Under consideration for publication in Math. Struct. in Comp. Science

On a Measurement-Free Quantum Lambda Calculus with Classical Control

Ugo Dal Lago

Dipartimento di Scienze dell'Informazione, Università di Bologna

dallago@cs.unibo.it

Andrea Masini

Dipartimento di Informatica, Università di Verona

andrea.masini@univr.it

Margherita Zorzi

Dipartimento di Informatica, Università di Verona

zorzim@sci.univr.it

Received 15 April 2008

We study a measurement-free, untyped λ–calculus with quantum data and classical control. This work stems from previous proposals by Selinger and Valiron and by Van Tonder. We focus on operational and expressiveness issues, rather than (denotational) semantics. We prove subject reduction, confluence and a standardization theorem. Moreover, we prove the computational equivalence of the proposed calculus with a suitable class of quantum circuit families.

1. Introduction

Quantum computing was conceived at the beginning of the eighties, starting from an idea by Feynman (Feynman 1981). It defines an alternative computational paradigm, based on quantum mechanics (Basdevant & Dalibard 2005) rather than digital electronics. The first proposal for a quantum abstract computer is due to Deutsch, who introduced quantum Turing machines (Deutsch 1985). Other quantum computational models have been subsequently defined by Yao (quantum circuits, (Yao 1993)) and Knill (quantum random access machines, (Knill 1996)). There are also other models of quantum computations, e.g. measurement–based (Danos et al. 2007), adiabatic (Aharonov et al. 2007).

The introduction of quantum abstract machines made it possible to develop a complexity theory of quantum computation. One of the most important result in quantum

complexity theory has been obtained by Shor, who showed that integers can be factorized in polynomial time (Shor 1994). We would like to stress also the importance of Grover's algorithm (Grover 1999) that definitely improves on the best classical complexity.

Nowadays, what are the main challenges in quantum computing? A lot of research is being devoted to understanding whether quantum computation can provide efficient algorithms for classically intractable problems. In the last years, the impressive results obtained in this area (e.g. Shor's fast factoring algorithm) have stimulated the development of quantum programming languages. The situation is not as easy as in the classical case. In addition to the concrete technical problems (up to now it is difficult to build even very simple quantum circuits), there is the necessity of developing adequate *calculi of quantum computable functions*. In particular, it is not clear how the idea of having functions as "first-class citizens" can be captured in a quantum setting. Indeed, quantum circuits and quantum Turing machines (the most widely known models of quantum computation) are essentially "first-order".

This paper is an attempt to give a contribution to the definition of a measurement-free quantum computational model for higher–order–functions.

The first attempt to define a quantum higher–order language has been done (to the authors' knowledge) in two unpublished papers by Maymin (Maymin 1996; Maymin 1997). Selinger (Selinger 2004) rigorously defined a first-order quantum functional language. Another interesting proposal in the framework of first-order quantum functional languages is the language QML (Altenkirch & Grattage 2005). Arrighi and Dowek has recently proposed an interesting extension of λ–calculus with potential applications in the field of quantum computing (Arrighi & Dowek 2006).

Focusing on higher-order functional programming languages, at least two distinct *foundational* proposals have already appeared in the literature: that by Selinger and Valiron (Selinger & Valiron 2006) (see also an interesting extension proposed by Perdrix (Perdrix 2005)) and the one by Van Tonder (van Tonder 2004). These two approaches seems to go in orthogonal directions: in the language proposed by Selinger and Valiron data (registers of qubits) are superimposed while control (lambda terms) is classical, whereas the approach of Van Tonder at a first glance *seems* to be based on the idea of putting *arbitrary λ–terms in superposition*. But, *is this the right picture*? In order to give an answer, let us examine more closely the two approaches.

Selinger and Valiron's Approach. The main goal of the work of Selinger and Valiron is to give the basis of a typed quantum functional language (with types in propositional multiplicative and exponential linear logic). The idea of Selinger and Valiron is to have defined a language where only data are superposed, and where programs live in a standard classical world. In particular, it is not necessary to have "exotic" objects such as λ–terms in superposition. The approach is well condensed by the slogan: *"classical control + quantum data"*. 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 allowing the program to observe the value of one of the quantum data.

Unfortunately, the expressive power of λ_{sv} has not been studied yet. The crucial issue

is the following: can we compare the expressive power of λ_{sv} with that of any well known computational model (e.g. quantum Turing machines or quantum circuits families)?

Van Tonder's Approach. The calculus introduced by Van Tonder (van Tonder 2004), called λ_q, has the same motivation and a number of *immediate similarities* with λ_{sv}, noticeably, the exploitation of linear types in controlling both copying and erasing of terms.

But there is a couple of glaring differences between λ_q and λ_{sv}. In fact it seems that λ_q *allows by design arbitrary superpositions of* λ*-terms.* In our opinion the essence of the approach of Van Tonder is in Lemma 5.1 of (van Tonder 2004), where it is stated that "*two terms* M*,* N *in superposition differ only for qubits values*". Moreover, if M reduces to M' and N reduces to N', the reduced redex in M is (up to quantum bits) the same redex reduced in N. This means λ_q has classical control, too: it is not possible to superimpose terms differing in a remarkable way, i.e. terms with a different computational evolution. Moreover, in λ_q measurement is not internalised, i.e. there is not any measurement operator as in λ_{sv}.

The weak point of Van Tonder's paper, is that some results and proofs are given too informally. In particular, the paper argues that the proposed calculus is computationally equivalent to quantum Turing machines without giving a detailed proof and, more importantly, without specifying which class of quantum Turing machines is considered (this is not pedantry). But clearly, such a criticism does not invalidate the foundational importance of the approach.

Our Proposal

In order to avoid possible misunderstandings, we start by stressing *what our proposal is not.*

- *We do not propose a new computational paradigm.* Indeed, we adopt the paradigm "quantum data and classical control" as proposed by Selinger and Valiron (and, implicitly, by Van Tonder); we extensively develop such a paradigm showing its potentiality from a computation-theoretic point of view. Moreover, we start our study with a measurement-free calculus, in this sense following Van Tonder.
- *We do not propose a programming language.* Our emphasis is on computability, not on the development of more or less concrete programming languages (exactly as for pure λ–calculus, that is not a programming language, but a calculus of computable functions). The development of quantum functional languages is certainly interesting, but is not our concern here.

Having clarified these facts we are ready to give an answer to the main question: *what is our proposal?* Our goal is to make a deep investigation on the "quantum data and classical control" paradigm in absense of measurement. In this sense, this paper can be seen both as a *continuation* and an *extention* of the two proposals we have just described.

- It is a *continuation* , because we propose a quantum λ–calculus with classical control and quantum data. We use a syntax for terms and configurations inspired by that of

Selinger and Valiron and moreover we implicitly use linear logic in a way similar to Van Tonder's λ_q.
- It is an *extension*, because we have focused on the *operational* and *expressiveness* study of the calculus.

The Operational Study. Even if the proposed calculus is untyped term formation is constrained by means of *well forming rules* (the structure of terms is strongly based on the formulation of Linear Logic as proposed by P. Wadler in (Wadler 1994)). In order to be correct wrt term reduction we have proved a suitable *subject reduction theorem*. The calculus we introduce here is not endowed with reduction strategy (it is neither call–by–value nor call–by–name). We prove *confluence*, which holds in a strong sense. Noticeably, a configuration (the quantum generalization of a term) is strongly normalizing iff it is weakly normalizing. Another remarkable feature of the calculus is given by the *(quantum) standardization theorem*. Roughly speaking: for each terminating computation there is another "canonical", equivalent, computation where computational steps are performed in the following order:

1. first, *classical* reduction: in this phase the quantum register is empty and all the computations steps are classical;
2. secondly, reductions that *build the quantum register*;
3. and finally *quantum* reductions, applying unitary transformations to the quantum register.

We think that standardization sheds some further light on the dynamics of quantum computation.

The Expressiveness Study. An important issue is the *real* expressive power of the proposed calculus. In particular: what is the relationship between this calculus and other quantum computing systems, such as quantum Turing machines (*à la* Vazirani and Berstein) and Quantum Families of Circuits (remember that the two formalisms have been showed to be equivalent (Nishimura & Ozawa 2002))? In order to face the expressive power problem, we prove the *equivalence between our calculus and quantum circuit families*. To our knowledge, it is the first time such a study has been done in a detailed and rigorous way for a quantum λ–calculus. The equivalence proofs are based on the standardization theorem and on suitable encodings.

On the Absence of Measurement Some words on the absence of an explicit measurement operator are now in order (see Section 8 for a more detailed discussion). A main concern of our proposal is about the absence of measurement, in the spirit of other quantum computational models like quantum circuits and quantum Turing machines. This is obviously a limitation of our system. Indeed, there is no doubt that any respectable quantum programming language should have a built-in measurement construct (indeed Selinger and Valiron equipped their call–by–value quantum language with a measurement operator). The possibility of making measurement is a key feature of any (quantum) programming language, because it simplifies the writing of programs.

On the other hand, the absence of measurement is a feature of standard quantum computability systems, such as QTM and the quantum circuit model. In all such systems there is no evidence how to *directly* code algorithms where classical computational steps depend on measurements (see e.g. the Shor's factoring algorithm), not to say of infinite quantum computations, where, it is possible to perform infinite sequences of quantum steps interleaved by measurements. But *explicit measurement* is not really needed in a calculus for total quantum computable functions: it is well known that any total quantum function can be formulated without introducing an explicit measurement operator. In practice it is possible to make a unique *implicit measurement* at the end of computation (see e.g. (Bernstein & Vazirani 1997), pag. 1420, (Nielsen & Chuang 2000) pages 185–187 and the introductory sections of (Selinger 2004) and (Selinger & Valiron 2006)). We have adopted this point of view, *assuming a unique implicit measurement at the end of computation*. And we will prove that, even in the absence of measurement, a computationally complete calculus is obtained.

This is just the first step in a long-term research effort oriented towards better understanding of the expressive power of higher-order quantum computational models.

Why "λ–calculus"? We have chosen λ–calculus as a basis of our proposal for a number of reasons:

- first of all, quantum computability and complexity theory are quite underdeveloped compared to their classical counterparts; in particular, there is almost no result relating classes of (first-order) functions definable in pure and typed λ–calculi with classes of functions from computability and complexity theory (in contrast with classical computability theory (Kleene 1936));
- we believe that the higher–order nature of λ–calculi could be useful in understanding the interactions between the classical world (the world of terms) and the quantum world (quantum registers). Those interactions are even stronger in presence of measurement.

The Structure of the Paper

The paper is structured as follows:

- in Section 2 we give the mathematical background on Hilbert Spaces (in order to define quantum registers);
- in Section 3, a λ–calculus, called the Q, is introduced. The Q has classical control and quantum data. The calculus is untyped, but is equipped with well-formation judgments for terms;
- in Section 4, we syntactically study the Q by means of a suitable formulation of subject reduction theorem and confluence theorems;
- in Section 5, we give some example of terms, configurations and computations;
- in Section 6, a further result on the dynamics of the Q is given by means of a standardization theorem (as explained above);
- in Sections 7, we study in detail the equivalence of the Q with Quantum Circuit Families;

- in Appendix we recall the basic notion about Hilbert spaces.

2. Mathematical Structures

This section is devoted to mathematical preliminaries. Clearly, we cannot hope to be completely self-contained here. See (Nielsen & Chuang 2000) for an excellent introduction to quantum computation and information.

2.1. *Quantum Computing Basics*

We informally recall here the basic notions on qubits and quantum registers (see (Nielsen & Chuang 2000) for a detailed introduction). In the next subsection such notions will be (re)defined in a rigorous way. The basic unit of quantum computation is called *quantum bit*, or *qubit* for short . The more direct way to represent a quantum bit is a unitary vector in the 2-dimensional Hilbert space $\mathbb{C}^2$. Let us denote with $|0\rangle$ and $|1\rangle$ the elements of an orthonormal basis of $\mathbb{C}^2$.

The states $|0\rangle$ and $|1\rangle$ of a qubit correspond to 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 $|\alpha|^2 + |\beta|^2 = 1$, can be a possible qubit state. These states are said to be *superposed*, and the two values α and β are called *amplitudes*.

While we can determine the state of a classical bit, for a qubit we can not establish with the same precision its quantum state, namely the values of 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 $|\beta|^2$.

In computational models, we need a generalization of the notion of a qubit, called a *quantum register* (Nishimura & Ozawa 2002; Selinger 2004; Selinger & Valiron 2006; van Tonder 2004). A quantum register of arity n is a normalized vector in $\otimes_{i=1}^{n}\mathbb{C}^2$.
We fix an orthonormal basis of $\otimes_{i=1}^{n}\mathbb{C}^2$, namely $\{|i\rangle \mid i \text{ is a binary string of length } n\}$. For example $1/\sqrt{2}|01\rangle + 1/\sqrt{2}|00\rangle \in \mathbb{C}^2 \otimes \mathbb{C}^2$ is a quantum register of two qubits.

An important property of quantum registers of n qubits is the fact that it is not always possible to decompose them into n isolated qubits (mathematically, this means that we are not able to describe the global state as the tensor product of the single states). These particular states are called *entangled* and enjoy properties that we cannot find in any object of classical physics. If (the state of) n qubits are entangled, they behave as connected, independently of the real physical distance. The strength of quantum computation is essentially based on the existence of entangled states.

2.2. *Hilbert Spaces and Quantum Registers*

Even if Hilbert spaces of the shape $\otimes_{i=1}^{n}\mathbb{C}^2 (\simeq \mathbb{C}^{2^n})$ are commonly used when defining quantum registers, other Hilbert spaces will be defined here (in order to help the reader we give in the appendix all the needed general notions on Hilbert spaces). As we will see,

they allow to handle very naturally the interaction between variable names in λ–terms and superimposed data.

A *quantum variable set (qvs)* is any finite set of quantum variables (ranged over by variables like p, r and q).

Usually, qubits of quantum registers are referred by means of their ordinal position (the i-th qubit, ...); in the proposed approach it is more useful to give names to qubits. Given a *qvs*, $\mathcal{V}$, a quantum register will be a function $\phi : \{0,1\}^{\mathcal{V}} \to \mathbb{C}$ (namely, an assignment of an amplitude to each classical valuation $f \in \{0,1\}^{\mathcal{V}}$) s.t. $\sum_{a \in \phi[\{0,1\}^{\mathcal{V}}]} |a|^2 = 1$ (normalization condition).

The set of quantum registers (for a given qvs $\mathcal{V}$) are in practice *normalized vectors* of a finite dimensional Hilbert space $\mathcal{H}(\mathcal{V})$ defined in the following way.

