϕ in any Hilbert space $\ell^2(\mathcal{S})$ is *computable* (*polynomial time computable*) if the range of ϕ (a function from $\mathcal{S}$ to complex numbers) is $\tilde{\mathbb{C}}$ ($\mathbf{P}\tilde{\mathbb{C}}$).

Now we can define the computable unitary operators:

Definition 30 ((polytime) computable operators). A unitary operator $\mathbf{U} : \ell^2(\mathcal{S}) \to \ell^2(\mathcal{S})$ is *computable* (*polynomial time computable*) if for every computable (*polynomial time computable*) normalized vector ϕ of $\ell^2(\mathcal{S})$, $\mathbf{U}(\phi)$ is computable (polynomial time computable).

It is significant to remark that the restriction to a smaller class of quantum gates is also forced by computability problems, as remarked in Kitaev *et al.* (2002). The authors say (remark 9.2, p. 90): *'the use of an arbitrary complete basis could lead to* pathologies'. In fact they prove that the gate

$$X \equiv \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$

where θ is a non-computable number, *'enables us to solve the halting problem!'*.

In the following section, we introduce QCF, the model we mostly deal with in this paper.

A.2. *Quantum circuit families*

Recall that an *n-qubit quantum gate* is a unitary operator $\mathbf{U} : \ell^2(\{0,1\}^n) \to \ell^2(\{0,1\}^n)$ (Sections 2, 3). Given two unit vectors $|\phi\rangle, |\psi\rangle \in \ell^2(\{0,1\}^n)$, if $U|\phi\rangle = |\psi\rangle$, we call $|\psi\rangle$ the output state for the input state $|\phi\rangle$.

A *$\mathcal{V}$-qubit gate* (where $\mathcal{V}$ is a set of names) is a unitary operator $\mathbf{G} : \mathcal{H}(\mathcal{V}) \to \mathcal{H}(\mathcal{V})$ (see again Section 2.3 for the definition of the Hilbert space $\mathcal{H}(\mathcal{V})$).

If $\mathcal{G}$ is a set of qubit gates, a *$\mathcal{V}$-circuit* $\mathbf{K}$ based on $\mathcal{G}$ is a sequence

$$\mathbf{U}_1, r_1^1, \ldots, r_{n_1}^1, \ldots, \mathbf{U}_m, r_1^m, \ldots, r_{n_m}^m$$

where, for every $1 \leqslant i \leqslant m$:

— $\mathbf{U}_i$ is an n_i-qubit gate in $\mathcal{G}$;
— $r_1^i, \ldots, r_{n_i}^i$ are distinct quantum variables (names of qubits) in $\mathcal{V}$.

The $\mathcal{V}$-gate *determined by* a $\mathcal{V}$-circuit

$$\mathbf{K} = \mathbf{U}_1, r_1^1, \ldots, r_{n_1}^1, \ldots, \mathbf{U}_m, r_1^m, \ldots, r_{n_m}^m$$

is the unitary operator

$$U_{\mathbf{K}} = (\mathbf{U}_m)_{\langle\langle r_1^m, \ldots, r_{n_m}^m \rangle\rangle} \circ \ldots \circ (\mathbf{U}_1)_{\langle\langle r_1^1, \ldots, r_{n_1}^1 \rangle\rangle}$$

(the notation $\mathbf{U}_{\langle\langle r_1, \ldots, r_k \rangle\rangle}$ has been introduced in Section 2.2).

Once we have fixed a class of operators, it is possible to have effective encoding of circuits as natural numbers and, as a consequence, effective enumeration of quantum circuits.

Definition 31 (quantum circuit family). Let $\mathcal{G}$ be a denumerable set of quantum gates and let $\{\mathbf{K}_i\}_{i \in \mathbb{N}}$ be an effective enumeration of quantum circuits. A *family of circuits* generated by $\mathcal{G}$ is a triple (f, g, h) where:

— $f : \mathbb{N} \to \mathbb{N}$;
— $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is such that $0 \leqslant g(n, m) \leqslant n + 1$ whenever $1 \leqslant m \leqslant f(n)$;
— $h : \mathbb{N} \to \mathbb{N}$ is such that for every $n \in \mathbb{N}$, $\mathbf{K}_{h(n)}$ is a $\{r_1, \ldots, r_{f(n)}\}$-circuit based on $\mathcal{G}$.

Any family of circuits (f, g, h) induces a function $\Phi_{f,g,h}$ (the *function induced by* (f, g, h)) which, given any finite sequence $c_1, \ldots, c_n$ in $\{0, 1\}^*$, returns an element of $\mathcal{H}(\{r_1, \ldots, r_{f(n)}\})$:

$$\Phi_{f,g,h}(c_1, \ldots, c_n) = U_{\mathbf{K}_{h(n)}}(|r_1 \mapsto c_{g(n,1)}, \ldots, r_{f(n)} \mapsto c_{g(n,f(n))}\rangle).$$

where c_0, c_{n+1} are assumed to be 0 and 1, respectively.

Notice that gave our definition taking into account an arbitrary set $\mathcal{G}$ of quantum gates and arbitrary functions f, g and h. The choice of such a set and functions is instead crucial, and it induces the definition of particular subclasses of QCF:

Definition 32 (subclasses of quantum circuit families).

i. Uniform QCF

Given a quantum circuit family $\mathcal{K} = (f, g, h)$, we say that $\mathcal{K}$ is *uniform* if the functions f, g, h are computable.

ii. Polynomial-size uniform QCF

Given a quantum circuit family $\mathcal{K} = (f, g, h)$, we say that $\mathcal{K}$ is *polynomial-size uniform* if the functions f, g, h are polytime.

iii. Finitely generated QCF

Given a set $\mathcal{G}$ of quantum gates, we say that a family of circuits (f, g, h) generated by $\mathcal{G}$ is *finitely generated* if $\mathcal{G}$ is a finite set.

By definition, any polytime uniformly generated QCF is finitely generated. Finitely generated uniform QCFs enjoy two main properties: a finitely generated uniform QCF is based on finite sets of elementary gates (differently from the uniform QCFs); the definition of a finitely generated uniform QCF is independent of the choice of the set $\mathcal{G}$ (differently from the polynomial time uniformly generated QCF). The class of finitely generated QCF is the computational model that we proved to be equivalent to the calculus Q in Section 5.

A.3. *Quantum Turing machines*

This is a brief *introduction* to QTMs. For an exhaustive treatment see Bernstein and Vazirani (1997), Hirvensalo (2004) and Nishimura and Ozawa (2009).

Let Σ be is a finite alphabet with a blank symbol $\square$ and let Q be a finite set of states. We distinguish in the set Q an initial state q_0 and a final state q_f (with $q_0 \neq q_f$). As for the classical case, the QTM is based on the reading/writing of the tape by a head.

Let us start with the definitions of *tape configuration* and *configuration*:

Definition 33 (tape configurations and configurations).

— The set of tape configurations is the set of functions $\Sigma^{\#} = \{t : \mathbb{Z} \to \Sigma | t(m) \neq \square \text{ only for a finite } m \in \mathbb{Z}\}$. Given $t \in \Sigma^{\#}$, a symbol $\sigma \in \Sigma$ and an integer $k \in \mathbb{Z}$, a new tape configuration t_k^{σ} will be

$$t_k^{\sigma}(m) = \begin{cases} \sigma & \text{if } m = k \\ t(m) & \text{if } m \neq k. \end{cases}$$

— We will call a *frame* the pair (Q, Σ). To each frame, we can associate the *configurations space* $\mathscr{C}(Q, \Sigma) = Q \times \Sigma^{\#} \times \mathbb{Z}$.
Each element $C = (q, t, k) \in \mathscr{C}(Q, \Sigma)$, where $k \in \Sigma$ is the *head position*, is called a *configuration*. Moreover, we define *a* final configuration as any configuration such that $q = q_f$.

It is possible to build a Hilbert space generated by the configurations space:

Definition 34 (quantum state space). The *quantum state space* is the Hilbert space $\ell^2(\mathscr{C}(Q, \Sigma))$ (see Section 2.1). In the following, we will denote it as $\mathcal{H}(Q, \Sigma)$.

We now define the *pre-QTM*, the *QTM*, and some related properties. In the following we will use the notation $[a, b]_{\mathbb{Z}}$ $(a, b \in \mathbb{Z})$ in order to represent the integers between a and b (with a and b included).

Definition 35 (prequantum Turing machine). A *pre-QTM* is a triple (Q, Σ, δ) where (Q, Σ) is a frame and δ is the quantum transition function $\delta : Q \times \Sigma \times Q \times \Sigma \times [-1, 1]_{\mathbb{Z}} \to \tilde{\mathbb{C}}$.

The transition function δ induces the so-called *time evolution operator*.

Definition 36 (time evolution operator). $U^{\delta}_{\mathscr{M}} : \mathcal{H}(Q,\Sigma) \to \mathcal{H}(Q,\Sigma)$, a linear operator defined as

$$U^{\delta}_{\mathscr{M}}|C\rangle = U^{\delta}_{\mathscr{M}}|q,t,k\rangle = \sum_{(p,\sigma,d)\in Q\times\Sigma\times[-1,1]_{\mathbb{Z}}} \delta(q,t(k),p,\sigma,d)\cdot|p,t^{\sigma}_{k},k+d\rangle.$$

The definition of QTMs requires computable amplitudes and a unitary time evolution operator:

Definition 37 (quantum Turing machine). QTM is a pre-QTM $\mathscr{M} = (Q,\Sigma,\delta)$ such that the time evolution operator $U^{\delta}_{\mathscr{M}}$ is unitary (i.e. ${U^{\delta}_{\mathscr{M}}}^{\dagger}U^{\delta}_{\mathscr{M}} = I = U^{\delta}_{\mathscr{M}}{U^{\delta}_{\mathscr{M}}}^{\dagger}$).

We can strengthen the definition above by limiting the transition amplitudes to the polynomial computable complex numbers $\mathbf{P}\tilde{\mathbb{C}}$ (Definition 29): this does not reduce the computational power of the QTM (Bernstein and Vazirani 1997; Nishimura and Ozawa 2002, 2009).

The following conditions for the unitarity of a time evolution operator are stated in Nishimura and Ozawa (2009):

Theorem 16. Given a pre-QTM, $\mathscr{M} = (Q,\Sigma,\delta)$, the time evolution operator $U^{\delta}_{\mathscr{M}}$ is unitary if and only if the function δ satisfies the following conditions:

— for each $(q,\tau)\in Q\times\Sigma$,

$$\sum_{(p,\sigma,d)\in Q\times\Sigma\times[-1,1]_{\mathbb{Z}}} |\delta(q,\tau,p,\sigma,d)|^2 = 1$$

— for each $(q,\tau),(q',\tau')\in Q\times\Sigma$ with $(q,\tau)\neq(q',\tau')$

$$\sum_{(p,\sigma,d)\in Q\times\Sigma\times[-1,1]_{\mathbb{Z}}} \delta(q',\tau',p,\sigma,d)^{*}\delta(q,\tau,p,\sigma,d) = 0$$

— for each $(q,\tau,\sigma),(q',\tau',\sigma')\in Q\times\Sigma\times\Sigma$

$$\sum_{(p,d)\in Q\times[-1,1]_{\mathbb{Z}}} \delta(q',\tau',p,\sigma',d-1)^{*}\delta(q,\tau,p,\sigma,d) = 0$$

— for each $(q,\tau,\sigma),(q',\tau',\sigma')\in Q\times\Sigma\times\Sigma$

$$\sum_{p\in Q} \delta(q',\tau',p,\sigma',-1)^{*}\delta(q,\tau,p,\sigma,1) = 0.$$

Note A.1. In addition to the unitary property, we also require that the time evolution operator $U^{\delta}_{\mathscr{M}}$ must be (efficiently) computable.

This is not all about QTMs. A QTM needs some input/output conventions and moreover some careful definitions about the final result, since a QTM halts as a

superposition of the tapes' final contents. At the end of the computation, the superposition of configurations can be observed, obtaining final results or another superposition of configurations (this is the case of a partial measurement). For the sake of completeness we recall the following definition (Bernstein and Vazirani 1997):

Definition 38 (observation of a QTM). When QTM $\mathscr{M}$ in superposition $\phi = \sum_i \alpha_i c_i$ is *observed* or *measured*, each configuration c_i is seen with probability $|\alpha_i|^2$. We may also perform a partial measurement, say only on the first cell of the tape. In this case, suppose that the first cell may contain the values 0 or 1, and suppose the superposition was $\phi = \sum_i \alpha 0_i c0_i + \sum_i \alpha_1 c1_i$, where the $c0_i$ are those configurations that have a 0 in the first cell, and $c1_i$ are those configurations that have a 1 in the first cell. If a value b, $b = 0, 1$ is observed, the new superposition is $\frac{1}{Pr[b]} \sum_i \alpha b_i c b_i$, where $Pr[b] = \sum_i |\alpha b_i|^2$.

For further details about these crucial discussions see the original paper (Bernstein and Vazirani 1997).

In the following section, we will introduce quantum polytime complexity classes and the notions of *polytime QTM* and *acceptance of a language by a QTM* (for decision problems).

A.4. *Quantum polynomial time complexity classes*

Quantum complexity theory is a big field of investigation. A number of important results about quantum complexity classes and quantum computational problems have been proved, contributing to shed light also on the relationships with the classical theory. Some significative examples can be found in Watrous *et al.* (2009) and Watrous *et al.* (2011) (in the latter, the equality between QIP, the quantum analogue of the classical complexity class IP and PSPACE has been proved).