Definition 1 (Hilbert Spaces on $\mathcal{V}$). Let $\mathcal{V}$ be a *qvs* (possibly empty) of cardinality $\#\mathcal{V} = n$, with $\mathcal{H}(\mathcal{V}) = \{\phi \mid \phi : \{0,1\}^{\mathcal{V}} \to \mathbb{C}\}$ we will denote the Hilbert Space of dimension 2^n equipped with:

i. An *inner sum* $+ : \mathcal{H}(\mathcal{V}) \times \mathcal{H}(\mathcal{V}) \to \mathcal{H}(\mathcal{V})$ defined by $(\varphi + \psi)(f) = \varphi(f) + \psi(f)$;
ii. A *multiplication by a scalar* $\cdot : \mathbb{C} \times \mathcal{H}(\mathcal{V}) \to \mathcal{H}(\mathcal{V})$ defined by $(c \cdot \varphi)(f) = c \cdot (\varphi(f))$;
iii. An *inner product* $\langle\ ,\ \rangle : \mathcal{H}(\mathcal{V}) \times \mathcal{H}(\mathcal{V}) \to \mathbb{C}$ defined by $\langle \varphi, \psi \rangle = \sum_{f \in \{0,1\}^{\mathcal{V}}} \varphi(f)^* \psi(f)$.

The space is equipped with the *orthonormal basis* $\mathcal{B}(\mathcal{V}) = \{|f\rangle : f \in \{0,1\}^{\mathcal{V}}\}$.[†] We call *standard* such a basis. For example, 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 $\mathcal{V}' \cap \mathcal{V}'' = \emptyset$. 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 $\mathcal{B}(\mathcal{V}') = \{|f_i\rangle : 0 \leq i < 2^n\}$ and $\mathcal{B}(\mathcal{V}'') = \{|g_j\rangle : 0 \leq j < 2^m\}$ are the orthonormal 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 orthonormal basis $\{|f_i\rangle \otimes |g_j\rangle : 0 \leq i < 2^n, 0 \leq j < 2^m\}$. We will abbreviate $|f\rangle \otimes |g\rangle$ with $|f, g\rangle$. If $\mathcal{V}$ is a qvs, then $I_{\mathcal{V}}$ is the identity on $\mathcal{H}(\mathcal{V})$, which is clearly unitary.

It is easy to show that if $\mathcal{V}' \cap \mathcal{V}'' = \emptyset$ then there is a standard *isomorphism*

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

In the rest of the paper we will assume to work up-to such an isomorphism[‡].

As for the case of $\mathbb{C}^{2^n}$, we need to define the notion of a *quantum register*.

Definition 2 (Quantum Register). Let $\mathcal{V}$ be a qvs, a *quantum register* is a normalized vector in $\mathcal{H}(\mathcal{V})$.

In particular 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 previous stated isomorphism) we will say that

[†] $|f\rangle : \{0,1\}^{\mathcal{V}} \to \mathbb{C}$ is defined by: $|f\rangle(g) = \begin{cases} 1 \text{ if } f = g \\ 0 \text{ if } f \neq g \end{cases}$

[‡] 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 $i_s(\mathcal{Q} \otimes |r \mapsto c\rangle) \in \mathcal{H}(\mathcal{V} \cup \{r\})$

$\mathcal{Q}' \otimes \mathcal{Q}''$ is a quantum register in $\mathcal{H}(\mathcal{V}' \cup \mathcal{V}'')$. In the rest of the paper we will denote with 1 the empty quantum register (that belongs to $\mathcal{H}(\emptyset)$).

Quantum computing is essentially based on the application of unitary operators to quantum registers. A linear operator $U : \mathcal{H}(\mathcal{V}) \to \mathcal{H}(\mathcal{V})$ is called *unitary* if for all $\phi, \psi \in \mathcal{H}(\mathcal{V})$, $\langle U(\phi), U(\psi)\rangle = \langle \phi, \psi\rangle$. The tensor product of unitary operators is defined as follows: $(U \otimes V)(\phi \otimes \psi) = U(\phi) \otimes U(\psi)$.

But which unitary operators are available in Q? We here assume that unitary operators can be chosen from $\mathcal{U}$, an arbitrary but fixed set of unitary operators, called *elementary operators*. Clearly, the expressive power of Q depends on this choice. If one want, for example, to capture Quantum Turing machines in the style of Bernstein and Vazirani (Bernstein & Vazirani 1997), one could fix $\mathcal{U}$ to be the set of so-called computable operators. On the other hand, the expressivity results in this paper relates Q and quantum circuit families; clearly, those that can be captured by Q terms with elementary operators in $\mathcal{U}$ are precisely those (finitely) generated by $\mathcal{U}$.

Let $U : \mathbb{C}^{2^n} \to \mathbb{C}^{2^n}$ be an elementary operator and let $\langle q_0, \ldots, q_{n-1}\rangle$ be any sequence of distinguished variables. U and $\langle q_0, \ldots, q_{n-1}\rangle$ induce an operator $U_{\langle q_0,\ldots,q_{n-1}\rangle} : \mathcal{H}(\{q_0, \ldots, q_{n-1}\}) \to \mathcal{H}(\{q_0, \ldots, q_{n-1}\})$ defined as follows: if $|f\rangle = |q_{j_0} \mapsto b_{j_0}, \ldots, q_{j_{n-1}} \mapsto b_{j_{n-1}}\rangle$ is an element of the orthonormal basis of $\mathcal{H}(\{q_0, \ldots, q_{n-1}\})$, then

$$U_{\langle q_0,\ldots,q_{n-1}\rangle}|f\rangle \stackrel{def}{=} U|b_{j_0}, \ldots, b_{j_{n-1}}\rangle.$$

Let $\mathcal{V}' = \{q_{i_0}, \ldots, q_{i_k}\} \subseteq \mathcal{V}$. We naturally extend (by suitable standard isomorphisms) the unitary operator $U_{\langle q_{j_0},\ldots,q_{j_k}\rangle} : \mathcal{H}(\mathcal{V}') \to \mathcal{H}(\mathcal{V}')$ to the unitary operator $U_{\langle\langle q_{j_0},\ldots,q_{j_k}\rangle\rangle} : \mathcal{H}(\mathcal{V}) \to \mathcal{H}(\mathcal{V})$ that acts as the identity on variables not in $\mathcal{V}'$ and as $U_{\langle q_{j_0},\ldots,q_{j_k}\rangle}$ on variables in $\mathcal{V}'$.

Example 1. Let us consider the standard operator

$$\mathbf{cnot} : \mathbb{C}^2 \otimes \mathbb{C}^2 \to \mathbb{C}^2 \otimes \mathbb{C}^2.$$

Intuitively, the **cnot** operator complements the target bit (the second one) if the control bit is 1, and otherwise does not perform any action:

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

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:

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

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 bit while $\mathbf{cnot}_{\langle\langle q,p\rangle\rangle}$ uses q. In general, when writing $U_{\langle\langle p_1,\ldots,p_n\rangle\rangle}$, the order in which the variables appear in the subscript matters.

3. The Syntax of Q

3.1. *A Gentle Introduction*

As written in the introduction, this paper is based on the paradigm *quantum data and classical control* as developed by Selinger and Valiron (Selinger & Valiron 2006).

The proposed quantum λ–calculus is based on the notion of *configuration* (a reformulation of the concept of *program state* (Selinger & Valiron 2006)).

A *configuration* is a triple $[\mathcal{Q}, \mathcal{QV}, M]$ that gives a full instantaneous description of the state of a quantum program, where M is a term from a suitable grammar, $\mathcal{Q}$ is a quantum register, $\mathcal{QV}$ is a set of quantum variables (a superset of those appearing in M). Configurations can evolve in two different ways:

- First of all, configurations can evolve *classically*: the term M is modified, but $\mathcal{Q}$ and $\mathcal{QV}$ are not touched. In other words, the reduction will have the following shape:

$$[\mathcal{Q}, \mathcal{QV}, M] \to [\mathcal{Q}, \mathcal{QV}, N]$$

  where the only relevant component of the step is the λ–term M. In this class of reductions we have all the standard λ–reductions (e.g. β–reduction).
- Configurations, however, can evolve *non-classically*: the term M and the quantum register will interact. There are two ways to modify the underlying quantum register:
  1. The *creation of a new quantum bit*, by reducing a term $\mathtt{new}(c)$ (where c is a classical bit). Such a reduction creates a *new quantum variable name* (a classical object) and a new qubit added to the quantum register. The new quantum variable name is a kind of pointer to the newly created qubit. A new reduction has the shape:

$$[\mathcal{Q}, \mathcal{QV}, M] \to_{\mathsf{new}} [\mathcal{Q}', \mathcal{QV}', N]$$

     where N is obtained by replacing in M the redex $\mathtt{new}(c)$ with a (fresh) variable name r, $\mathcal{Q}'$ is the new quantum register with a new qubit referenced by r and $\mathcal{QV}'$ is simply $\mathcal{QV} \cup \{r\}$.
  2. The application of a *unitary transformation to the quantum register*. This computation step consists in reducing a term $U\langle r_1, \ldots, r_n\rangle$, where U is the name of an unitary operator and $r_1, \ldots, r_n$ are quantum variables. A unitary reduction has the shape:

$$[\mathcal{Q}, \mathcal{QV}, M] \to_{\mathsf{Uq}} [\mathcal{Q}', \mathcal{QV}, N]$$

     where $\mathcal{Q}'$ is $U_{\langle\langle r_1,\ldots,r_n\rangle\rangle}\mathcal{Q}$ and N is obtained by replacing in M the redex $U\langle r_1, \ldots, r_n\rangle$ with $\langle r_1, \ldots, r_n\rangle$.

3.2. *On Linearity*

One of the main features of our calculus (and in general of quantum languages) is linearity, where with linearity we mean that a term is not duplicable nor erasable. In the proposed system, linearity corresponds to the constraint that in every term $\lambda x.M$ there is exactly one free occurrence of the variable x in M This way we are able to guarantee that the "non cloning and non erasing property" is indeed satisfied. Indeed, whenever $(\lambda x.M)N$ and x occurs (free) exactly once in M, the *quantum* variables in $(\lambda x.M)N$ are exactly the ones in $M\{N/x\}$ and if any *quantum* variable occurs once in the redex, it will occur once in the reduct, too.

But even if we cannot duplicate terms with references to quantum data, we need to duplicate and erase classical terms. To this purpose, the syntax of terms includes a modal operator ! (called the "bang" operator). The bang operator has been introduced in term calculi corresponding to MELL (see for example Wadler's syntax (Wadler 1994)) and allows to distinguish between those syntactical objects (λ–terms) that can be duplicated or erased and those that cannot. 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 below.

This is not the only possible solution to the problem of non cloning and non erasing property; other solutions have been proposed in literature, see e.g (Arrighi & Dowek 2006) where it is possible to duplicate base vectors, and (Altenkirch & Grattage 2005) where duplication is modelled by means of sharing.

3.3. *The Language of Terms*

Let us associate to each elementary operator $\mathbf{U_i} \in \mathcal{U}$ a symbol U_i The set of the *term expressions*, or *terms* for short, is defined by the following grammar:

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

We assume to work modulo variable renaming, i.e. *terms are equivalence classes modulo α-conversion*. With $M \equiv M'$ we denote that the terms (equivalence classes) M and M' are syntactically equal. Substitution up to α-equivalence is defined in the usual way.

Let us denote with $\mathbf{Q}(M_1, \ldots, M_k)$ the set of quantum variables occurring in $M_1, \ldots, M_k$. Notice that:

- Variables are either *classical* or *quantum*: the first ones are the usual variables of lambda calculus (and can be bound), while each quantum variable refers to a qubit in the underlying quantum register (to be defined shortly).

- There are two sorts of constants as well, namely *boolean constants* (0 and 1) and *unitary operators*: the first ones are useful for generating qubits and play no role in classical computations, while unitary operators are applied to (tuples of) quantum variables when performing quantum computation.
- The term constructor $\texttt{new}(\cdot)$ creates a new qubit when applied to a boolean constant.

The rest of the calculus is a standard linear lambda calculus, similar to the one introduced in (Wadler 1994). Patterns (and, consequently, lambda abstractions) can only refer to classical variables.

There is not any measurement operator in the language. We will comment on that in Section 8.

3.4. *Judgements and Well–Formed Terms*

Judgements are defined from various notions of environments, that take into account the way the variables are used. Following common notations in type theory and proof theory, 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, x, y, z or the empty set $\emptyset$. Given a classic environment $\Delta = x_1, \ldots, x_n$, $!\Delta$ denotes the set $!x_1, \ldots, !x_n$.
- A *quantum environment* is a (possibly empty) set (denoted by Θ, possibly indexed) of quantum variables. Examples of quantum environments are r_1, r_2, r_3 and the empty set $\emptyset$.
- A *linear environment* is (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.

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

$$\frac{!\Delta \vdash M}{!\Delta \vdash !M}\ \textsf{prom} \qquad \frac{\Lambda_1, !\Delta \vdash M_1 \quad \Lambda_2, !\Delta \vdash M_2}{\Lambda_1, \Lambda_2, !\Delta \vdash (M_1)M_2}\ \textsf{app} \qquad \frac{\Lambda_1, !\Delta \vdash M_1 \cdots \Lambda_k, !\Delta \vdash M_k}{\Lambda_1, \ldots, \Lambda_k, !\Delta \vdash \langle M_1, \ldots, M_k \rangle}\ \textsf{tens}$$

$$\frac{\Gamma \vdash M}{\Gamma \vdash \texttt{new}(M)}\ \textsf{new} \qquad \frac{\Gamma, x_1, \ldots, x_n \vdash M}{\Gamma \vdash \lambda\langle x_1, \ldots, x_n \rangle.M}\ \multimap_1 \qquad \frac{\Gamma, x \vdash M}{\Gamma \vdash \lambda x.M}\ \multimap_2 \qquad \frac{\Gamma, !x \vdash M}{\Gamma \vdash \lambda !x.M}\ \rightarrow$$

Fig. 1. Well Forming Rules

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 1. The rules app and tens are subject to the

constraint that for each $i \neq j$ $\Lambda_i \cap \Lambda_j = \emptyset$ (notice that Λ_i and Λ_j are set of linear and quantum variables, being linear environments). With $d \triangleright \Gamma \vdash M$ we denote 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.

Proposition 1. If a term M is well formed then all the classical variables in it are bound.

4. Computations

As previously written, the computations are defined by means of configurations. A *preconfiguration* is a triple $[\mathcal{Q}, \mathcal{QV}, M]$ where:

- M is a term;
- $\mathcal{QV}$ is a finite quantum variable set 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 3 (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 C', then we will write $C \equiv C'$. The relation $\equiv$ is an equivalence relation.

A *configuration* is an equivalence class of preconfigurations modulo the relation $\equiv$. Let $\mathcal{C}$ be the set of configurations.

Remark 1. The way configurations have been defined, namely quotienting preconfigurations over $\equiv$, is very reminiscent of usual α-conversion in lambda-terms.

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 2. Please notice the presence of two commutative reduction rules (namely $\mathsf{l.cm}$ and $\mathsf{r.cm}$). Since Q is untyped, the rôle of commutative reductions is not guaranteeing that normal forms have certain properties, but rather preventing quantum reductions to block classical ones (see Section 6).

For any subset $\mathscr{S}$ of $\mathscr{L}$, we can construct 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, $\overset{*}{\to}$ will denote the transitive and reflexive closure of $\to$.