In this section and in this paper, we will focus our attention only on (the definition of) quantum polytime complexity classes, extensively used in Section 8.

As for the classical case, quantum complexity classes are defined on a computational model, but in this setting the definitions are more delicate.

Let us consider the class of QTMs à la Bernstein and Vazirani (1997): each computation evolves as a superposition in a space of configurations (Definition 34), and each classical computation in the superposition can evolve independently. So, the result of a quantum computation is obtained, at the end, with a measurement of the several superpositional results: as a consequence, it is irremediably probabilistic. This induces the definition of three distinct quantum polytime classes (EQP, BQP, ZQP), since different constraints can be imposed on success or error.

Firstly, we need to introduce the polynomial time QTM:

Definition 39 (polynomial time QTM). We say that a QTM $\mathscr{M}$ *halts with running time* T on input x if when $\mathscr{M}$ is run with input x, at time T the superposition contains only final configurations, and at any time $T_i < T$ the superposition contains no final configurations.

A *polynomial time* QTM $\mathscr{M}$ is a QTM which for every input x halts in time T with T polynomial in the length of x.

Note A.2. In this article, we refer to the complexity of the so-called *decision problems*. A decision problem is a function $Q : \{0,1\}^* \to \{0,1\}$ and, very informally, it performs a question on which the QTM has to give an answer of kind yes/no.

The following is the notion of *acceptance* for QTMs:

Definition 40 (acceptance of languages by a QTM). We say that a QTM $\mathscr{M}$ *accepts a language* $\mathcal{L}$ *with probability* p, if $\mathscr{M}$ accepts with probability at least p every string $x \in \mathcal{L}$, and rejects with probability at least p every string $x \notin \mathcal{L}$.

EQP is the error-free (or exact) quantum polynomial time complexity class. This means that the success probability is 1 on all input instances.

Definition 41 (quantum complexity class EQP). The class EQP is the set of the languages $\mathcal{L}$ accepted by polynomial QTM $\mathscr{M}$ with probability 1.

BQP is the bounded-error quantum polynomial time complexity class where the success probability is imposed to be strictly greater that $2/3$ on all input instances.

Definition 42 (quantum complexity class BQP). The class BQP is the set of the languages $\mathcal{L}$ accepted by polynomial QTM $\mathscr{M}$ with probability $2/3$.

The class ZQP is the zero-error extension of the class BQP. In fact the QTM never gives a wrong answer, but in each case with probability $1/3$ gives a 'don't-know' answer.

Definition 43 (quantum complexity class ZQP). The class ZQP is the set of the languages $\mathcal{L}$ accepted by polynomial QTM $\mathscr{M}$ such that, for every string x:

— if $x \in \mathcal{L}$, then $\mathscr{M}$ accepts x with probability $p > 2/3$ and rejects with probability $p = 0$;
— if $x \notin \mathcal{L}$, then $\mathscr{M}$ rejects x with probability $p > 2/3$ and accepts with probability $p = 0$.

The inclusions $\mathsf{EQP} \subseteq \mathsf{ZQP} \subseteq \mathsf{BQP}$ obviously hold. Moreover, $\mathsf{P} \subseteq \mathsf{BPP} \subseteq \mathsf{BQP} \subseteq \mathsf{PSPACE}$, where BPP is the class of the (bounded error) probabilistic polynomial time (Arora and Barak 2009).

The separation between BPP and BQP is still an open problem. In Bernstein and Vazirani (1997), the authors exhibit a problem, relative to an oracle, which requires polynomial time on a QTM, but requires superpolynomial time on a bounded-error probabilistic Turing machine. A similar, stronger result is given in Simon (1994), where the author gives the construction of an oracle problem that requires polynomial time on a QTM and exponential time on a classical Turing machine. Unfortunately, nowadays we have not got any strong separation theorem between BPP and BQP. This absence of strict separation is not an exception in computational complexity and it does not impair the convincing difference between the two models. A 'flavour' of separation comes from algorithms: quantum algorithms can exploit phenomena such as superposition and interference that a probabilistic model cannot emulate (see Section 4).

Notice that if we are able to prove the separation between the classes of probabilistic and quantum polytime we obtain the separation between P and PSPACE as a by-product.

A.5. *Equivalence between quantum Turing machines and quantum circuit families*

The equivalence between (a subclass of) QCF and polytime QTMs has been shown in Nishimura and Ozawa (2009).

This equivalence is called 'perfect', since it means a full correspondence between all the three polytime quantum complexity classes BQP, EQP, ZQP (Definitions 41–43) and their counterparts defined on QCF (Nishimura and Ozawa 2002).

The equivalence is stated and proved taking into account the class of *finitely generated* QCF (Definition 32): more precisely, the families have to be based on a finite subset of the set $\mathcal{G}_u = \{\Lambda_1(N), R(\theta), P(\theta') | \theta, \theta' \in \mathbf{P}\tilde{\mathbb{C}} \cap [0, 2\pi]\}$ (where $\Lambda_1(N)$ is a controlled-not gate, $R(\theta)$ is a rotation gate by angle θ and $P(\theta')$ is a phase shift gate by angle θ').

Notice that all definitions must be based on the notions of computable numbers and computable operators. In other words, the entries of the matrices representing the quantum gates in the universal set are required to be polynomial time computable numbers (as pointed out in Shor (1997)) and the amplitudes of the polynomial time QTMs have to be in the set $\mathbf{P}\tilde{\mathbb{C}}$ (Definitions 29 and 30).

We state the main results as in Nishimura and Ozawa (2009).

Theorem 17. Polynomial time QTM with amplitudes from $\mathbf{P}\tilde{\mathbb{C}}$ and finitely generated uniform QCF are perfectly equivalent: any polynomial time QTM with amplitudes from $\mathbf{P}\tilde{\mathbb{C}}$ can be exactly simulated by a finitely generated QCF, and vice-versa.

The following corollary comes as an immediate consequence of the previous theorem:

Corollary 4. The class of languages recognized with certainty (respectively with zero-error and bounded-error) by finitely generated QCFs coincide with the corresponding complexity class EQP (respectively ZQP and BQP) for polynomial time QTMs.

Nishimura and Ozawa's proof is largely based in the encoding of QTM into QCF proposed in Yao (1993).

We recall here a sketch of Yao's encoding, since it is used in Section 8.5.

Suppose we work with finite alphabets including a special symbol, called *blank* and denoted with $\square$. Moreover, each alphabet comes equipped with a function $\sigma : \Sigma \to \{0,1\}^{\lceil \lg_2(|\Sigma|) \rceil}$. Σ^ω is the set of infinite strings on the alphabet Σ, i.e. elements of Σ^ω are functions from $\mathbb{Z}$ to Σ. $\Sigma^\#$ is a subset of Σ^ω containing strings which are different from $\square$ in finitely many positions.