Notice that $\to$ is not a strategy, (the only limitation is that we forbid reductions under the scope of a "!"), nevertheless, confluence holds. This is in contrast with λ_{sv}, where

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 transform of quantum register

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

Creation of a new qubit and quantum variable

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

Commutative reductions

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

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

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. 2. Reduction rules.

a strategy is indeed necessary (even if we do not take into account the non-deterministic effects of the measurement operator).

4.1. *Subject Reduction*

In this section we give a subject reduction theorem and some related results.

First of all we stress that, even though Q is type-free, a set of admissible terms is

isolated by way of well–forming rules. It is therefore necessary to prove a suitable subject reduction property.

Variables can be created dynamically in Q. Consider, for example, the reduction

$$[1, \emptyset, \mathtt{new}(0)] \rightarrow_{\mathsf{new}} [|p \mapsto 0\rangle, \{p\}, p].$$

The term $\mathtt{new}(0)$ does not contain any variable, while p is indeed a (quantum) variable. In general, notice that the new reduction rule

$$[\mathcal{Q}, \mathcal{QV}, \mathtt{new}(c)] \rightarrow_{\mathsf{new}} [\mathcal{Q} \otimes |r \mapsto c\rangle, \mathcal{QV} \cup \{r\}, r]$$

generates not only a new qubit, but also the new quantum variable r.

In order to control the creation of quantum variables, the Subject Reduction theorem must be given in the following formulation: if $d \triangleright \Gamma \vdash M$ and $[\mathcal{Q}, \mathcal{QV}, M] \rightarrow [\mathcal{Q}', \mathcal{QV}', M']$ then $\triangleright \ \Gamma, \mathcal{QV}' - \mathcal{QV} \vdash M'$ where $\mathcal{QV}' - \mathcal{QV}$ is the (possibly empty) set of quantum variables generated by the reduction. In our example, we have $\triangleright \vdash \mathtt{new}(0)$ and, indeed, $p \vdash p$ is well formed. In other words, we must guarantee that terms appearing during reduction are well-formed, taking into account the set of quantum variables created in the reduction itself.

Theorem 1 (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 (sketch). In order to prove the theorem we need a number of intermediate results. First we must prove that weakening is admissible, namely:

if $\triangleright \Gamma \vdash M$ and x does not occur in Γ then $\triangleright \Gamma, !x \vdash M$.

Then, as usual, the proof of subject reduction requires suitable *substitution lemmas.* In particular we need to prove that:

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

non linear case: if $\triangleright \Lambda_1, !\Delta, !x \vdash M$ and $\triangleright !\Delta \vdash !N$, then $\triangleright \Lambda_1, !\Delta \vdash M\{N/x\}$.

quantum case: for every non empty 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 $\triangleright \Lambda, !\Delta, r_1, \ldots, r_n \vdash M\{r_1/x_1, \ldots, r_n/x_n\}$.

The proof of subject reduction is standard and proceeds by means of a long and easy induction on the derivation of $\rightarrow$.

For the sake of clarity let us to show a case:

let us suppose that M is $(\lambda x.P)N$ and the reduction rule is $[\mathcal{Q}, \mathcal{QV}, (\lambda x.P)N] \rightarrow_{\mathsf{l}.\beta} [\mathcal{Q}, \mathcal{QV}, P\{N/x\}]$. The derivation of M must be:

$$\dfrac{\dfrac{\begin{array}{c}d_1\\ \vdots\\ \Lambda_1, !\Delta, x \vdash P\end{array}}{\Lambda_1, !\Delta \vdash \lambda x.P}\multimap_2 \qquad \begin{array}{c}d_2\\ \vdots\\ \Lambda_2, !\Delta \vdash N\end{array}}{\Lambda_1, \Lambda_2, !\Delta \vdash (\lambda x.P)N}\,\mathsf{app}$$

We note that the reduction does not modify the $\mathcal{QV}$ set, so we have just to apply the substitution property (linear case) to d_1 and d_2 obtaining $\triangleright \Lambda_1, \Lambda_2, !\Delta \vdash P\{N/x\}$. □

An immediate corollary (trivially provable by induction) is:

Corollary 1. If $\triangleright \Gamma \vdash M$ and $[\mathcal{Q}, \mathcal{QV}, M] \xrightarrow{*} [\mathcal{Q}', \mathcal{QV}', M']$ then $\triangleright\ \Gamma, \mathcal{QV}' - \mathcal{QV} \vdash M$.

The notion of well–formed–judgement can be extended to configurations:

Definition 4. A configuration $[\mathcal{Q}, \mathcal{QV}, M]$ is said to be *well–formed* iff there is a context Γ such that $\Gamma \vdash M$ is well-formed.

As a consequence of Subject Reduction, the set of well–formed configurations is closed under reduction:

Corollary 2. If M is well formed and $[\mathcal{Q}, \mathcal{QV}, M] \xrightarrow{*} [\mathcal{Q}', \mathcal{QV}', M']$ then M' is well formed.

In the following, with *configuration* we will mean *well–formed configuration*. Now, let us give the definitions of normal form, configuration and computation.

Definition 5. 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 with NF the set of configurations in normal form.

We define a computation as a suitable sequence of configurations:

Definition 6. If C_0 is any configuration, a *computation* of length $\varphi \leq \omega$ 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 2. Let $\{[\mathcal{Q}_i, \mathcal{QV}_i, M_i]\}_{i<\varphi}$ be a computation, such that $\mathbf{Q}(M_0) = \emptyset$. 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 inspection of reduction rules we immediately have that $\mathcal{QV}' = \mathbf{Q}(M')$ whenever $\mathcal{QV} = \mathbf{Q}(M)$, and conclude. □

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

4.2. *On the Linearity of the Calculus: Dynamics*

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

the rules do not allow reductions under the scope of a bang.

Let us consider the following well–formed configuration: $[1, \emptyset, (\lambda !x.\mathbf{cnot}\langle x, x\rangle)!(\mathtt{new}(1))]$. It is immediate to observe that $!(\mathtt{new}(1))$ is a duplicable term because it does not contain

references to quantum data and in fact the following is a correct computation:

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

But, what happens if we permit to reduce under the scope of the bang (namely reducing $\mathtt{new}(1)$ before executing the $\mathsf{c}.\beta$–reduction)? We would obtain the wrong computation:

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

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

4.3. *Confluence*

Commutative reduction steps behave very differently to other reduction steps when considering confluence. As a consequence, it is useful to define two subsets of $\mathscr{L}$ as follows:

Definition 7. We distinguish two particular subsets of $\mathscr{L}$, namely $\mathscr{O} = \{\mathsf{r.cm}, \mathsf{l.cm}\}$ and $\mathscr{N} = \mathscr{L} - \mathscr{O}$.

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

First of all, we need to show that whenever $M \to_\alpha N$, the underlying quantum register evolves in a uniform wa:

Lemma 1 (Uniformity). For every M, M' such that $M \to_\alpha M'$, exactly one of the following conditions holds:

1. $\alpha \neq \mathsf{new}$ and there is a unitary transformation $U_{M,M'} : \mathcal{H}(\mathbf{Q}(M)) \to \mathcal{H}(\mathbf{Q}(M))$ such that $[\mathcal{Q},\mathcal{QV},M] \to_\alpha [\mathcal{Q}',\mathcal{QV}',M']$ iff $[\mathcal{Q},\mathcal{QV},M] \in \mathcal{C}$, $\mathcal{QV}' = \mathcal{QV}$ and $\mathcal{Q}' = (U_{M,M'} \otimes I_{\mathcal{QV}-\mathbf{Q}(M)})\mathcal{Q}$.
2. $\alpha = \mathsf{new}$ and there are a constant c and a quantum variable r such that $[\mathcal{Q},\mathcal{QV},M] \to_{\mathsf{new}} [\mathcal{Q}',\mathcal{QV}',M']$ iff $[\mathcal{Q},\mathcal{QV},M] \in \mathcal{C}$, $\mathcal{QV}' = \mathcal{QV} \cup \{r\}$ and $\mathcal{Q}' = \mathcal{Q} \otimes |r \mapsto c\rangle$.

Proof. We go by induction on M. M cannot be a variable nor a constant nor a unitary operator nor a term $!N$. If M is an abstraction $\lambda\pi.N$, then $M' \equiv \lambda\pi.N'$, $N \to_\alpha N'$ and the thesis follows from the inductive hypothesis. Similarly when $M \equiv \lambda!x.N$. If $M \equiv NL$, then we distinguish a number of cases:

- $M' \equiv N'L$ and $N \to_\alpha N'$. The thesis follows from the inductive hypothesis.
- $M' \equiv NL'$ and $L \to_\alpha L'$. The thesis follows from the inductive hypothesis.
- $N \equiv U^n$, $L \equiv \langle r_{i_1},...,r_{i_n}\rangle$ and $M' \equiv \langle r_{i_1},...,r_{i_n}\rangle$. Then case 1 holds. In particular, $\mathbf{Q}(M) = \{r_{i_1},...,r_{i_n}\}$ and $U_{M,M'} = U_{\langle\langle r_{i_1},...,r_{i_n}\rangle\rangle}$.

- $N \equiv \lambda x.P$ and $M' = P\{L/x\}$. Then case 1 holds. In particular $U_{M,M'} = I_{\mathbf{Q}(M)}$.
- $N \equiv \lambda\langle x_1, \ldots, x_n\rangle.P$, $L = \langle r_1, \ldots, r_n\rangle$ and $M' \equiv P\{r_1/x_1, \ldots, r_n/x_n\}$. Then case 1 holds and $U_{M,M'} = I_{\mathbf{Q}(M)}$.
- $N \equiv \lambda!x.P$, $L =!Q$ and $M' \equiv P\{Q/x\}$. Then case 1 holds and $U_{M,M'} = I_{\mathbf{Q}(M)}$.
- $L \equiv (\lambda\pi.P)Q$ and $M' \equiv (\lambda\pi.NP)Q$. Then case 1 holds and $U_{M,M'} = I_{\mathbf{Q}(M)}$.
- $N \equiv (\lambda\pi.P)Q$ and $M' \equiv (\lambda\pi.PL)Q$. Then case 1 holds and $U_{M,M'} = I_{\mathbf{Q}(M)}$.

If $M \equiv \mathtt{new}(c)$ then M' is a quantum variable r and case 2 holds. This concludes the proof. □

Notice that $U_{M,M'}$ is always the identity function when performing classical reduction.

The following technical lemma will be useful when proving confluence:

Lemma 2. Suppose $[\mathcal{Q}, \mathcal{QV}, M] \to_\alpha [\mathcal{Q}', \mathcal{QV}', M']$.

1. If $[\mathcal{Q}, \mathcal{QV}, M\{N/x\}] \in \mathcal{C}$, then
$$[\mathcal{Q}, \mathcal{QV}, M\{N/x\}] \to_\alpha [\mathcal{Q}', \mathcal{QV}', M'\{N/x\}].$$
2. If $[\mathcal{Q}, \mathcal{QV}, M\{r_1/x_1, \ldots, r_n/x_n\}] \in \mathcal{C}$, then
$$[\mathcal{Q}, \mathcal{QV}, M\{r_1/x_1, \ldots, r_n/x_n\}] \to_\alpha [\mathcal{Q}', \mathcal{QV}', M'\{r_1/x_1, \ldots, r_n/x_n\}].$$
3. If $x, \Gamma \vdash N$ and $[\mathcal{Q}, \mathcal{QV}, N\{M/x\}] \in \mathcal{C}$, then
$$[\mathcal{Q}, \mathcal{QV}, N\{M/x\}] \to_\alpha [\mathcal{Q}', \mathcal{QV}', N\{M'/x\}].$$

Proof. Claims 1 and 2 can be proved by induction on the proof of $[\mathcal{Q}, \mathcal{QV}, M] \to_\alpha [\mathcal{Q}', \mathcal{QV}', M']$. Claim 3 can be proved by induction on N. □

A property similar to one-step confluence holds in Q. This is a consequence of having adopted the so-called *surface reduction*: it is not possible to reduce inside a subterm in the form $!M$ and, as a consequence, it is not possible to erase a diverging term. This has been already pointed out (Simpson 2005).

Strictly speaking, one-step confluence does not hold in Q. For example, if $[\mathcal{Q}, \mathcal{QV}, (\lambda\pi.M)((\lambda x.N)L)] \in \mathcal{C}$, then both

$$[\mathcal{Q}, \mathcal{QV}, (\lambda\pi.M)((\lambda x.N)L)] \to_{\mathscr{N}} [\mathcal{Q}, \mathcal{QV}, (\lambda\pi.M)(N\{L/x\})]$$

and

$$[\mathcal{Q}, \mathcal{QV}, (\lambda\pi.M)((\lambda x.N)L)] \to_{\mathscr{O}} [\mathcal{Q}, \mathcal{QV}, (\lambda x.(\lambda\pi.M)N)L] \to_{\mathscr{N}} [\mathcal{Q}, \mathcal{QV}, (\lambda\pi.M)(N\{L/x\})]$$

However, this phenomenon is only due to the presence of commutative rules:

Proposition 3 (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 $\alpha \in \mathscr{O}$ and $\beta \in \mathscr{O}$, then there is F with $D \to_{\mathscr{O}} F$ and $E \to_{\mathscr{O}} F$.
2. If $\alpha \in \mathscr{N}$ and $\beta \in \mathscr{N}$, then there is F with $D \to_{\mathscr{N}} F$ and $E \to_{\mathscr{N}} F$.
3. If $\alpha \in \mathscr{O}$ and $\beta \in \mathscr{N}$, then either $D \to_{\mathscr{N}} E$ or there is F with $D \to_{\mathscr{N}} F$ and $E \to_{\mathscr{O}} F$.