Consider a polytime QTM (Definition 39) $\mathcal{M} = (Q, \Sigma, \delta)$ working in time bounded by a polynomial $t : \mathbb{N} \to \mathbb{N}$. The computation of $\mathcal{M}$ on input of length n can be simulated by a quantum circuit $L_{t(n)}$ built as follows:

— for each m, L_m has $\eta + k(\lambda + 2)$ inputs (and outputs), where $\eta = \lceil \log_2 |Q| \rceil$, $k = 2m + 1$ and $\lambda = \lceil \log_2 |\Sigma| \rceil$. The first η qubits correspond to a binary encoding q of a state in Q. The other inputs correspond to a sequence $\sigma_1 s_1, \ldots, \sigma_k s_k$ of binary strings, where each σ_i (with $|\sigma_i| = \lambda$) corresponds to the value of a cell of $\mathcal{M}$, while each s_i (with $|s_i| = 2$) encodes a value from $\{0, 1, 2, 3\}$ controlling the simulation.
— L_m is built up by composing m copies of a circuit K_m, which is depicted in Figure 8 and has $\eta + k(\lambda + 2)$ inputs (and outputs) itself.

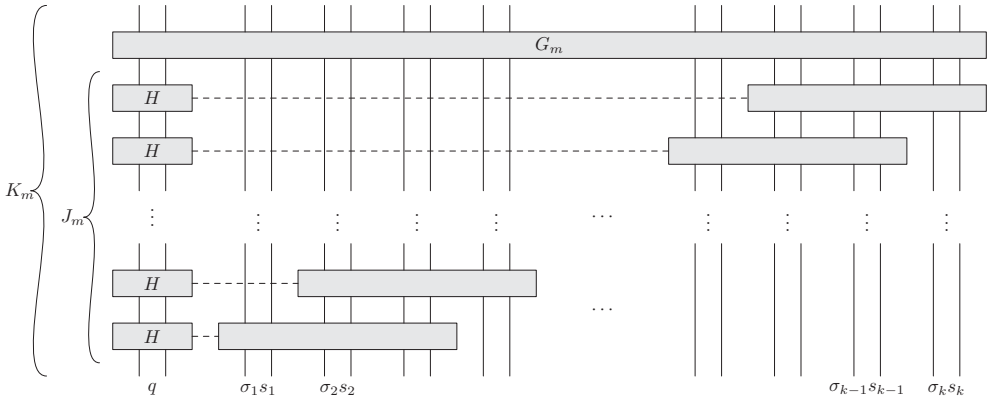


Fig. 8. The quantum circuit computing one step of the simulation.

— K_m is built up by composing G_m with J_m. G_m does nothing but switching the inputs corresponding to each s_i from 1 to 2 and vice-versa.
— J_m can be decomposed into $k-3$ instances of a single circuit H with $\eta + 3(\lambda + 2)$ inputs, acting on different qubits as shown in Figure 8. Notice that H can be assumed to be computable (in the sense of Definition 28), because $\mathcal{M}$ can be assumed to have amplitudes in $\mathbf{P}\tilde{\mathbb{C}}$ (Nishimura and Ozawa 2009).

Theorem 18 (Yao 1993). The circuit family $\{L_m\}_{m\in N}$ simulates the QTM $\mathcal{M}$.

See the original articles Yao (1993) and Nishimura and Ozawa (2009) for a full explanation of the correspondence results.

Appendix B. On Measurement Operators

We give here the detailed proofs of some properties of the destructive measurement maps defined in Section 2.4.

Destructive measurements (Definition 4) have to satisfy a completeness condition:

Proposition 13 (completeness condition). Let $r \in \mathcal{QV}$ and $\mathcal{Q} \in \mathcal{H}(\mathcal{QV})$. Then, $\mathsf{m}^{\dagger}_{r,0}\mathsf{m}_{r,0} + \mathsf{m}^{\dagger}_{r,1}\mathsf{m}_{r,1} = Id_{\mathcal{H}(\mathcal{QV})}$.

Proof. In order to prove the proposition we will use the following general property of inner product spaces: let $\mathcal{H}$ be an inner product space and let $A : \mathcal{H} \to \mathcal{H}$ be a linear map. If for each $x, y \in \mathcal{H}$, $\langle Ax, y\rangle = \langle x, y\rangle$ then A is the identity map (such a property is an immediate consequence of the Riesz representation theorem, see e.g. Roman (2008)). Let $\mathcal{Q}, \mathcal{R} \in \mathcal{H}(\mathcal{QV})$. If $\{b_i\}_{i\in[1,2^n]}$ is the computational basis of $\mathcal{H}(\mathcal{QV} - \{r\})$, then:

$$\mathcal{Q} = \sum_{i=1}^{2^n} \alpha_i |r \mapsto 0\rangle \otimes b_i + \sum_{i=1}^{2^n} \beta_i |r \mapsto 1\rangle \otimes b_i$$

$$\mathcal{R} = \sum_{i=1}^{2^n} \gamma_i |r \mapsto 0\rangle \otimes b_i + \sum_{i=1}^{2^n} \delta_i |r \mapsto 1\rangle \otimes b_i.$$

We have

$$\begin{aligned}\langle(\mathsf{m}^\dagger_{r,0}\mathsf{m}_{r,0}+\mathsf{m}^\dagger_{r,1}\mathsf{m}_{r,1})(\mathcal{Q}),\mathcal{R}\rangle &= \langle\mathsf{m}^\dagger_{r,0}\mathsf{m}_{r,0}(\mathcal{Q}),\mathcal{R}\rangle+\langle\mathsf{m}^\dagger_{r,1}\mathsf{m}_{r,1}(\mathcal{Q}),\mathcal{R}\rangle\\ &= \langle\mathsf{m}_{r,0}(\mathcal{Q}),\mathsf{m}_{r,0}(\mathcal{R})\rangle+\langle\mathsf{m}_{r,1}(\mathcal{Q}),\mathsf{m}_{r,1}(\mathcal{R})\rangle\\ &= \langle\sum_{i=1}^{2^n}\alpha_i b_i,\sum_{i=1}^{2^n}\gamma_i b_i\rangle+\langle\sum_{i=1}^{2^n}\beta_i b_i,\sum_{i=1}^{2^n}\delta_i b_i\rangle\\ &= \sum_{i=0}^{2^n}\alpha_i\gamma_i+\sum_{i=0}^{2^n}\beta_i\delta_i\\ &= \langle\mathcal{Q},\mathcal{R}\rangle.\end{aligned}$$

This concludes the proof. □

For $c\in\{0,1\}$, the measurement operators $\mathsf{m}_{r,c}$ also enjoy the following properties:

Proposition 14. Let $\mathcal{Q}\in\mathcal{H}(\mathcal{QV})$. Then:

1. $\mathsf{m}_{r,c}(\mathcal{Q}\otimes|q\mapsto d\rangle)=(\mathsf{m}_{r,c}(\mathcal{Q}))\otimes|q\mapsto d\rangle$ if $r\in\mathcal{QV}$ and $q\notin\mathcal{QV}$;
2. $\langle\mathcal{Q}\otimes|s\mapsto d\rangle|\mathsf{m}^\dagger_{r,c}\mathsf{m}_{r,c}|\mathcal{Q}\otimes|s\mapsto d\rangle\rangle=\langle\mathcal{Q},\mathsf{m}^\dagger_{r,c}\mathsf{m}_{r,c}|\mathcal{Q}\rangle$; if $r\in\mathcal{QV}$ and $r\neq s$;
3. $\mathsf{m}_{q,e}(\mathsf{m}_{r,d}(\mathcal{Q}))=\mathsf{m}_{r,d}(\mathsf{m}_{q,e}(\mathcal{Q}))$; if $r,q\in\mathcal{QV}$.

Proof.

1. Given the computational basis $\{b_i\}_{i\in[1,2^n]}$ of $\mathcal{H}(\mathcal{QV}-\{r\})$, we have that

$$\mathcal{Q}\otimes|q\mapsto d\rangle=\sum_{i=1}^{2^n}\alpha_i|r\mapsto 0\rangle\otimes b_i\otimes|q\mapsto d\rangle+\sum_{i=1}^{2^n}\beta_i|r\mapsto 1\rangle\otimes b_i|r\mapsto d\rangle$$

and therefore

$$\begin{aligned}\mathsf{m}_{r,0}(\mathcal{Q}\otimes|q\mapsto d\rangle) &= \sum_{i=1}^{2^n}\alpha_i(b_i\otimes|r\mapsto d\rangle)\\ &= \left(\sum_{i=1}^{2^n}\alpha_i b_i\right)\otimes|q\mapsto d\rangle\\ &= (\mathsf{m}_{r,c}(\mathcal{Q}))\otimes|q\mapsto d\rangle.\end{aligned}$$

In the same way we prove the equality for $\mathsf{m}_{r,1}$.

2. Just observe that

$$\begin{aligned}\langle\mathcal{Q}\otimes|s\mapsto d\rangle|\mathsf{m}^\dagger_{r,c}\mathsf{m}_{r,c}|\mathcal{Q}\otimes|s\mapsto d\rangle\rangle &= \langle\mathcal{Q}\otimes|s\mapsto d\rangle,\mathsf{m}^\dagger_{r,c}(\mathsf{m}_{r,c}(\mathcal{Q}\otimes|s\mapsto d\rangle))\rangle\\ &= \langle\mathsf{m}_{r,c}(\mathcal{Q}\otimes|s\mapsto d\rangle),\mathsf{m}_{r,c}(\mathcal{Q}\otimes|s\mapsto d\rangle)\rangle\\ &= \langle\mathsf{m}_{r,c}(\mathcal{Q}),\mathsf{m}_{r,c}(\mathcal{Q})\rangle\\ &= \langle\mathcal{Q},\mathsf{m}^\dagger_{r,c}\mathsf{m}_{r,c}\mathcal{Q}\rangle=\langle\mathcal{Q}|\mathsf{m}^\dagger_{r,c}\mathsf{m}_{r,c}|\mathcal{Q}\rangle.\end{aligned}$$

3. Given the computational basis $\{b_i\}_{i\in[1,2^n]}$ of $\mathcal{H}(\mathcal{QV}-\{r,q\})$, we have that

$$\mathcal{Q}=\sum_{i=1}^{2^n}\alpha_i|r\mapsto 0\rangle\otimes|q\mapsto 0\rangle\otimes b_i+\sum_{i=1}^{2^n}\beta_i|r\mapsto 0\rangle\otimes|q\mapsto 1\rangle\otimes b_i+$$
$$\sum_{i=1}^{2^n}\gamma_i|r\mapsto 1\rangle\otimes|q\mapsto 0\rangle\otimes b_i+\sum_{i=1}^{2^n}\delta_i|r\mapsto 1\rangle\otimes|q\mapsto 1\rangle\otimes b_i.$$

Let us show that $\mathsf{m}_{q,0}(\mathsf{m}_{r,0}(\mathcal{Q}))=\mathsf{m}_{r,0}(\mathsf{m}_{q,0}(\mathcal{Q}))$, the proof of other cases follow the same pattern.

$$\begin{aligned}\mathsf{m}_{r,0}(\mathsf{m}_{q,0}(\mathcal{Q}))&=\mathsf{m}_{r,0}\left(\sum_{i=1}^{2^n}\alpha_i|r\mapsto 0\rangle\otimes b_i+\sum_{i=1}^{2^n}\gamma_i|r\mapsto 1\rangle\otimes b_i\right)\\&=\sum_{i=1}^{2^n}\alpha_i b_i=\mathsf{m}_{q,0}\left(\sum_{i=1}^{2^n}\alpha_i|q\mapsto 0\rangle\otimes b_i+\sum_{i=1}^{2^n}\beta_i|q\mapsto 1\rangle\otimes b_i\right)\\&=\mathsf{m}_{q,0}(\mathsf{m}_{r,0}(\mathcal{Q})).\end{aligned}$$

This concludes the proof. □

The following proposition holds for the normalized maps $\mathcal{M}_{r,0}$ and $\mathcal{M}_{r,1}$ (Definition 5):

Proposition 15. Let $\mathcal{Q}\in\mathcal{H}(\mathcal{QV})$ be a quantum register. Then

1. $\mathcal{M}_{r,c}(\mathcal{Q})$ is a quantum register;
2. $\mathcal{M}_{q,e}(\mathcal{Q}\otimes|r\mapsto d\rangle)=(\mathcal{M}_{q,e}(\mathcal{Q}))\otimes|r\mapsto d\rangle$, with $q\in\mathcal{QV}$ and $q\neq r$;
3. $\mathcal{M}_{q,e}(\mathcal{M}_{r,d}(\mathcal{Q}))=\mathcal{M}_{r,d}(\mathcal{M}_{q,e}(\mathcal{Q}))$, with $q,r\in\mathcal{QV}$;
4. if $q,r\in\mathcal{QV}$, $p_{r,c}=\langle\mathcal{Q}|\mathsf{m}_{r,c}^\dagger\mathsf{m}_{r,c}|\mathcal{Q}\rangle$, $p_{q,d}=\langle\mathcal{Q}|\mathsf{m}_{q,d}^\dagger\mathsf{m}_{q,d}|\mathcal{Q}\rangle$, $\mathcal{Q}_{r,c}=\mathcal{M}_{r,c}(\mathcal{Q})$, $\mathcal{Q}_{q,d}=\mathcal{M}_{q,d}(\mathcal{Q})$, $s_{r,c}=\langle\mathcal{Q}_{q,d}|\mathsf{m}_{r,c}^\dagger\mathsf{m}_{r,c}|\mathcal{Q}_{q,d}\rangle$, $s_{q,d}=\langle\mathcal{Q}_{r,c}|\mathsf{m}_{q,d}^\dagger\mathsf{m}_{q,d}|\mathcal{Q}_{r,c}\rangle$ then $p_{r,c}\cdot s_{q,d}=p_{q,d}\cdot s_{r,c}$;
5. $(\mathbf{U}_{\langle q_1,\ldots,q_k\rangle}\otimes\mathbf{I}_{\mathcal{QV}-\{q_1,\ldots,q_k\}})(\mathcal{M}_{r,c}(\mathcal{Q}))=\mathcal{M}_{r,c}((\mathbf{U}_{\langle q_1,\ldots,q_k\rangle}\otimes\mathbf{I}_{\mathcal{QV}-\{q_1,\ldots,q_k\}})(\mathcal{Q}))$ with $\{q_1,\ldots,q_k\}\subseteq\mathcal{QV}$ and $r\neq q_j$ for all $j=1,\ldots,k$.