Proof. Let $C \equiv [\mathcal{Q}, QV, M]$. We go by induction on M. M cannot be a variable nor a constant nor a unitary operator. If M is an abstraction $\lambda\pi.N$, then $D \equiv [\mathcal{Q}', \mathcal{QV}', \lambda\pi.N']$, $D' \equiv [\mathcal{Q}'', \mathcal{QV}'', \lambda\pi.N'']$ and

$$\begin{array}{lll} [\mathcal{Q}, \mathcal{QV}, N] & \to_\alpha & [\mathcal{Q}', \mathcal{QV}', N'] \\ [\mathcal{Q}, \mathcal{QV}, N] & \to_\beta & [\mathcal{Q}'', \mathcal{QV}'', N''] \end{array}$$

The IH easily leads to the thesis. Similarly when $M \equiv \lambda!x.N$. If $M \equiv NL$, we can distinguish a number of cases depending on the last rule used to prove $C \to_\alpha D$, $C \to_\beta E$:

- $D \equiv [\mathcal{Q}', \mathcal{QV}', N'L]$ and $E \equiv [\mathcal{Q}'', \mathcal{QV}'', NL']$
  where $[\mathcal{Q}, \mathcal{QV}, N] \to_\alpha [\mathcal{Q}', \mathcal{QV}', N']$ and $[\mathcal{Q}, \mathcal{QV}, L] \to_\beta [\mathcal{Q}'', \mathcal{QV}'', L']$.
  We need to distinguish four sub-cases:
  - If $\alpha, \beta = \mathsf{new}$, then, by Lemma 1, there exist two quantum variables $r', r'' \notin \mathcal{QV}$ and two constants c', c'' such that $\mathcal{QV}' = \mathcal{QV} \cup \{r'\}$, $\mathcal{QV}'' = \mathcal{QV} \cup \{r''\}$, $\mathcal{Q}' = \mathcal{Q} \otimes |r' \mapsto c'\rangle$ and $\mathcal{Q}'' = \mathcal{Q} \otimes |r'' \mapsto c''\rangle$. Applying 1 again, we obtain

    $$\begin{array}{lll} D & \to_{\mathsf{new}} & [\mathcal{Q} \otimes |r' \mapsto c'\rangle \otimes |r''' \mapsto c''\rangle, \mathcal{QV} \cup \{r', r'''\}, N'L'\{r'''/r''\}] \equiv F \\ E & \to_{\mathsf{new}} & [\mathcal{Q} \otimes |r'' \mapsto c''\rangle \otimes |r'''' \mapsto c'\rangle, \mathcal{QV} \cup \{r'', r''''\}, N'\{r''''/r'\}L'] \equiv G \end{array}$$

    As can be easily checked, $F \equiv G$.
  - If $\alpha = \mathsf{new}$ and $\beta \neq \mathsf{new}$, then, by Lemma 1 there exists a quantum variable r and a constant c such that $\mathcal{QV}' = \mathcal{QV} \cup \{r\}$, $\mathcal{Q}' = \mathcal{Q} \otimes |r \mapsto c\rangle$, $\mathcal{QV}'' = \mathcal{QV}$ and $\mathcal{Q}'' = (U_{L,L'} \otimes I_{\mathcal{QV} - \mathbf{Q}(L)})\mathcal{Q}$. As a consequence, applying Lemma 1 again, we obtain

    $$\begin{array}{lll} D & \to_\beta & [(U_{L,L'} \otimes I_{\mathcal{QV} \cup \{r\} - \mathbf{Q}(L)})(\mathcal{Q} \otimes |r \mapsto c\rangle), \mathcal{QV} \cup \{r\}, N'L'] \equiv F \\ E & \to_{\mathsf{new}} & [((U_{L,L'} \otimes I_{\mathcal{QV} - \mathbf{Q}(L)})\mathcal{Q}) \otimes |r \mapsto c\rangle, \mathcal{QV} \cup \{r\}, N'L'] \equiv G \end{array}$$

    As can be easily checked, $F \equiv G$.
  - If $\alpha \neq \mathsf{new}$ and $\beta = \mathsf{new}$, then we can proceed as in the previous case.
  - If $\alpha, \beta \neq \mathsf{new}$, then by Lemma 1, there exist $\mathcal{QV}'' = \mathcal{QV}' = \mathcal{QV}$, $\mathcal{Q}' = (U_{N,N'} \otimes I_{\mathcal{QV} - \mathbf{Q}(N)})\mathcal{Q}$ and $\mathcal{Q}'' = (U_{L,L'} \otimes I_{\mathcal{QV} - \mathbf{Q}(L)})\mathcal{Q}$. Applying 1 again, we obtain

    $$\begin{array}{lll} D & \to_\beta & [(U_{L,L'} \otimes I_{\mathcal{QV} - \mathbf{Q}(L)})((U_{N,N'} \otimes I_{\mathcal{QV} - \mathbf{Q}(N)})\mathcal{Q}), \mathcal{QV}, N'L'] \equiv F \\ E & \to_\alpha & [(U_{N,N'} \otimes I_{\mathcal{QV} - \mathbf{Q}(L)})((U_{L,L'} \otimes I_{\mathcal{QV} - \mathbf{Q}(L)})\mathcal{Q}), \mathcal{QV}, N'L'] \equiv G \end{array}$$

    As can be easily checked, $F \equiv G$.
- $D \equiv [\mathcal{Q}', \mathcal{QV}', N'L]$ and $E \equiv [\mathcal{Q}'', \mathcal{QV}'', N''L]$,
  where $[\mathcal{Q}, QV, N] \to [\mathcal{Q}', QV', N']$ and $[\mathcal{Q}, \mathcal{QV}, N] \to [\mathcal{Q}'', \mathcal{QV}'', N'']$.
  Here we can apply the inductive hypothesis.
- $D \equiv [\mathcal{Q}', \mathcal{QV}', NL']$ and $E \equiv [\mathcal{Q}'', \mathcal{QV}'', NL'']$,
  where $[\mathcal{Q}, QV, L] \to [\mathcal{Q}', QV', L']$ and $[\mathcal{Q}, \mathcal{QV}, L] \to [\mathcal{Q}'', \mathcal{QV}'', L'']$.
  Here we can apply the inductive hypothesis as well.
- $N \equiv (\lambda x.P)$, $D \equiv [\mathcal{Q}, \mathcal{QV}, P\{L/x\}]$, $E \equiv [\mathcal{Q}', \mathcal{QV}', NL']$,
  where $[\mathcal{Q}, \mathcal{QV}, L] \to_\beta [\mathcal{Q}', \mathcal{QV}', L']$.

Clearly $[\mathcal{Q}, \mathcal{QV}, P\{L/x\}] \in \mathcal{C}$ and, by Lemma 2, $[\mathcal{Q}, \mathcal{QV}, P\{L/x\}] \to [\mathcal{Q}', \mathcal{QV}', P\{L'/x\}]$.
Moreover, $[\mathcal{Q}', \mathcal{QV}', NL'] \equiv [\mathcal{Q}', \mathcal{QV}', (\lambda x.P)L'] \to [\mathcal{Q}', \mathcal{QV}', P\{L'/x\}]$

- $N \equiv (\lambda x.P)$, $D \equiv [\mathcal{Q}, \mathcal{QV}, P\{L/x\}]$, $E \equiv [\mathcal{Q}', \mathcal{QV}', (\lambda x.P')L]$,
where $[\mathcal{Q}, \mathcal{QV}, P] \to_\beta [\mathcal{Q}', \mathcal{QV}', P']$.
Clearly $[\mathcal{Q}, \mathcal{QV}, P\{L/x\}] \in \mathcal{C}$ and, by Lemma 2, $[\mathcal{Q}, \mathcal{QV}, P\{L/x\}] \to_\beta [\mathcal{Q}', \mathcal{QV}', P'\{L/x\}]$.
Moreover, $[\mathcal{Q}', \mathcal{QV}', (\lambda x.P')L] \to_\beta [\mathcal{Q}', \mathcal{QV}', P'\{L/x\}]$
- $N \equiv (\lambda !x.P)$, $L \equiv !Q$, $D \equiv [\mathcal{Q}, \mathcal{QV}, P\{Q/x\}]$, $E \equiv [\mathcal{Q}', \mathcal{QV}', (\lambda !x.P')L]$,
where $[\mathcal{Q}, \mathcal{QV}, P] \to_\beta [\mathcal{Q}', \mathcal{QV}', P']$.
Clearly $[\mathcal{Q}, \mathcal{QV}, P\{Q/x\}] \in \mathcal{C}$ and, by Lemma 2, $[\mathcal{Q}, \mathcal{QV}, P\{Q/x\}] \to_\beta [\mathcal{Q}', \mathcal{QV}', P'\{Q/x\}]$.
Moreover, $[\mathcal{Q}', \mathcal{QV}', (\lambda x.P')!Q] \to_\beta [\mathcal{Q}', \mathcal{QV}', P'\{Q/x\}]$
- $N \equiv (\lambda \langle x_1, \ldots, x_n \rangle .P)$, $L \equiv \langle r_1, \ldots, r_n \rangle$, $D \equiv [\mathcal{Q}, \mathcal{QV}, P\{r_1/x_1, \ldots, r_n/x_n\}]$,
$E \equiv [\mathcal{Q}', \mathcal{QV}', (\lambda \langle x_1, \ldots, x_n \rangle .P')L]$,
where $[\mathcal{Q}, \mathcal{QV}, P] \to_\beta [\mathcal{Q}', \mathcal{QV}', P']$.
Clearly $[\mathcal{Q}, \mathcal{QV}, P\{r_1/x_1, \ldots, r_n/x_n\}] \in \mathcal{C}$ and, by Lemma 2,
$[\mathcal{Q}, \mathcal{QV}, P\{r_1/x_1, \ldots, r_n/x_n\}] \to_\beta [\mathcal{Q}', \mathcal{QV}', P'\{r_1/x_1, \ldots, r_n/x_n\}]$.
Moreover, $[\mathcal{Q}', \mathcal{QV}', (\lambda \langle x_1, \ldots, x_n \rangle .P')L] \to_\beta [\mathcal{Q}', \mathcal{QV}', P'\{r_1/x_1, \ldots, r_n/x_n\}]$.
- $N \equiv (\lambda x.P)Q$, $D \equiv [\mathcal{Q}, \mathcal{QV}, (\lambda x.PL)Q]$, $E \equiv [\mathcal{Q}, \mathcal{QV}, (P\{Q/x\})L]$, $\alpha = \mathsf{r.cm}$, $\beta = \mathsf{l}.\beta$.
Clearly, $[\mathcal{Q}, \mathcal{QV}, (\lambda x.PL)Q] \to_{\mathsf{l}.\beta} [\mathcal{Q}, \mathcal{QV}, (P\{Q/x\})L]$.
- $N \equiv (\lambda \pi.P)Q$, $D \equiv [\mathcal{Q}, \mathcal{QV}, (\lambda \pi.PL)Q]$, $E \equiv [\mathcal{Q}', \mathcal{QV}', ((\lambda \pi.P')Q)L]$, $\alpha = \mathsf{r.cm}$,
where $[\mathcal{Q}, \mathcal{QV}, P] \to_\beta [\mathcal{Q}', \mathcal{QV}', P']$.
Clearly, $[\mathcal{Q}, \mathcal{QV}, (\lambda x.PL)Q] \to_{\mathsf{r.cm}} [\mathcal{Q}', \mathcal{QV}', (\lambda x.P'L)Q]$ and $[\mathcal{Q}', \mathcal{QV}', ((\lambda \pi.P')Q)L] \to_\beta [\mathcal{Q}', \mathcal{QV}', (\lambda \pi.P'L)Q]$.
- $N \equiv (\lambda \pi.P)Q$, $D \equiv [\mathcal{Q}, \mathcal{QV}, (\lambda x.PL)Q]$, $E \equiv [\mathcal{Q}', \mathcal{QV}', ((\lambda \pi.P)Q')L]$, $\alpha = \mathsf{r.cm}$,
where $[\mathcal{Q}, \mathcal{QV}, Q] \to_\beta [\mathcal{Q}', \mathcal{QV}', Q']$.
Clearly, $[\mathcal{Q}, \mathcal{QV}, (\lambda x.PL)Q] \to_{\mathsf{r.cm}} [\mathcal{Q}', \mathcal{QV}', (\lambda x.PL)Q']$ and $[\mathcal{Q}', \mathcal{QV}', ((\lambda \pi.P)Q')L] \to_\beta [\mathcal{Q}', \mathcal{QV}', (\lambda \pi.PL)Q']$.
- $N \equiv (\lambda \pi.P)Q$, $D \equiv [\mathcal{Q}, \mathcal{QV}, (\lambda x.PL)Q]$, $E \equiv [\mathcal{Q}', \mathcal{QV}', ((\lambda \pi.P)Q)L']$, $\alpha = \mathsf{r.cm}$,
where $[\mathcal{Q}, \mathcal{QV}, L] \to_\beta [\mathcal{Q}', \mathcal{QV}', L']$.
Clearly, $[\mathcal{Q}, \mathcal{QV}, (\lambda x.PL)Q] \to_{\mathsf{r.cm}} [\mathcal{Q}', \mathcal{QV}', (\lambda x.PL')Q]$ and $[\mathcal{Q}', \mathcal{QV}', ((\lambda \pi.P)Q)L'] \to_\beta [\mathcal{Q}', \mathcal{QV}', (\lambda \pi.PL')Q]$.
- $N \equiv (\lambda \pi.P)$, $L \equiv (\lambda x.Q)R$, $D \equiv [\mathcal{Q}, \mathcal{QV}, (\lambda x.NQ)R]$, $E \equiv [\mathcal{Q}, \mathcal{QV}, N(Q\{R/x\})]$,
$\alpha = \mathsf{l.cm}$, $\beta = \mathsf{l}.\beta$.
Clearly, $[\mathcal{Q}, \mathcal{QV}, (\lambda x.NQ)R] \to \mathsf{l}.\beta[\mathcal{Q}, \mathcal{QV}, N(Q\{R/x\})]$.

M cannot be in the form $\mathtt{new}(c)$, because in that case $D \equiv E$. □

Even in absence of types, we cannot build an infinite sequence of commuting reductions:

Lemma 3. The relation $\to_{\mathscr{O}}$ is strongly normalizing. In other words, there cannot be any infinite sequence $C_1 \to_{\mathscr{O}} C_2 \to_{\mathscr{O}} C_3 \to_{\mathscr{O}} \ldots$.

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 $|M|_\lambda \leq |M|^2$. Moreover, if $[\mathcal{Q}, \mathcal{QV}, M] \to_{\mathscr{O}} [\mathcal{Q}, \mathcal{QV}, M']$, then $|M'| = |M|$ but $|M'|_\lambda > |M|_\lambda$. This concludes the proof. □

Finally, we can prove the main results of this section:

Theorem 2 (Unicity of Normal Forms). Any configuration C has at most one normal form.

Proof. If C is a configuration and D and E are distinct normal forms for C, we can iteratively apply Proposition 3 obtaining a configuration F such that both $D \xrightarrow{*} F$ and $E \xrightarrow{*} F$. This however, is a contradiction. □

Since a very strong notion of confluence holds here, strong normalization and weak normalization are equivalent properties of configurations:

Theorem 3. *C is strongly normalizing iff C is weakly normalizing.*

5. Examples

We give now some simple examples in order to show how to compute with Q when the length of the input is fixed. In section 7.2 we will show in detail how to code (infinite) circuit families.

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

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

the term **M** takes two qubits in input and then gives as output an EPR (entangled) state.

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

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

After some new reductions, two quantum variables p and q appear in the term, while the quantum register is modified accordingly. Finally, unitary operators corresponding to **cnot** and **H** are applied to the quantum register.

The final 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.

Deutsch's Algorithm

Deutsch's algorithm is the first quantum algorithm that has been defined. It has interesting applications: for example it allows to compute a global property of a function by combining results from two components of a superposition. We refer here to the *Deutsch's Algorithm* as presented in (Nielsen & Chuang 2000), pages 32 and 33 (a detailed explanation of the algorithm is outside the scope of this paper).

Let $\mathbf{W}_f$ be the unitary transform s.t. $\mathbf{W}_f|c_1c_2\rangle = |c_1, c_2 \oplus f(c_1)\rangle$ (for a given boolean function f), and let $\mathbf{H}$ be the Hadamard transform.