Proof. The proofs of 1, 2 and 5 are immediate consequences of Proposition 14 and of general basic properties of Hilbert spaces. About 3 and 4: if $\mathcal{Q}=0_{\mathcal{QV}}$ then the proof is trivial; if either $p_{r,c}=0$ or $p_{q,d}=0$ (possibly both), observe that $s_{r,c}=s_{q,d}=0$ and $\mathcal{M}_{q,e}(\mathcal{M}_{r,d}(\mathcal{Q}))=\mathcal{M}_{r,d}(\mathcal{M}_{q,e}(\mathcal{Q}))=0_{\mathcal{QV}-\{q,r\}}$ and conclude. Suppose now that $\mathcal{Q}\neq 0_{\mathcal{QV}}$, $p_{r,c}\neq 0$ and $p_{q,d}\neq 0$. Given the computational basis $\{b_i\}_{i\in[1,2^n]}$ of $\mathcal{H}(\mathcal{QV}-\{r,q\})$, we have that:

$$\mathcal{Q}=\sum_{i=1}^{2^n}\alpha_i|r\mapsto 0\rangle\otimes|q\mapsto 0\rangle\otimes b_i+\sum_{i=1}^{2^n}\beta_i|r\mapsto 0\rangle\otimes|q\mapsto 1\rangle\otimes b_i+$$
$$\sum_{i=1}^{2^n}\gamma_i|r\mapsto 1\rangle\otimes|q\mapsto 0\rangle\otimes b_i+\sum_{i=1}^{2^n}\delta_i|r\mapsto 1\rangle\otimes|q\mapsto 1\rangle\otimes b_i.$$

Let us examine the case $c = 0$ and $d = 0$ (the other cases can be handled in the same way).

$$p_{r,0} = \sum_{i=1}^{2^n} |\alpha_i|^2 + \sum_{i=1}^{2^n} |\beta_i|^2; \qquad p_{q,0} = \sum_{i=1}^{2^n} |\alpha_i|^2 + \sum_{i=1}^{2^n} |\gamma_i|^2;$$

$$\mathcal{Q}_{r,0} = \mathcal{M}_{r,0}(\mathcal{Q}) = \frac{\sum_{i=1}^{2^n} \alpha_i |q \mapsto 0\rangle \otimes b_i + \sum_{i=1}^{2^n} \beta_i |q \mapsto 1\rangle \otimes b_i}{\sqrt{p_{r,0}}}$$

$$\mathcal{Q}_{q,0} = \mathcal{M}_{q,0}(\mathcal{Q}) = \frac{\sum_{i=1}^{2^n} \alpha_i |r \mapsto 0\rangle \otimes b_i + \sum_{i=1}^{2^n} \gamma_i |r \mapsto 1\rangle \otimes b_i}{\sqrt{p_{q,0}}}$$

Now let us consider the two states:

$$\mathcal{Q}_{r,0}^{q,0} = \mathsf{m}_{q,0}(\mathcal{Q}_{r,0}) = \frac{\sum_{i=1}^{2^n} \alpha_i b_i}{\sqrt{p_{r,0}}} \qquad \mathcal{Q}_{q,0}^{r,0} = \mathsf{m}_{r,0}(\mathcal{Q}_{q,0}) = \frac{\sum_{i=1}^{2^n} \alpha_i b_i}{\sqrt{p_{q,0}}}$$

By definition

$$s_{q,0} = \frac{\sum_{i=1}^{2^n} |\alpha_i|^2}{p_{r,0}} \qquad s_{r,0} = \frac{\sum_{i=1}^{2^n} |\alpha_i|^2}{p_{q,0}}$$

and therefore $p_{r,0} \cdot s_{q,d} = p_{q,0} \cdot s_{r,0}$. Moreover, if $\mathcal{QV} = \varnothing$ then $\mathcal{M}_{q,0}(\mathcal{Q}_{r,0}) = \mathcal{M}_{r,0}(\mathcal{Q}_{q,0}) = 1$, otherwise

$$\mathcal{M}_{q,0}(\mathcal{Q}_{r,0}) = \frac{\mathcal{Q}_{r,0}^{q,0}}{\sqrt{\bar{p}_{q,0}}} = \frac{\sum_{i=1}^{2^n} \alpha_i b_i}{\sqrt{p_{r,0}} \cdot \sqrt{\bar{p}_{q,0}}} = \frac{\sum_{i=1}^{2^n} \alpha_i b_i}{\sqrt{p_{r,0}} \cdot \sqrt{\frac{\sum_{i=1}^{2^n} |\alpha_i|^2}{p_{r,0}}}} = \frac{\sum_{i=1}^{2^n} \alpha_i b_i}{\sqrt{\sum_{i=1}^{2^n} |\alpha_i|^2}}$$

$$\mathcal{M}_{r,0}(\mathcal{Q}_{s,0}) = \frac{\mathcal{Q}_{q,0}^{r,0}}{\sqrt{\bar{p}_{r,0}}} = \frac{\sum_{i=1}^{2^n} \alpha_i b_i}{\sqrt{p_{q,0}} \cdot \sqrt{\bar{p}_{r,0}}} = \frac{\sum_{i=1}^{2^n} \alpha_i b_i}{\sqrt{p_{q,0}} \cdot \sqrt{\frac{\sum_{i=1}^{2^n} |\alpha_i|^2}{p_{q,0}}}} = \frac{\sum_{i=1}^{2^n} \alpha_i b_i}{\sqrt{\sum_{i=1}^{2^n} |\alpha_i|^2}}$$

and therefore, $\mathcal{M}_{q,0}(\mathcal{Q}_{r,0}) = \mathcal{M}_{r,0}(\mathcal{Q}_{s,0})$.
This concludes the proof. □

References

AA.VV. (2013) Website of the QML Project. Available at: http://fop.cs.nott.ac.uk/qml/.

Abramsky, S., Haghverdi, E. and Scott, P. J. (2002) Geometry of Interaction and Linear Combinatory Algebras. *Mathematical Structures in Computer Science* **12** (5) 625–665.

Aharonov, D., Kitaev, A. and Nisan, N. (1998) Quantum circuits with mixed states. In: *Proceedings of the 30th Annual ACM Symposium on the Theory of Computing* 20–30.

Aharonov, D., van Dam, W., Kempe, J., Landau, Z., Lloyd, S. and Regev, O. (2007) Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM Journal on Computing* **37** (1) 166–194.

Altenkirch, T. and Grattage, J. (2005) A functional quantum programming language. In: *Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science* 249–258.

Altenkirch, T., Grattage, J., Vizzotto, J. and Sabry, A. (2007) An algebra of pure quantum programming. *Electronic Notes in Theoretical Computer Science* **210** (1) 23–47.

Arora, S. and Barak, B. (2009) *Computational Complexity: A Modern Approach*, Cambridge University Press.

Arrighi, P., Díaz-Caro, A., Gadella, M. and Grattage, G. (2008) Measurements and confluence in quantum Lambda calculi with explicit qubits. *Electronic Notes in Theoretical Computer Science* **270** (1) 59–74.

Arrighi P., Díaz-Caro, A. and Valiron B. (2012a) A system F accounting for scalars. *Logical Methods in Computer Science* **8** 1–11.

Arrighi P., Díaz-Caro, A. and Valiron B. (2012b) A type system for the vectorial aspects of the linear-algebraic lambda-calculus. In: Proceedings of the 7th International Workshop on Developments of Computational Methods. *Electronic Proceedings in Theoretical Computer Science* **88** 1–15.

Arrighi, P. and Dowek, G. (2008) Linear-algebraic lambda-calculus: Higher-order, encodings and confluence. In: *Proceedings of the 19th Annual Conference on Term Rewriting and Applications* 17–31.

Baillot, P., Coppola, P. and Dal Lago, U. (2011) Light logics and optimal reduction: Completeness and complexity. *Information and Computation* **209** (2) 118–142.

Baillot, P., Marion, Y.-J., Ronchi Della Rocca S. (eds.) (2009) Special issue on implicit complexity. *ACM Transactions on Computational Logic* **10** (4).

Baillot, P. and Mogbil, V. (2004) Soft lambda-calculus: A language for polynomial time computation. In: Proceedings of the 7th International Conference Foundations of Software Science and Computation Structures. *Lecture Notes in Computer Science* **2987** 27–41.

Barebdregt, H. (1984) *The lambda calculus: its Syntax and Semantics*, North-Holland Publishing Co.

Basdevant, J. L. and Dalibard, J. (2005) *Quantum Mechanics*, Springer.

Benioff, P. (1980) The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines. *Journal of Statistical Physics* **5** 563–591.

Bennett, C. H. and Landauer, R. (1985) The fundamental physical limits of computation. *Scientific American* **253** (1) 48–56.

Bernstein, E. and Vazirani, U. (1997) Quantum complexity theory. *SIAM Journal on Computing* **26** (5) 1411–1473.

Birkhoff, G. and Mac Lane, S. (1967) *Algebra*, The Macmillan Co. press.

Brassard, G. (1994) Cryptology column–quantum computing: The end of classical cryptography? *SIGACT News* **25** (4) 15–21.

Conway, J. (1990) *A Course in Functional Analysis*, Springer-Verlag.

Dalla Chiara M. L., Giuntini, R. and Leporini, R (2003) Quantum computational logics. A survey. In: *Proceedings of Trends in Logic. 50 Years of Studia Logica* 229–271.

Dal Lago, U., Masini, A. and Zorzi, M. (2009) On a measurement-free quantum lambda calculus with classical control. *Mathematical Structures in Computer Science* **19** (2) 297–335.

Dal Lago, U., Masini, A. and Zorzi, M. (2010) Quantum implicit computational complexity. *Theoretical Computer Science* **411** (2) 377–409.

Dal Lago, U., Masini, A. and Zorzi, M. (2011) Confluence results for a quantum lambda calculus with measurements. *Electronics Notes in Theoretical Computer Science* **207** (2) 251–261.

Dal Lago, U. and Zorzi, M. (2012) Probabilistic operational semantics for the lambda calculus. *RAIRO - Theoretical Informatics and Applications* **46** (3) 413–450.

Dal Lago, U. and Zorzi, M. (2013) Wave-style token machines and quantum lambda calculi. Long version available at: http://arxiv.org/abs/1307.0550.

Dal Lago, U. and Zorzi, M. (2014) Wave-style token machines and quantum lambda calculi. In: *Informal Proceedings (with revision) of LINEARITY'14 (FloC'14)*, 13th July, Vienna 8 pp.

Danos, V., Kashefi, E. and Panangaden, P. (2005) Distributed measurement based quantum computation. *Electronic Notes in Theoretical Computer Science* **170** 73–94.

Danos, V., Kashefi, E. and Panangaden, P. (2007) The measurement calculus. *Journal of the ACM* **52** (2) Article No. 8.

Danos, V., Kashefi, E., Panangaden, P. and Perdrix S. (2009) Extended measurement calculus. In: *Semantic Techniques in Quantum Computation*. Cambridge University Press 235–310.

Deutsch, D. (1985) Quantum theory, the Church–Turing principle and the universal quantum computer. In: *Proceedings of the Royal Society of London A* **400** 97–117.

Deutsch, D., Ekert, A. and Lupacchini, R. (2000) Machines, logic and quantum physics. *Bullettin of Symbolic Logic* **6** (3) 265–283.

Delbecque, Y. (2011) Game semantics for quantum data. *Electronics Notes in Theoretical Computer Science* **270** (1) 41–57.

Dirac, P. (1947) Quantum theory, the Church–Turing principle and the universal quantum computer. In: *Proceedings of the Royal Society of London A* **A400** 97–117.

Einstein, A., Podolsky, B. and Rosen N. (1935) Can quantum-mechanical description of physical reality be considered complete? *Physical Review* **47** 777–780.

Feynman, R. P. (1982) Simulating physics with computers. *International Journal of Theoretical Physics* **21** (6–7) 467–488.

Gaboardi, M., Marion, J.-Y. and Ronchi Della Rocca, S. (2013) An implicit characterization of PSPACE. *ACM Transactions on Computational Logic* **13** (2) 18:1–18:36.

Gay, S. (2011) Bibliography on quantum programming languages. Available at: `http://www.dcs.gla.ac.uk/ simon/quantum/.`

Gay, S. and Makie, I. (eds.) (2009) *Semantic Techniques in Quantum Computation*, Cambridge University Press.

Girard, J.-Y. (1987) Linear logic. *Theoretical Computer Science* **50** (1) 1–102.

Girard, J.-Y., Scedrov, A. and Scott, P. (1992) Bounded linear logic: A modular approach to polynomial time computability. *Theoretical Computer Science* **97** 1–66.

Girard, J.-Y. (1998) Light linear logic. *Information and Computation* **14** (3) 175–204.