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

$$\mathbf{D} \equiv \lambda\langle x, y\rangle.((\lambda\langle w, z\rangle.\langle \mathbf{H}w, z\rangle)(\mathbf{W}_f\langle \mathbf{H}x, \mathbf{H}y\rangle)).$$

The concern of the Deutsch's algorithm is to use *quantum parallelism* and *interference* 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:

$$[1, \mathbf{D}\langle \mathtt{new}(0), \mathtt{new}(1)\rangle]$$

$$
\begin{aligned}
& [1, \mathbf{D}\langle \mathtt{new}(0), \mathtt{new}(1)\rangle] \\
\to^2_{\mathsf{new}} \quad & [|p \mapsto 0\rangle \otimes |q \mapsto 1\rangle, (\lambda\langle x, y\rangle(\lambda\langle w, z\rangle.\langle \mathbf{H}w, z\rangle)(\mathbf{W}_f\langle \mathbf{H}x, \mathbf{H}y\rangle))\langle p, q\rangle] \\
\to_{\mathsf{q}.\beta} \quad & [|p \mapsto 0\rangle \otimes |q \mapsto 1\rangle, (\lambda\langle w, z\rangle.\langle \mathbf{H}w, z\rangle)(\mathbf{W}_f\langle \mathbf{H}p, \mathbf{H}q\rangle)] \\
\to_{\mathsf{Uq}} \quad & [\frac{|p \mapsto 0\rangle + |p \mapsto 1\rangle}{\sqrt{2}} \otimes |q \mapsto 1\rangle, (\lambda\langle w, z\rangle.\langle \mathbf{H}w, z\rangle)(\mathbf{W}_f\langle p, \mathbf{H}q\rangle)] \\
\to_{\mathsf{Uq}} \quad & [\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 \mathbf{H}w, z\rangle)(\mathbf{W}_f\langle p, q\rangle)] \\
= \quad & [\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 \mathbf{H}w, z\rangle)(\mathbf{W}_f\langle p, q\rangle)] \\
\to_{\mathsf{Uq}} \quad & [\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}, \\
& (\lambda\langle w, z\rangle.\langle \mathbf{H}w, z\rangle)\langle p, q\rangle)] \\
\to_{\mathsf{q}.\beta} \quad & [\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}, \\
& \langle \mathbf{H}p, q\rangle] \\
\to_{\mathsf{Uq}} \quad & [\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} + \\
& \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]
\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.

Exchange

Consider the following lambda term, written in Q's syntax:

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

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

$$\begin{array}{rll}
 & [1, \mathbf{L}\ \langle \mathtt{new}(1), \mathtt{new}(0)\rangle] & \\
\xrightarrow{2} & [|p \mapsto 1\rangle \otimes |q \mapsto 0\rangle, (\lambda\langle x, y\rangle.(\lambda\langle a, b\rangle.\mathbf{cnot}\langle b, a\rangle)((\lambda\langle w, z\rangle.\mathbf{cnot}\langle z, w\rangle)(\mathbf{cnot}\langle x, y\rangle))\langle p, q\rangle] & (1) \\
\rightarrow_{\mathsf{q}.\beta} & [|p \mapsto 1\rangle \otimes |q \mapsto 0\rangle, (\lambda\langle a, b\rangle.\mathbf{cnot}\langle b, a\rangle)((\lambda\langle w, z\rangle.\mathbf{cnot}\langle z, w\rangle)\mathbf{cnot}\langle p, q\rangle])] & \\
\rightarrow_{\mathsf{Uq}} & [|p \mapsto 1\rangle \otimes |q \mapsto 1\rangle, (\lambda\langle a, b\rangle.\mathbf{cnot}\langle b, a\rangle)((\lambda\langle w, z\rangle.\mathbf{cnot}\langle z, w\rangle)\langle p, q\rangle)] & \\
\rightarrow_{\mathsf{q}.\beta} & [|p \mapsto 1\rangle \otimes |q \mapsto 1\rangle, (\lambda\langle a, b\rangle.\mathbf{cnot}\langle b, a\rangle)\mathbf{cnot}\langle q, p\rangle] & \\
\rightarrow_{\mathsf{Uq}} & [|p \mapsto 0\rangle \otimes |q \mapsto 1\rangle, (\lambda\langle a, b\rangle.\mathbf{cnot}\langle b, a\rangle)\langle q, p\rangle] & \\
\rightarrow_{\mathsf{q}.\beta} & [|p \mapsto 0\rangle \otimes |q \mapsto 1\rangle, (\mathbf{cnot}\langle p, q\rangle)] & \\
\rightarrow_{\mathsf{Uq}} & [|p \mapsto 0\rangle \otimes |q \mapsto 1\rangle, \langle p, q\rangle] & (2)
\end{array}$$

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

6. Standardizing Computations

One of the most interesting properties of Q is the capability of performing computational steps in the following order:

- First perform classical reductions.
- Secondly, perform reductions that build the underlying quantum register.
- Finally, perform quantum reductions.

In this section, we provide a *standardization theorem*, that strengthens the common idea that a universal quantum computer should consists of a classical device "setting up" a quantum circuit that is then fed with an input.

We distinguish three particular subsets of $\mathscr{L}$, namely $\mathscr{Q} = \{\mathsf{Uq}, \mathsf{q}.\beta\}$, $n\mathscr{C} = \mathscr{Q} \cup \{\mathsf{new}\}$, and $\mathscr{C} = \mathscr{L} - n\mathscr{C}$. Let $C \rightarrow_\alpha C'$ 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*.

Definition 8. A configuration C is called *non classical* if $\alpha \in n\mathscr{C}$ whenever $C \rightarrow_\alpha C'$. Let NCL be the set of non classical configurations. A configuration C is called *essentially quantum* if $\alpha \in \mathscr{Q}$ whenever $C \rightarrow_\alpha C'$. Let EQT be the set of essentially quantum configurations.

Before claiming the standardization theorem, we need the following definition:

Definition 9. A CNQ computation starting with a configuration C is a computation $\{C_i\}_{i<\varphi}$ such that $C_0 \equiv C$, $\varphi \leq \omega$ 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 $C_i \rightarrow_{\mathscr{Q}} C_{i+1}$.

More informally, a CNQ computation is a computation such that any new reduction is always performed after any classical reduction and any quantum reduction is always performed after any new reduction.

NCL is closed under new reduction, while EQT is closed under quantum reduction:

Lemma 4. If $C \in \mathsf{NCL}$ and $C \rightarrow_{\mathsf{new}} D$ then $D \in \mathsf{NCL}$.

Lemma 5. If $C \in \mathsf{EQT}$ and $C \rightarrow_{\mathscr{Q}} D$ then $D \in \mathsf{EQT}$.

This way we are able to state and prove the Standardization Theorem.

Theorem 4 (Standardization). For every computation $\{C_i\}_{i<\varphi}$ such that $\varphi \in \mathbb{N}$ there is a CNQ computation $\{C'_i\}_{i<\xi}$ such that $C_0 \equiv C'_0$ and $C_{\varphi-1} \equiv C'_{\xi-1}$.

Proof. We build a CNQ computation in three steps:
1. Let us start to reduce $C'_0 \equiv C_0$ *by using* $\mathscr{C}$ reductions as much as possible. By Theorem 3 we must obtain a finite reduction sequence $C'_0 \rightarrow_{\mathscr{C}} \ldots \rightarrow_{\mathscr{C}} C'_k$ s.t. $0 \leq k < \varphi$ and no $\mathscr{C}$ reductions are applicable to C'_k
2. Reduce C'_k *by using* new reductions as much as possible. By Theorem 3 we must obtain a finite reduction sequence $C'_k \rightarrow_{\mathsf{new}} \ldots \rightarrow_{\mathsf{new}} C'_j$ s.t. $k \leq j < \varphi$ and no new reductions are applicable to C'_j. Note that by Lemma 4 such reduction steps cannot generate classical redexes and in particular no classical redex can appear in C'_j.
3. Reduce C'_j *by using* $\mathscr{Q}$ reductions as much as possible. By Theorem 3 we must obtain a finite reduction sequence $C'_j \rightarrow_{\mathscr{Q}} \ldots \rightarrow_{\mathscr{Q}} C'_m$ such that $j \leq m < \varphi$ and no $\mathscr{Q}$ reductions are applicable to C'_m. Note that by Lemma 5 such reduction steps cannot generate neither $\mathscr{C}$ redexes nor new redexes and in particular neither $\mathscr{C}$ nor new reductions are applicable to C'_m. Therefore C'_m *is in normal form.*

The reduction sequence $\{C'_i\}_{i<m+1}$ is such that $C'_0 \rightarrow_{\mathscr{C}} \ldots \rightarrow_{\mathscr{C}} C'_k \rightarrow_{\mathsf{new}} \ldots \rightarrow_{\mathsf{new}} C'_j \rightarrow_{\mathscr{Q}} \ldots \rightarrow_{\mathscr{Q}} C'_m$ is a CNQ computation. By Theorem 2 we observe that $C_{\varphi-1} \equiv C'_m$, which implies the thesis. □

The *intuition* behind a CNQ computation is the following: the first phase of the computation is responsible for the construction of a λ–term (abstractly) representing a quantum circuit and does not touch the underlying quantum register. The second phase builds the quantum register without introducing any superposition. The third phase corresponds to proper quantum computation (unitary operators are applied to the quantum register, possibly introducing superposition). This intuition will become a technical recipe in order to prove a side of the equivalence between Q and quantum circuit families formalism (see Section 7.3).

We conclude by examining the case of *non terminating computations*. From a quantum point of view, 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 (see also Section 8 for a discussion on the measurement).

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.

Lemma 6. 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 $[\mathcal{Q}, \mathcal{QV}, M] \rightarrow_{n\mathscr{C}} [\mathcal{Q}, \mathcal{QV}, M']$ then $|M'| < |M|$ and conclude. □

As a trivial consequence of the Lemma we have that

Proposition 4. Any infinite CNQ computation only includes classical reduction steps.

Finally we can state the theorem:

Theorem 5 (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 6 and Theorem 3. □

7. Expressive Power

In this section we study the expressive power of Q, showing that it is equivalent to finitely generated quantum circuit families, and consequently (via the result of Ozawa and Nishimura (Nishimura & Ozawa 2002)) we have the equivalence with quantum Turing machines as defined by Bernstein and Vazirani (Bernstein & Vazirani 1997). The fact the considered class of circuit families only contains finitely generated ones is not an accident: if we want to represent an entire family by one single lambda term (which is, by definition, a finite object) we must restrict to families which are generated by a discrete set of gates.

Before going into the details, an informal description of how our encoding works is in order. Data will be encoded using some variations on Scott's numerals (Wadsworth 1980). These can be used both for classical and quantum data. In the latter case, a more strict linear discipline (quantum bits cannot be erased, in general) is enforced through a slightly different encoding. Our analysis will concentrate on terms in Q satisfying a simple constraint: when applied to a list of classical bits, they produce a list of quantum variables. These are the *quantum relevant* terms. What is crucial from a computational point of view is the way a quantum relevant term modifies the underlying quantum register.

7.1. Q and the Lambda Calculus

The careful reader might be tempted to believe that since the usual pure, untyped lambda calculus can be embedded in Q, the encoding of circuit families into Q should be very easy. The situation, however, is slightly more complicated.

It's true that Girard's encodings of intuitionistic logic into linear logic can be somehow generalized to translations from pure, untyped, lambda calculus to untyped linear lambda terms, like the ones of Q (see, for example, (Wadler 1994)). Beta reduction in the lambda calculus, however, does *not* correspond to surface reduction in Q. Take, for example, the lambda term $M \equiv x((\lambda y.yy)(\lambda y.yy))$: it is not normalizable, but its (call-by-name) translation $\overline{M} \equiv x!((\lambda !y.y!y)!(\lambda !y.y!y))$ is clearly a normal form in Q. There are some connections between weak head reduction in the lambda calculus and surface reduction in Q: if M rewrites to N by weak head reduction, then $\overline{M}$ rewrites to $\overline{N}$ in Q. The converse is not true: $M = \lambda x.((\lambda y.yy)(\lambda y.yy))$ is a weak head normal form, but $\overline{M}$ is not normalizable in the Q. Similar considerations hold for weak call-by-value reduction when the translation function $\overline{(\cdot)}$ is the one induced by the embedding $A \to B \equiv !(A \multimap B)$. On the other hand, lambda calculus is Turing complete for any decent encoding of natural numbers into it. This holds for Scott numerals, for example. But does this correspondence scale down to more restricted notions of reduction, like weak head reduction?

Even if you manage to give a positive answer to the above question, that would not be the end of the story. If Q is proved to have the classical expressive power of Turing machines, this simply implies you could compute the *code* D_n of the n-th circuit C_n of any quantum circuit family from input n. But D_n is nothing but a natural number, the "Gödel number" of C_n. Since you want to evaluate C_n inside Q, you 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 Scott numerals, this is *not* a consequence of the alleged (classical) Turing completeness of Q.

For these reasons, we have decided to show the encoding of Quantum circuit families into Q in full detail. This is the subject of Section 7.2.

7.2. *Encoding Quantum Circuits Families*

In this Section we will show that each (finitely generated) quantum circuit family can be captured by a *quantum relevant* term.

7.2.1. *On the Classical Strength of the Q.*

Natural numbers are encoded as Q terms as follows:

$$\begin{array}{rrcl} & \lfloor 0 \rfloor & = & !\lambda !x.\lambda !y.y \\ \forall n & \lfloor n+1 \rfloor & = & !\lambda !x.\lambda !y.x\lfloor n \rfloor \end{array}$$

This way, we can compute the successor and the predecessor of a natural number as follows:

$$\begin{array}{rcl} \mathsf{succ} & = & \lambda z.!\lambda !x.\lambda !y.xz \\ \mathsf{pred} & = & \lambda !z.z!(\lambda x.x)!\lfloor 0 \rfloor \end{array}$$

Indeed:

$$\begin{array}{rcl}
\mathsf{succ}\ \lfloor n\rfloor & \to_{\mathscr{C}} & !\lambda!x.\lambda!y.x\lfloor n\rfloor \equiv \lfloor n+1\rfloor;\\
\mathsf{pred}\ \lfloor 0\rfloor & \to_{\mathscr{C}} & (\lambda!x.\lambda!y.y)!(\lambda x.x)!\lfloor 0\rfloor \to^{*}_{\mathscr{C}} \lfloor 0\rfloor;\\
\mathsf{pred}\ \lfloor n+1\rfloor & \to_{\mathscr{C}} & (\lambda!x.\lambda!y.x\lfloor n\rfloor)!(\lambda x.x)!\lfloor 0\rfloor \to_{\mathscr{C}} (\lambda x.x)\lfloor n\rfloor\\
& \to_{\mathscr{C}} & \lfloor n\rfloor
\end{array}$$

The following terms are very useful when writing definitions by cases:

$$\begin{array}{rcl}
\mathsf{case}^{\mathsf{nat}}_{0} & \equiv & \lambda!x.\lambda!y_0.\lambda!z.x!(\lambda!w.z)!y_0\\
\mathsf{case}^{\mathsf{nat}}_{n+1} & \equiv & \lambda!x.\lambda!y_0.\ldots.\lambda!y_{n+1}.\lambda!z.x!(\lambda!w.\mathsf{case}^{\mathsf{nat}}_{n}w!y_1\ldots!y_{n+1}!z)!y_0
\end{array}$$

They behave as follows:

$$\begin{array}{lrcl}
\forall m \leq n & \mathsf{case}^{\mathsf{nat}}_{n}\ \lfloor m\rfloor!M_0\ldots!M_n!N & \to^{*}_{\mathscr{C}} & M_m\\
\forall m > n & \mathsf{case}^{\mathsf{nat}}_{n}\ \lfloor m\rfloor!M_0\ldots!M_n!N & \to^{*}_{\mathscr{C}} & N
\end{array}$$

We can capture *linear lists*, too: 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:

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

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

$$\begin{array}{rcl}
\mathsf{cons} & = & \lambda z.\lambda w.\lambda!x.\lambda!y.xzw;\\
\mathsf{sel} & = & \lambda x.\lambda y.\lambda z.xyz.
\end{array}$$

They behave as follows on lists:

$$\begin{array}{rcl}
\mathsf{cons}\ M[M_1,\ldots,M_n] & \to^{*}_{\mathscr{C}} & [M,M_1,\ldots,M_n]\\
\mathsf{sel}\ []!N!L & \to^{*}_{\mathscr{C}} & L\\
\mathsf{sel}\ [M,M_1,\ldots,M_n]!N!L & \to^{*}_{\mathscr{C}} & NM[M_1,\ldots,M_n]
\end{array}$$

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

$$\begin{array}{lrcl}
 & \mathsf{append}_n[N_1,\ldots,N_m]M_1\ldots M_n & \to^{*}_{\mathscr{C}} & [M_1,\ldots,M_n,N_1,\ldots,N_m]\\
\forall m \leq n & \mathsf{extract}_n M[N_1,\ldots,N_m] & \to^{*}_{\mathscr{C}} & M[]N_mN_{m-1}\ldots N_1\\
\forall m > n & \mathsf{extract}_n M[N_1,\ldots N_m] & \to^{*}_{\mathscr{C}} & M[N_{n+1}\ldots N_m]N_nN_{n-1}\ldots N_1
\end{array}$$

Terms append_n can be built by induction on n:

$$\begin{array}{rcl}
\mathsf{append}_0 & = & \lambda x.x\\
\mathsf{append}_{n+1} & = & \lambda x.\lambda y_1.\ldots.\lambda y_{n+1}.\mathsf{cons}\ y_1(\mathsf{append}_n x y_2\ldots y_{n+1})
\end{array}$$

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

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

The encodings of natural numbers and lists are similar and are both in the style of the so-called Scott's numerals (Wadsworth 1980). However, there is an essential difference between the two:

- Natural numbers are encoded *non-linearly*: any natural number is duplicable by construction, since it has the shape $!M$ for some M.
- Lists are encoded *linearly*: 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.

We need *recursion and iteration*, in order to be able to build-up terms in a functional-programming style. The term rec is defined as $\mathsf{rec}_{\mathsf{aux}}!\mathsf{rec}_{\mathsf{aux}}$, where

$$\mathsf{rec}_{\mathsf{aux}} \equiv \lambda!x.\lambda!y.y!((x!x)!y).$$

For each term M,

$$\mathsf{rec}!M \rightarrow^*_{\mathscr{C}} M!(\mathsf{rec}!M)$$

This will help us in encoding algorithms via recursion. Structural recursion over natural numbers is available through $\mathsf{rec}^{\mathsf{nat}} \equiv \mathsf{rec}!\mathsf{rec}^{\mathsf{nat}}_{\mathsf{aux}}$, where

$$\mathsf{rec}^{\mathsf{nat}}_{\mathsf{aux}} \equiv \lambda!x.\lambda y.\lambda!w.\lambda!z.y!(\lambda!v.w!(x!v!w!z)!v)!z$$

Indeed:

$$\begin{array}{rcl} \mathsf{rec}^{\mathsf{nat}}\ \lfloor 0 \rfloor!M!N & \rightarrow^*_{\mathscr{C}} & \mathsf{rec}^{\mathsf{nat}}_{\mathsf{aux}}\ !(\mathsf{rec}^{\mathsf{nat}})\lfloor 0 \rfloor!M!N \\ & \rightarrow^*_{\mathscr{C}} & (\lambda!x.\lambda!y.y)!(\lambda!v.M!(\mathsf{rec}^{\mathsf{nat}}\ !v!M!N)!v)!N \\ & \rightarrow^*_{\mathscr{C}} & N \\ \mathsf{rec}^{\mathsf{nat}}\ \lfloor n+1 \rfloor!M!N & \rightarrow^*_{\mathscr{C}} & (\lambda!x.\lambda!y.x\lfloor n \rfloor)!(\lambda!v.M!(\mathsf{rec}^{\mathsf{nat}}\ !v!M!N)!v)!N \\ & \rightarrow^*_{\mathscr{C}} & (\lambda!v.M!(\mathsf{rec}^{\mathsf{nat}}\ !v!M!N)!v)!\lfloor n \rfloor \\ & \rightarrow^*_{\mathscr{C}} & M!(\mathsf{rec}^{\mathsf{nat}}\ \lfloor n \rfloor!M!N)!\lfloor n \rfloor \end{array}$$

Iteration is available on lists, too. Let $\mathsf{iter}^{\mathsf{list}} \equiv \mathsf{rec}!\mathsf{iter}^{\mathsf{list}}_{\mathsf{aux}}$, where

$$\mathsf{iter}^{\mathsf{list}}_{\mathsf{aux}} \equiv \lambda!x.\lambda y.\lambda!w.\lambda!z.y!(\lambda v.\lambda u.w(xu!w!z)v)!z$$

Indeed:

$$\begin{array}{rcl} \mathsf{iter}^{\mathsf{list}}\ []!M!N & \rightarrow^*_{\mathscr{C}} & \mathsf{iter}^{\mathsf{list}}_{\mathsf{aux}}\ !(\mathsf{iter}^{\mathsf{list}})[]!M!N \\ & \rightarrow^*_{\mathscr{C}} & []!(\lambda v.\lambda u.M(\mathsf{iter}^{\mathsf{list}}\ u!M!N)v)!N \\ & \rightarrow^*_{\mathscr{C}} & N \\ \mathsf{iter}^{\mathsf{list}}\ [L, L_1, \ldots, L_n]!M!N & \rightarrow^*_{\mathscr{C}} & [L, L_1, \ldots, L_n]!(\lambda v.\lambda u.M(\mathsf{iter}^{\mathsf{list}}\ u!M!N)v)!N \\ & \rightarrow^*_{\mathscr{C}} & (\lambda v.\lambda u.M(\mathsf{iter}^{\mathsf{list}}\ u!M!N)v)L[L_1, \ldots, L_n] \\ & \rightarrow^*_{\mathscr{C}} & M(\mathsf{iter}^{\mathsf{list}}[L_1, \ldots, L_n]!M!N)L \end{array}$$

Definition 10. A function $f : \mathbb{N}^n \to \mathbb{N}$ is *representable* iff there is a term M_f such that:

- Whenever $M_f\lfloor m_1\rfloor \dots \lfloor m_n\rfloor$ has a normal form N (with respect to $\to^*_{\mathscr{C}}$), then $N \equiv \lfloor m\rfloor$ for some natural number m.
- $M_f\lfloor m_1\rfloor \dots \lfloor m_n\rfloor \to^*_{\mathscr{C}} \lfloor m\rfloor$ iff $f(m_1, \dots, m_n)$ is defined and equal to m.

As we have already mentioned at the beginning of this Section, the following result cannot is part of the folklore, but it deserves an explicit proof since the reduction relation considered here is not the standard one:

Proposition 5. The class of representable functions coincides with the class of partial recursive functions (on natural numbers).

Proof. Kleene's partial recursive functions can be embedded into Q:

- Constant functions, the successor and projections can be easily encoded.
- The composition $f : \mathbb{N}^m \to \mathbb{N}$ of $h : \mathbb{N}^n \to \mathbb{N}$ and $g_1, \dots, g_n : \mathbb{N}^m \to \mathbb{N}$ can be represented as follows:

$$M_f \equiv \lambda!x_1.\dots.\lambda!x_m.M_h(M_{g_1}!x_1\dots!x_m)\dots(M_{g_n}!x_1\dots!x_m).$$

- The function $f : \mathbb{N}^{n+1} \to \mathbb{N}$ obtained from $h : \mathbb{N}^{n+2} \to \mathbb{N}$ and $g : \mathbb{N}^n \to \mathbb{N}$ by primitive recursion can be represented as follows:

$$M_f \equiv \lambda y.\lambda!x_1.\dots.\lambda!x_n.\mathsf{rec}^{\mathsf{nat}}y!(\lambda z.\lambda w.M_h wz!x_1\dots!x_n)!(M_g!x_1\dots!x_n).$$

- The function $f : \mathbb{N}^n \to \mathbb{N}$ obtained from $g : \mathbb{N}^{n+1} \to \mathbb{N}$ and $g : \mathbb{N}^n \to \mathbb{N}$ by minimization can be represented as follows:

$$M_f \equiv \lambda x_1.\dots.\lambda x_n.\mathsf{rec}!(N_g)\lfloor 0\rfloor x_1\dots x_n$$

where

$$N_g \equiv \lambda!x.\lambda!y.\lambda!x_1.\dots.\lambda!x_n.(M_g!y!x_1\dots!x_n)!(\lambda!z.x(\mathsf{succ}!y)!x_1\dots!x_n)!y.$$

On the other hand, any representable function is trivially partially recursive. □

In this section, we will introduce the class of quantum relevant terms. In the next sections, we will prove that the class of functions which are captured by quantum relevant terms coincides with the class of functions which can be computed by finitely generated quantum circuit families.

Definition 11. Let $\mathscr{S}$ be any subset of $\mathscr{L}$. The expression $C \Downarrow_{\mathscr{S}} D$ means that $C \to^*_{\mathscr{S}} D$ 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 the equivalence between weakly normalizing and strongly normalizing configurations authorize the following definition:

Definition 12. A term M is called *quantum relevant* (shortly, *qrel*) if it is well formed and for each list $![!c_1, ..., !c_n]$ there are a quantum register $\mathcal{Q}$ and a natural number m such that $[1, \emptyset, M![!c_1, ..., !c_n]] \Downarrow [\mathcal{Q}, \{r_1, \dots, r_n\}, [r_1, \dots, r_m]]$.

In other words, a quantum relevant term is the analogue of a pure λ-term representing a function on natural numbers. It is immediate to observe that the class of qrel terms is not recursively enumerable.

We now need to introduce the notion of a (finitely generated) quantum circuit family. This is the computational model which will prove equivalent to Q.

An *n-qubit gate* (or, simply, a *qubit gate*) is a unitary operator $\mathbf{U} : \mathbb{C}^{2^n} \to \mathbb{C}^{2^n}$, while a *$\mathcal{V}$-qubit gate* (where $\mathcal{V}$ is a qvs) is a unitary operator $G : \mathcal{H}(\mathcal{V}) \to \mathcal{H}(\mathcal{V})$. We here work with elementary operators only. 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 $1 \leq i \leq m$:

- $\mathbf{U}_i$ is an n_i-qubit gate in $\mathcal{G}$;
- $r_1^i, \ldots, r_{n_i}^i$ are distinct quantum variables in $\mathcal{V}$.

The $\mathcal{V}$-gate *determined by* a $\mathcal{V}$-circuit

$$\mathbf{K} = U_1, r_1^1, \ldots, r_{n_1}^1, \ldots, U_m, r_1^m, \ldots, r_{n_m}^m$$

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 way we have defined unitary operators allows us to talk about effective encodings of circuits as natural numbers and, as a consequence, about effective enumerations of quantum circuits. 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:

- $f : \mathbb{N} \to \mathbb{N}$ is a computable function;
- $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is a computable function such that $0 \leq g(n, m) \leq n + 1$ whenever $1 \leq m \leq f(n)$;
- $h : \mathbb{N} \to \mathbb{N}$ is a computable function 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 $\mathcal{H}(\{r_1, \ldots, r_{f(n)}\})$, namely

$$U_{\mathbf{K}_{h(n)}}(|r_1 \mapsto c_{g(n,1)}, \ldots, r_{f(n)} \mapsto c_{g(n,f(n))}\rangle).$$

where c_0, c_{n+1} are assumed to be 0 and 1, respectively. A family of circuits (f, g, h) generated by a finite set $\mathcal{G}$ is said to be *finitely generated*.

7.2.2. *The Result.* In this section, we will show that Q is at least as computationally strong as finitely generated quantum circuit families. Our task will not be too difficult, since we already know from Proposition 5 that any recursive function can be represented in Q. As a conseuqnece, we can assume that f, g and h are represetable whenever (f, g, h) is a family of circuits.

The n-th elementary permutation of m elements (where $1 \leq n < m$) 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.

Lemma 7. Any (finite) permutation can be effectively decomposed into a product of elementary permutations.

A term M *computes the n-th elementary permutation on lists* iff for every list $[N_1,\ldots,N_m]$ with $m > n$, $M[N_1,\ldots,N_m] \to_{\mathscr{C}}^{*} [N_1,\ldots,N_{n-1},N_{n+1},N_n,N_{n+2},\ldots,N_m]$.

Lemma 8. There is a term M_{el} such that, for every natural number n, $M_{el}\lfloor n\rfloor$ computes the $n+1$-th elementary permutation on lists.