Grattage, J. (2006) *A Functional Quantum Programming Language*, Ph.D. thesis, University of Nottingham.

Grattage, J. (2011) An overview of QML with a concrete implementation in Haskell. *Electronic Notes in Theoretical Computer Science* **270** (1) 165–174.

Grover, L. K. (1999) Quantum search on structured problems. *Quantum Computing and Quantum Communications* **1509** 126–139.

Haghverdi, E. and Scott, P. J. (2002) Geometry of interaction and the dynamics of proof reduction: A tutorial. *New Structures for Physics, Lectures Notes in Physics* **813** 357–417.

Hashuo, I. and Hoshino, N. (2011) Semantics of higher-order quantum computation via geometry of interaction. In: *Proceedings of 26th Annual Symposium on Logic in Computer Science* 237–246.

Hirvensalo, M. (2004) *Quantum Computing*, Springer-Verlag.

Isham, C. (1995) *Lectures on Quantum Theory*, Imperial College Press.

Jain, R., Ji, Z., Upadhyay, S. and Watrous, J. (2011) QIP=PSPACE. *Journal of the ACM* **58** (6) Article No. 30.

Jain, R., Upadhyay, S. and Watrous, J. (2009) Two-message quantum interactive proofs are in PSPACE. In: *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science*, IEEE Computer Society 534–543.

Kaye, P., Laflamme, R. and Mosca, M (2007) *An Introduction to Quantum Computing*, Oxford University Press.

Kitaev, A., Shen, A. and Vyalyi, M. (2002) *Classical and Quantum Computation*, AMS press.

Kleene, S. C. (1936) λ-definability and recursiveness. *Duke Mathematical Journal* **2** (2) 340–353.

Knill, E. (1996) Conventions for quantum pseudocode. Technical Report LAUR-96-2724, Los Alamos National Laboratory.

Lafont, Y. (2004) Soft linear logic and polynomial time. *Theoretical Computer Science* **318** 163–180.

Lanzagorta, M. and Uhlmann, J. (2009) *Quantum Computer Science*, Morgan and Claypool Press.

Maccone, L. and Salasnish, L. (2008) *Fisica moderna. Meccanica quantistica, caos e sistemi complessi*, Carocci Press.

Masini, A., Viganò, L. and Zorzi, M. (2008) A qualitative modal representation of quantum register transformations. In: *Proceedings of the 38th IEEE International Symposium on Multiple-Valued Logic (22–23 May 2008, Dallas, Texas, USA),* IEEE CS Press 131–137.

Masini, A., Viganò, L. and Zorzi, M. (2011) Modal deduction systems for quantum state transformations. *Multiple-Valued Logic and Soft Computing* **17** (5–6) 475–519.

Maymin, P. (1996) Extending the λ-calculus to express randomized and quantumized algorithms. `ArXiv:quant-ph/9612052`. Unpublished.

Maymin, P. (1997) The lambda-q calculus can efficiently simulate quantum computers. `ArXiv:quant-ph/9702057`. Unpublished.

Nakahara, M. and Ohmi, T. (2008) *Quantum Computing. From Linear Algebra to Physical Realizations*, CRC Press.

Nielsen, M. A. (2003) Universal quantum computation using only projective measurement, quantum memory, and preparation of the 0 state. *Physical Letters A* **308** (2-3) 96–100.

Nielsen, M. A. and Chuang, I. L. (2000) *Quantum Computation and Quantum Information. Physical Review,* volume 308, Cambridge University Press.

Nishimura, H. and Ozawa, M. (2002) Computational complexity of uniform quantum circuit families and quantum Turing machines. *Theoretical Computer Science* **276** 147–181.

Nishimura, H. and Ozawa, M. (2009) Perfect computational equivalence between quantum Turing machines and finitely generated uniform quantum circuit families. *Quantum Information Processing* **8** (1) 12–24.

Papadimitriou, C. (1994) *Computational Complexity*, Addison-Wesley.

Perdrix, S. (2006) *Modèles formels du calcul quantique: Ressources, machines abstraites et calcul par mesure*, Ph.D. thesis, Institut National Polytechnique de Grenoble.

Perdrix, S. (2007) Quantum patterns and types for entanglement and separability. *Electronic Notes in Theoretical Computer Science* **170** 125–138.

Perdrix, V. and Jorrand, P., (2006) Classically controlled quantum computation. *Mathematical Structures in Computer Science* **16** (4) 601–620.

Preskill, J. (2006) *Quantum Information and Computation*, Lecture Notes in Physics volume 229, Rinton Press.

Roman, S. (2008) *Advanced Linear Algebra*, Springer.

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

Selinger, P. and Valiron, B. (2006) A λ-calculus for quantum computation with classical control. *Mathematical Structures in Computer Science* **16** (3) 527–552.

Selinger, P. and Valiron, B. (2009) Quantum λ-calculus. In: *Semantic Techniques in Quantum Computation,* Cambridge University Press 135–172.

Shor, P. W. (1994) Algorithms for quantum computation: Discrete logarithms and factoring. In: *Proceedings of 35th Annual Symposium on Foundations of Computer Science* 124–134.

Shor, P. W. (1997) Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing* **26** 1484–1509.

Simon, D. (1994) On the power of quantum computation. *SIAM Journal on Computing* **26** 116–123.

Simpson, A. (2005) Reduction in a linear lambda-calculus with applications to operational semantics. In: *Proceedings of the 16th Annual Conference on Term Rewriting and Applications* 219–234.

van Tonder, A. (2004) A λ-calculus for quantum computation. *SIAM Journal on Computing* **33** (5) 1109–1135.

Vaux, L. (2006) Lambda-calculus in an algebraic setting. *Research report*, Institut de Mathmatiques de Luminy.

Vaux, L. (2009) The algebraic lambda-calculus. *Mathematical Structures in Computer Science* **19** (5) 1029–1059.

Volpe, M., Viganò, L. and Zorzi, M. (2014) Quantum state transformations and branching distributed temporal logic. In: Proceedings of the 21st Workshop on Logic, Language, Information and Computation (1–4 September, 2014, Valparaiso, Chile). *Lecture Notes in Computer Science* **8652** 1–19.

Wadler, P. (1994) A syntax for linear logic. In: *Proceedings of the 9th International Conference on Mathematical Foundations of Programming Semantics* 513–529.

Wadsworth, C. (1980) Some unusual λ-calculus numeral systems. In: *To H.B. Curry: Essays on Combinatory Logic, λ-Calculus and Formalism*, Academic Press.

Wootters, W. K. and Zurek, W. H. (1982) A single quantum cannot be cloned. *Nature* **299** 802–803.

Yao, A. (1993) Quantum circuit complexity. In: *Proceedings of the 34th Annual Symposium on Foundations of Computer Science* 352–360.

Zorzi, M. (2009) *Lambda Calculi and Logics for Quantum Computing*, Ph.D. thesis, Dipartimento di Informatica, Università degli Studi di Verona.