Proof. For every $n < m$, let ρ_m^n be the n-th elementary permutation of m elements. Observe that $\rho_m^{n+1}(1) = 1$ (whenever $n+1 < m$) and that $\rho_m^{n+1}(i+1) = \rho_{m-1}^n(i)+1$ (whenever $n \leq m$ and $i < m$). M_{el} is the term

$$\lambda x.\mathsf{rec}^{\mathsf{nat}}\ x!N!L$$

where

$$\begin{array}{rcl} N & \equiv & \lambda!y.\lambda!z.\lambda w.\mathsf{extract}_1(\lambda q.\lambda s.\mathsf{append}_1(yq)s)w \\ L & \equiv & \lambda y.\mathsf{extract}_2(\lambda z.\lambda w.\lambda q.\mathsf{append}_2 zwq)y. \end{array}$$

Indeed:

$$\begin{array}{rcl} M_{el}\lfloor 0\rfloor & \to_{\mathscr{C}} & \mathsf{rec}^{\mathsf{nat}}\ \lfloor 0\rfloor!N!L \to_{\mathscr{C}} L \\ L[M_1,\ldots,M_m] & \to_{\mathscr{C}} & \mathsf{extract}_2(\lambda z.\lambda w.\lambda q.\mathsf{append}_2 zwq)[M_1,\ldots,M_m] \\ & \to_{\mathscr{C}}^{*} & (\lambda z.\lambda w.\lambda q.\mathsf{append}_2 zwq)[M_3,\ldots,M_m]M_2M_1 \\ & \to_{\mathscr{C}}^{*} & \mathsf{append}_2[M_3,\ldots,M_m]M_2M_1 \\ & \to_{\mathscr{C}}^{*} & [M_2,M_1,M_3,\ldots,M_m] \equiv [M_{\rho_m^1(1)},\ldots,M_{\rho_m^1(m)}] \\ M_{el}\lfloor n+1\rfloor & \to_{\mathscr{C}} & \mathsf{rec}^{\mathsf{nat}}\ \lfloor n+1\rfloor!N!L \to_{\mathscr{C}} L \\ & \to_{\mathscr{C}}^{*} & N!(\mathsf{rec}^{\mathsf{nat}}\ \lfloor n\rfloor!N!L)\lfloor n\rfloor \\ & \to_{\mathscr{C}} & \lambda w.\mathsf{extract}_1(\lambda q.\lambda s.\mathsf{append}_1((\mathsf{rec}^{\mathsf{nat}}\ \lfloor n\rfloor!N!L)q)s)w \\ & \to_{\mathscr{C}} & \lambda w.\mathsf{extract}_1(\lambda q.\lambda s.\mathsf{append}_1(Pq)s)w \equiv Q \\ Q[M_1,\ldots,M_n] & \to_{\mathscr{C}} & \mathsf{extract}_1(\lambda q.\lambda s.\mathsf{append}_1(Pq)s)[M_1,\ldots,M_n] \\ & \to_{\mathscr{C}}^{*} & (\lambda q.\lambda s.\mathsf{append}_1(Pq)s)[M_2,\ldots,M_m]M_1 \\ & \to_{\mathscr{C}}^{*} & \mathsf{append}_1(P[M_2,\ldots,M_m])M_1 \\ & \to_{\mathscr{C}}^{*} & \mathsf{append}_1([M_{\rho_{m-1}^n(1)+1},\ldots,M_{\rho_{m-1}^n(m-1)+1}])M_1 \\ & \to_{\mathscr{C}}^{*} & [M_1,M_{\rho_{m-1}^n(1)+1},\ldots,M_{\rho_{m-1}^n(m-1)+1}] \equiv [M_{\rho_m^{n+1}(1)},\ldots,M_{\rho_m^{n+1}(m)}] \end{array}$$

This completes the proof. □

Lemma 9. There is a term M_{length} such that, for every list $[!N_1,\ldots,!N_n]$, $M_{length}[!N_1,\ldots,!N_n] \to_{\mathscr{C}}^{*} \lfloor n\rfloor$.

Proof. M_{length} is the term

$$\lambda x.\mathsf{iter}^{\mathsf{list}} x!(\lambda y.\lambda!z.\mathsf{succ}y)!\lfloor 0\rfloor.$$

Indeed:

$$\begin{array}{rcl}
M_{length}[] & \to_{\mathscr{C}} & \mathsf{iter}^{\mathsf{list}}[]!(\lambda y.\lambda!z.\mathsf{succ}y)!\lfloor 0\rfloor \\
& \to_{\mathscr{C}}^{*} & \lfloor 0\rfloor; \\
M_{length}[!N, !N_1, \ldots, !N_n] & \to_{\mathscr{C}} & \mathsf{iter}^{\mathsf{list}}[!N, !N_1, \ldots, !N_n]!(\lambda y.\lambda!z.\mathsf{succ}y)!\lfloor 0\rfloor \\
& \to_{\mathscr{C}}^{*} & (\lambda y.\lambda!z.\mathsf{succ}y)(\mathsf{iter}^{\mathsf{list}}[!N_1, \ldots, !N_n]!(\lambda y.\lambda!z.\mathsf{succ}y)!\lfloor 0\rfloor)!N \\
& \to_{\mathscr{C}}^{*} & (\lambda y.\lambda!z.\mathsf{succ}y)\lfloor n\rfloor!N \\
& \to_{\mathscr{C}}^{*} & \lfloor n+1\rfloor.
\end{array}$$

This completes the proof. □

Lemma 10. There is a term M_{choose} such that for every list $[!N_1, \ldots, !N_m]$:

$$\begin{array}{rrcl}
 & M_{choose}\lfloor 0\rfloor[!N_1, \ldots, !N_m] & \to_{\mathscr{C}}^{*} & !\lfloor 0\rfloor \\
\forall 1 \le n \le m & M_{choose}\lfloor n\rfloor[!N_1, \ldots, !N_m] & \to_{\mathscr{C}}^{*} & !N_n \\
 & M_{choose}\lfloor m+1\rfloor[!N_1, \ldots, !N_m] & \to_{\mathscr{C}}^{*} & !\lfloor 1\rfloor
\end{array}$$

Proof. M_{choose} is the term

$$\lambda x.\lambda y.(\mathsf{iter}^{\mathsf{list}} y!L!P)x$$

where

$$\begin{array}{rcl}
L & \equiv & \lambda z.\lambda!w.\lambda!q.q!(\lambda s.\lambda r.(s!L_{\ge 2}!L_{=1})r)!(L_{=0})z \\
L_{=0} & \equiv & \lambda t.t\lfloor 0\rfloor \\
L_{=1} & \equiv & \lambda t.(\lambda!u.!w)(t\lfloor 0\rfloor) \\
L_{\ge 2} & \equiv & \lambda u.\lambda t.t(\mathsf{succ}\ u) \\
P & \equiv & \lambda!z.z!(\lambda!w.!\lfloor 1\rfloor)!\lfloor 0\rfloor
\end{array}$$

Indeed:

$$\begin{array}{rcl}
M_{choose}\lfloor 0\rfloor[] & \to_{\mathscr{C}}^{*} & (\mathsf{iter}^{\mathsf{list}}[]!L!P)\lfloor 0\rfloor \\
 & \to_{\mathscr{C}}^{*} & P\lfloor 0\rfloor \\
 & \to_{\mathscr{C}}^{*} & (\lambda!x;\lambda!y.y)!(\lambda!w.!\lfloor 1\rfloor)!\lfloor 0\rfloor \to_{\mathscr{C}}^{*} !\lfloor 0\rfloor \\
M_{choose}\lfloor 1\rfloor[] & \to_{\mathscr{C}}^{*} & P\lfloor 1\rfloor \\
 & \to_{\mathscr{C}}^{*} & (\lambda!x;\lambda!y.x\lfloor 0\rfloor)!(\lambda!w.!\lfloor 1\rfloor)!\lfloor 0\rfloor \to_{\mathscr{C}}^{*} !\lfloor 1\rfloor \\
M_{choose}\lfloor n\rfloor[!N, !N_1, \ldots, !N_m] & \to_{\mathscr{C}}^{*} & (\mathsf{iter}^{\mathsf{list}}[!N, !N_1, \ldots, !N_m]!L!P)\lfloor n\rfloor \\
 & \to_{\mathscr{C}}^{*} & L(\mathsf{iter}^{\mathsf{list}}[!N_1, \ldots, !N_m]!L!P)!N\lfloor n\rfloor \\
 & \to_{\mathscr{C}}^{*} & \lfloor n\rfloor!(\lambda!s.\lambda r.(s!L_{\ge 2}!(L_{=1}\{N/w\}))r)!(L_{=0}) \\
 & & \quad (\mathsf{iter}^{\mathsf{list}}[!N_1, \ldots, !N_m]!L!P) \\
 & \equiv & \lfloor n\rfloor!Q!(L_{=0})S
\end{array}$$

where

$$\begin{array}{rcl} Q & \equiv & \lambda!s.\lambda r.(s!L_{\geq 2}!(L_{=1}\{N/w\}))r \\ S & \equiv & \mathsf{iter}^{\mathsf{list}}[!N_1,\ldots,!N_m]!L!P \end{array}$$

Now:

$$\begin{array}{rcl} \lfloor 0\rfloor!Q!(L_{=0})S & \rightarrow^*_{\mathscr{C}} & L_{=0}S \rightarrow^*_{\mathscr{C}} S\lfloor 0\rfloor \rightarrow^*_{\mathscr{C}} !\lfloor 0\rfloor \\ \lfloor 1\rfloor!Q!(L_{=0})S & \rightarrow^*_{\mathscr{C}} & (\lambda!s.\lambda r.(s!L_{\geq 2}!L_{=1}\{N/w\})r)\lfloor 0\rfloor S \\ & \rightarrow^*_{\mathscr{C}} & (\lambda!x.\lambda!y.y)!L_{\geq 2}!(L_{=1}\{N/w\})S \\ & \rightarrow^*_{\mathscr{C}} & L_{=1}\{N/w\}S \\ & \rightarrow^*_{\mathscr{C}} & (\lambda!u.!N)(S!\lfloor 0\rfloor) \\ & \rightarrow^*_{\mathscr{C}} & (\lambda!u.!N)!\lfloor 0\rfloor \rightarrow^*_{\mathscr{C}} !N \\ \lfloor n+2\rfloor!Q!(L_{=0})S & \rightarrow^*_{\mathscr{C}} & (\lambda!s.\lambda r.(s!L_{\geq 2}!L_{=1}\{N/w\})r)\lfloor n+1\rfloor S \\ & \rightarrow^*_{\mathscr{C}} & (\lambda!x.\lambda!y.x\lfloor n\rfloor)!L_{\geq 2}!(L_{=1}\{N/w\})S \\ & \rightarrow^*_{\mathscr{C}} & L_{\geq 2}\lfloor n\rfloor S \\ & \rightarrow^*_{\mathscr{C}} & S\lfloor n+1\rfloor \end{array}$$

This completes the proof. □

We now have all the necessary ingredients to prove that any finitely generated family of circuits can be represented into Q:

Theorem 6. For every finitely generated family of circuits (f,g,h) there is a quantum relevant term $M_{f,g,h}$ such that for each $c_1,\ldots,c_n$, the following two conditions are equivalent

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

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

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

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 $\mathcal{G} = \{U_0,\ldots,U_b\}$ such that for every $i \in \mathbb{N}$ the gates $U^{h(i)}_1,\ldots,U^{h(i)}_{k(i)}$ are all from $\mathcal{G}$. Let $ar(0),\ldots,ar(b)$ the arities of $U_0,\ldots,U_b$. 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 $U^{h(i)}_j$ is U_x. Moreover, we know that for every $i \in \mathbb{N}$ and for every $1 \leq j \leq k(h(i))$, 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 is a permutation 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 $1 \leq x \leq p(h(i),k(h(i)))$. 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. 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 (via Lemma 7).
- A function $q : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that $q(i, j, x) = y$ iff the x-th elementary permutation of $\{1, \ldots, f(h(i))\}$ in which π^i_j can be decomposed is the y-th 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 (via Lemma 7).
- A function $t : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that $t(i, j, x) = y$ iff the x-th elementary permutation of $\{1, \ldots, f(h(i))\}$ in which ρ^i_j can be decomposed is the y-th elementary permutation.

Now, let us build up a term M_{init} that, given a list L of boolean constants and a natural number $\lfloor n \rfloor$, computes the input list for $\mathbf{K}_{h(n)}$ from L.

$$M_{init} \equiv \lambda!x.\lambda!y.\mathsf{rec}^{\mathsf{nat}}(M_f\ !y)!N!([])$$

where

$$N \equiv \lambda w.\lambda z.\mathsf{cons}((\lambda!q.\mathtt{new}(q))(M_{choose}(M_g!y(z))x)w.$$

Moreover, we need another term M_{circ}, that, given a natural number $\lfloor n \rfloor$ computes a term computing the unitary transformations involved in $\mathbf{K}_{h(n)}$ acting on lists of quantum variables with length $f(n)$. The term is:

$$M_{circ} \equiv \lambda!w.\mathsf{rec}^{\mathsf{nat}}(M_k(M_h!w))!(\lambda y.\lambda!z.\lambda q.M_\rho(M_{unit}(M_\pi(yq))))!(\lambda y.y)$$

where

$$\begin{array}{rcl}
M_\pi & \equiv & \mathsf{rec}^{\mathsf{nat}}(M_r!w!z)!(\lambda y.\lambda!x.\lambda t.(M_{el}(M_q!w!z!x))(yt))!(\lambda y.y) \\
M_{unit} & \equiv & \lambda y.(\mathsf{case}^{\mathsf{nat}}_b(M_u!x!w)!N_0 \ldots !N_b!(\lambda z.z))y \\
N_i & \equiv & \lambda y.\mathsf{extract}_{ar(i)}(\lambda z.\lambda x_{ar(i)}.\ldots.\lambda x_1.M_{ar(i)}(U_i\langle x_1, \ldots, x_{ar(i)}\rangle))y \\
M_{ar(i)} & \equiv & \lambda\langle x_1, \ldots, x_{ar(i)}\rangle.\mathsf{append}_{ar(i)}zx_1 \ldots x_{ar(i)} \\
M_\rho & \equiv & \mathsf{rec}^{\mathsf{nat}}(M_s!w!z)!(\lambda y.\lambda!x.\lambda t.(M_{el}(M_t!w!z!x))(yt))!(\lambda y.y)
\end{array}$$

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

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

This concludes the proof. □

7.3. *From Q to Circuits*

We prove here the converse of Theorem 6. This way we will complete the proof of the equivalence with quantum circuit families. We will stay more informal here: the arguments are rather intuitive.

Let M be a qrel term, let $![!c_1, ..., !c_n], ![!d_1, ..., !d_n]$ be two lists of bits (with the same length) and suppose that $[1, M![!c_1, ..., !c_n]] \Downarrow_{n\mathscr{Q}} [\mathcal{Q}, N]$. Clearly, N cannot contain any boolean constant, since M is assumed to be qrel. By applying exactly the same computation steps that lead from $[1, M![!c_1, ..., !c_n]]$ to $[\mathcal{Q}, N]$, we can prove that $[1, M![!d_1, ..., !d_n]] \Downarrow_{n\mathscr{Q}} [\mathcal{Q}', N]$, 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 computation step leading from $[1, M![!c_1, ..., !c_n]]$ to $[\mathcal{Q}, N]$ is effective, i.e. it is intuitively computable (in the classical sense). Therefore, by means of Church-Turing's Thesis we obtain the following:

Proposition 6. For each qrel M there exist 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 $c_1, \ldots, c_n$, $[1, M![!c_1, ..., !c_n] \Downarrow_{n\mathscr{Q}} [|r_1 \mapsto c_{g(n,1)}, \ldots, r_{f(n)} \mapsto c_{g(n,f(n))}\rangle, N]$, where we conventionally set $c_0 \equiv 0$ and $c_{n+1} \equiv 1$.

Let us consider $[\mathcal{Q}, M] \in \mathsf{EQT}$ 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 to build in an effective way a unitary transformation U such that $\mathcal{Q}' = U_{\langle r_1, \ldots, r_m \rangle}(\mathcal{Q})$. Summarizing, we have the following:

Proposition 7. 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, K is generated by gates appearing in M. Furthermore K can be effectively computed from M.

As a direct consequence of propositions 6 and 7 we obtain the following:

Theorem 7. For each qrel M there is a quantum circuit family (f, g, h) such that for each list $c_1, ..., c_n$ the following two conditions are equivalent:

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

Notice that the standardization theorem helps very much here. Without it, we would not be able to assume that all non-quantum reduction steps can be done before any quantum reduction step.

8. On the Measurement Operator

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. This in contrast 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 so-called measurement calculus (Danos et al. 2007) (where the possibility of observing is even more central).

Extending Q with a measurement operator $\mathtt{meas}(\cdot)$ (in the style of λ_{sv}) would not be particularly problematic. However, some of the properties we proved here would not be true anymore. In particular:

- The reduction relation would be probabilistic, since observing a qubit can have different outcomes. As a consequence, confluence would not be true anymore.

- The standardization theorem would not hold in the form it has been presented here. In particular, the application of unitary transformations to the underlying quantum register could not always be postponed until the end of a computation.

The main reason why we have restricted our attention to a calculus without any explicit measurement operator is that the (extensional) expressive power of the obtained calculus (**i.e. the extensional class of quantum computable functions**) would presumably be the same. **More precisely, in (Bernstein & Vazirani 1997), pag.1420, the authors write that *"A priori it is not clear whether multiple observations might increase the power of QTMs. . . . one may assume without loss of generality that a QTM is only observed once. . . "*. See also the so called *"principle of deferred measurement"* (Nielsen & Chuang 2000), pag. 186**, and the discussion in the introduction.

As a consequence, we have adopted the standard point of view followed by papers dealing with quantum computability: *we assume to have a unique implicit measurement at the end of computation.*

However, please notice that the possibility of measuring qubits internally (e.g. by a construct like `meas`$\cdot$) could allow to solve certain problems more efficiently, by exploiting the inherent nondeterminism involved in measurements. Indeed, it is not known whether measurement-based quantum computation can be efficiently simulated by measurement-free quantum computation. However, this interesting question goes well beyond the scope of this paper.

It would be absolutely straightforward to add an *explicit*, *final* and *full* measurement on the quantum register, without any consequence on the previously stated results. Simply add to the calculus the following rule:

$$\frac{[\sum_{i=1}^{n} a_i|f_i\rangle, \mathcal{QV}, M] \in \mathsf{NF}}{[\sum_{i=1}^{n} a_i|f_i\rangle, \mathcal{QV}, M] \rightarrow_{|a_i|^2} f_i}\ \mathsf{measurement}$$

where $[\sum_{i=1}^{n} a_i|f_i\rangle, \mathcal{QV}, M] \rightarrow_{|a_i|^2} f_i$ means that the measurement of the quantum register gives the value f_i with probability $|a_i|^2$.

9. Conclusion and Further Work

We have studied Q, a quantum lambda calculus based on the paradigm "quantum data and classical control". Differently from most of the related literature, which focus on semantical issues, we faced the problem of expressiveness, proving the computational equivalence of our calculus with a suitable class of quantum circuit families (or equivalently, with the quantum Turing machines *à la* Bernstein and Vazirani).

We have also given a standardization theorem, that should help clarifying the interaction between the classical and the quantum world (at least in a λ–calculus setting). Operational properties of the calculus, such as subject reduction and confluence, have been studied.

A next step of our research will concern the development of type systems. An interesting question is the following: is it possible to give type systems controlling the (quantum) computational complexity of representable functions?

Another possible direction of our future research will regard the study of measurement, in order to move from a calculus of computable functions toward a more concrete functional programming language.

References

D. Aharonov & W. van Dam & J. Kempe & Z. Landau & S. Lloyd & O. Regev (2007). 'Adiabatic quantum computation is equivalent to standard quantum computation'. *SIAM J. Comput.* **37**(1):166–194 (electronic).

T. Altenkirch & J. Grattage (2005). 'A functional quantum programming language'. In *Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science*.

P. Arrighi & G. Dowek (2006). 'Linear-algebraic Lambda-calculus: higher-order, encodings and confluence'. *ArXiv.org*. URL: http://www.citebase.org/abstract?id=oai:arXiv.org:quant-ph/0612199.

J.-L. Basdevant & J. Dalibard (2005). *Quantum mechanics*. Springer-Verlag, Berlin. Corrected second printing, With 1 CD-ROM by Manuel Joffre.

E. Bernstein & U. Vazirani (1997). 'Quantum Complexity Theory'. *SIAM J. Comput.* **26**(5):1411–1473.

V. Danos, et al. (2007). *The Measurement Calculus. J. ACM* **54**(2):8–45 (electronic).

D. Deutsch (1985). 'Quantum theory, the Church-Turing principle and the universal quantum computer'. *Proceedings of the Royal Society of London Ser. A* **A400**:97–117.

R. P. Feynman (1981). 'Simulating physics with computers'. *Internat. J. Theoret. Phys.* **21**(6-7):467–488. Physics of computation, Part II (Dedham, Mass., 1981).

Lov K. Grover (1999). 'Quantum search on structured problems'. In *Quantum computing and quantum communications (Palm Springs, CA, 1998)*, vol. 1509 of *Lecture Notes in Comput. Sci.*, pp. 126–139. Springer, Berlin.

S. C. Kleene (1936). 'λ-definability and recursiveness'. *Duke Math. J.* **2**(2):340–353.

E. Knill (1996). 'Conventions for quantum pseudocode'. Tech. Rep. LAUR-96-2724, Los Alamos National Laboratory.

P. Maymin (1996). 'Extending the lambda calculus to express randomized and quantumized algorithms'. Tech. Rep. arXiv:quant-ph/9612052, arXiv.

P. Maymin (1997). 'The lambda-q calculus can efficiently simulate quantum computers'. Tech. Rep. arXiv:quant-ph/9702057, arXiv.

M. A. Nielsen & I. L. Chuang (2000). *Quantum computation and quantum information*. Cambridge University Press, Cambridge.

H. Nishimura & M. Ozawa (2002). 'Computational complexity of uniform quantum circuit families and quantum Turing machines'. *Theor. Comput. Sci.* **276**(1-2):147–181.

S. Perdrix (2005). 'Quantum patterns and types for entanglement and separability'. In P. Selinger (ed.), *Proceedings of the 3rd International Workshop on Quantum Programming Languages*, Electronic Notes in Theoretical Computer Science. Elsevier.

P. Selinger (2004). 'Towards a Quantum Programming Language'. *Math. Structures in Comput. Sci.* **14**(4):527–586.

P. Selinger & B. Valiron (2006). 'A lambda calculus for quantum computation with classical control'. *Math. Structures in Comput. Sci.* **16**(3):527–552.

P. W. Shor (1994). 'Algorithms for quantum computation: discrete logarithms and factoring'. In *35th Annual Symposium on Foundations of Computer Science (Santa Fe, NM, 1994)*, pp. 124–134. IEEE Comput. Soc. Press, Los Alamitos, CA.

A. K. Simpson (2005). 'Reduction in a Linear Lambda-Calculus with Applications to Operational Semantics.'. In *Term rewriting and applications*, Vol. 3467 of *Lecture Notes in Comput. Sci.*, pp. 219–234. Springer, Berlin.

A. van Tonder (2004). 'A lambda calculus for quantum computation'. *SIAM J. Comput.* **33**(5):1109–1135 (electronic).

P. Wadler (1994). 'A syntax for linear logic'. In *Mathematical foundations of programming semantics (New Orleans, LA, 1993)*, vol. 802 of *Lecture Notes in Comput. Sci.*, pp. 513–529. Springer, Berlin.

C. Wadsworth (1980). 'Some unusual λ-calculus numeral systems'. In J. Seldin & J. Hindley (eds.), *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism.* Academic Press.

A. Yao (1993). 'Quantum Circuit Complexity'. In *Proceedings of the 34th Annual Symposium on Foundations of Computer Science*, pp. 352–360, Los Alamitos, California. IEEE Press.

Appendix A. Appendix: Hilbert spaces

Definition 13 (Hilbert Space). An Hilbert space $\mathcal{H}$ is:
a vector space on the field $\mathbb{C}$ equipped with:

1 an *inner product* $\langle\ ,\ \rangle_{\mathcal{H}} : \mathcal{H} \times \mathcal{H} \to \mathbb{C}$ s.t.
   (a) $\langle \phi, \psi \rangle_{\mathcal{H}} = \langle \psi, \phi \rangle^*_{\mathcal{H}}$;
   (b) $\langle \psi, \psi \rangle_{\mathcal{H}}$ is a real number non negative;
   (c) if $\langle \psi, \psi \rangle_{\mathcal{H}} = 0$ then $\psi = \mathbf{0}$
   (d) $\langle c_1\phi_1 + c_2\phi_2, \psi \rangle_{\mathcal{H}} = c_1^* \langle \phi_1, \psi \rangle_{\mathcal{H}} + c_2^* \langle \phi_2, \psi \rangle_{\mathcal{H}}$;
   (e) $\langle \phi, c_1\psi_1 + c_2\psi_2 \rangle_{\mathcal{H}} = c_1 \langle \phi, \psi_1 \rangle_{\mathcal{H}} + c_2 \langle \phi, \psi_2 \rangle_{\mathcal{H}}$.
2 a *norm* $||\ ||_{\mathcal{H}} : \mathcal{H} \to \mathbb{R}^+$ defined by $||v||_{\mathcal{H}} = \langle v, v \rangle^{1/2}_{\mathcal{H}}$;

Given the metric $d(\psi, \phi) = ||\psi - \phi||_{\mathcal{H}}$, the space $\mathcal{H}$ must be complete (all the Cauchy sequences are convergent).

Proposition 8. Each finite dimensional complex vector space $\mathcal{H}$ equipped with an inner product $\langle\ ,\ \rangle_{\mathcal{H}}$ is an Hilbert space w.r.t. the metric $d(\psi, \phi) = ||\psi - \phi||_{\mathcal{H}}$.

In the following, when it is clear from the context, we will write simply $\langle\ ,\ \rangle$ and $||\ ||$ instead of $\langle\ ,\ \rangle_{\mathcal{H}}$ and $||\ ||_{\mathcal{H}}$.

Let $\mathcal{H}$ be a Hilbert space and ϕ, ψ generic vectors of $\mathcal{H}$: $\phi \in$ is *normalized* if $||\phi|| = 1$. ϕ and ψ are *orthogonal* if $\langle \phi, \psi \rangle = 0$.
A set $\mathbf{b} = \{\phi_0, \ldots, \phi_{n-1}\} \subseteq \mathcal{H}$ is a an orthonormal base if:

1 if $\phi \in \mathcal{H}$ then $\phi = \sum_{i=0}^{n-1} d_i \phi_i$ (where each d_i is in $\mathbb{C}$);
2 each $\phi \in \mathbf{b}$ is normalized;
3 if $\phi, \psi \in \mathbf{b}$ and $\phi \neq \psi$ then $\langle \phi, \psi \rangle = 0$.

Definition 14 (Unitary operators). Let $\mathcal{H}'$ ad $\mathcal{H}''$ two finite Hilbert spaces with the same dimension, and let $U : \mathcal{H}' \to \mathcal{H}''$, a linear transform, the *adjoint* of U, is the unique linear transform $U^\dagger : \mathcal{H}'' \to \mathcal{H}'$ such that for all ϕ, ψ $\langle U\phi, \psi\rangle = \langle \phi, U^\dagger\psi\rangle$. If $U^\dagger U = Id$ we say that U is *unitary*. If $\mathcal{H}' = \mathcal{H}''$ then the unitary transform U is called *unitary operator*.

Definition 15 (Tensor of Hilbert Spaces). Let $\mathcal{H}', \mathcal{H}''$ be two Hilbert spaces with inner products $\langle\ ,\ \rangle_{\mathcal{H}'}, \langle\ ,\ \rangle_{\mathcal{H}''}$.
The *tensor product* of $\mathcal{H}'$ and $\mathcal{H}''$ is the Hilbert space $\mathcal{H}' \otimes \mathcal{H}''$ built in the following way. Let $\mathcal{H}' \bullet \mathcal{H}''$ the space freely generated by the set $\mathcal{H}' \times \mathcal{H}''$. Now let us consider the subspace S of $\mathcal{H}' \bullet \mathcal{H}''$ generated by the elements:

$$(d_1\phi_1 + d_2\phi_2, \psi) - d_1(\phi_1, \psi) - d_2(\phi_2, \psi)$$

$$(\phi, d_1\psi_1 + d_2\psi_2, \psi) - d_1(\phi, \psi_1) - d_2(\phi, \psi_2)$$

with $d_1, d_2 \in \mathbb{C}$, $\phi \in \mathcal{H}'$, $\psi \in \mathcal{H}''$.

Let us consider the quotient space $(\mathcal{H}' \bullet \mathcal{H}'')/S$, we define the tensor product of two Hilbert spaces in the following way[§]:

$$\mathcal{H}' \otimes \mathcal{H}'' \stackrel{def}{=} (\mathcal{H}' \bullet \mathcal{H}'')/S.$$

The inner product in $\mathcal{H}' \otimes \mathcal{H}''$ is defined by:

1 $\langle \phi_1 \otimes \psi_1, \phi_2 \otimes \psi_2\rangle_{\mathcal{H}} = \langle \phi_1, \phi_2\rangle_{\mathcal{H}'} \langle \psi_1, \psi_2\rangle_{\mathcal{H}''}$;
2 $\langle c_1\phi_1 + c_2\phi_2, \psi\rangle_{\mathcal{H}} = c_1^*\langle \phi_1, \psi\rangle_{\mathcal{H}} + c_2^*\langle \phi_2, \psi\rangle_{\mathcal{H}}$;
3 $\langle \phi, c_1\psi_1 + c_2\psi_2\rangle_{\mathcal{H}} = c_1\langle \phi, \psi_1\rangle_{\mathcal{H}} + c_2\langle \phi, \psi_2\rangle_{\mathcal{H}}$.

Proposition 9. The map

$$\otimes : \mathcal{H}' \bullet \mathcal{H}'' \to \mathcal{H}' \otimes \mathcal{H}''$$

defined by $(\phi, \psi) \mapsto S + (\phi, \psi)$ is bilinear (S is the subspace defined above).

Proposition 10. Let $\mathcal{H}', \mathcal{H}''$ be two Hilbert spaces with orthonormal bases $\mathbf{b}', \mathbf{b}''$; the set $\{\phi' \otimes \phi'' | \phi' \in \mathbf{b}', \phi'' \in \mathbf{b}''\}$ is an orthonormal base of $\mathcal{H}' \otimes \mathcal{H}''$.

Definition 16. Let $\mathcal{H}'$ and $\mathcal{H}''$ two Hilbert spaces and let U, V be unitary operators respectively in $\mathcal{H}'$ and $\mathcal{H}''$. The unitary operator $U \otimes V$ in $\mathcal{H}' \otimes \mathcal{H}''$ is defined by:

1 $(U \otimes V)(\phi \otimes \psi) = (U\phi) \otimes (V\psi)$
2 $(U \otimes V)(\sum_{i=0}^{k}\ b_i\phi_i) = \sum_{i=0}^{k}\ b_{ij}(U \otimes V)\phi_i$

[§] $(\mathcal{H}' \bullet \mathcal{H}'')/S$ is the quotient space with respect to the cosets $S + (\phi, \psi)$, with $(\phi, \psi) \in \mathcal{H}' \times \mathcal{H}''$