Math. Struct. in Comp. Science: page 1 of 39. © 2009 Cambridge University Press
doi:10.1017/S096012950800741X

On a measurement-free quantum lambda calculus with classical control

UGO DAL LAGO[†], ANDREA MASINI[‡] and MARGHERITA ZORZI[§]

†*Dipartimento di Scienze dell'Informazione, Università di Bologna*
Email: dallago@cs.unibo.it
‡*Dipartimento di Informatica, Università di Verona*
Email: andrea.masini@univr.it
§*Dipartimento di Informatica, Università di Verona*
Email: margherita.zorzi@univr.it

Received 21 May 2007; revised 20 March 2008

We study a measurement-free, untyped λ-calculus with quantum data and classical control. This work arises from previous proposals by Selinger and Valiron, and Van Tonder. We focus on operational and expressiveness issues, rather than (denotational) semantics. We prove subject reduction and confluence, and a standardisation 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 1982). It defines an alternative computational paradigm based on quantum mechanics (Basdevant and Dalibard 2005) rather than digital electronics. The first proposal for a quantum abstract computer is due to Deutsch, who introduced the notion of 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 a number of other quantum computational models, such as measurement-based models (Danos *et al.* 2007) and adiabatic models (Aharonov *et al.* 2007).

The introduction of quantum abstract machines allowed the development of a complexity theory of quantum computation. One of the most important results in quantum complexity theory was obtained by Shor, who showed that integers can be factorised in polynomial time (Shor 1994). We would like to stress also the importance of Grover's algorithm (Grover 1999), which 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 recent years, the impressive results obtained in this area (for example, Shor's fast factoring algorithm) have stimulated the

[†] This author was partially supported by the FIRB grant RBIN04M8S8, 'Intern. Inst. for Applicable Math.'

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 has been 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 quantum computational model) are essentially 'first-order'.

This paper is an attempt to make 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 was (as far as we are aware) in two unpublished papers by Maymin (Maymin 1996; 1997). Later, Selinger rigorously defined a first-order quantum functional language (Selinger 2004). Another interesting proposal within the framework of first-order quantum functional languages is the language QML (Altenkirch and Grattage 2005). Arrighi and Dowek have also recently proposed an interesting extension of λ-calculus with potential applications in the field of quantum computing (Arrighi and Dowek 2008).

Focusing on higher-order functional programming languages, at least two distinct *foundational* proposals have already appeared in the literature: by Selinger and Valiron (Selinger and Valiron 2006) (see also the interesting extension proposed in Perdrix (2007)) and by Van Tonder (van Tonder 2004). These two approaches seem 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; while the approach of Van Tonder *seems*, at first glance, to be based on the idea of putting *arbitrary λ-terms in superposition*. But, *is this the correct picture*? In order to give an answer, we need to examine the two approaches more closely.

Selinger and Valiron's approach. The main goal of Selinger and Valiron's work is to provide the basis for a typed quantum functional language (with types in propositional multiplicative and exponential linear logic). Their idea is to define a language where only data are superposed, and where programs live in a standard classical world. In particular, there is no need for 'exotic' objects such as λ-terms in superposition. The approach is well summarised by the slogan *'classical control + quantum data'*. The proposed calculus, here called 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 whether we can compare the expressive power of λ_{sv} with that of any well-known computational model (for example, 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. However, there are a couple of glaring differences between λ_q and λ_{sv}. In fact, it seems that *by design*,

λ_q *allows arbitrary superpositions of* λ*-terms*. In our opinion the essence of Van Tonder's approach is contained in van Tonder (2004, Lemma 5.1), where it is stated that '*two terms* M *and* 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 that λ_q has classical control too: it is not possible to superimpose terms differing in a marked way, that is, terms with a different computational evolution. Moreover, measurement in λ_q is not internalised, that is, there is no 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), though, clearly, such a criticism does not invalidate the foundational importance of the approach.

Our proposal

In order to avoid possible misunderstandings, we will begin by stressing *what our proposal is not*:

— *We do not propose a new computational paradigm*. Indeed, we adopt the 'quantum data and classical control' paradigm as proposed by Selinger and Valiron (and, implicitly, by Van Tonder); we extensively develop this paradigm showing its potential from a computation-theoretic point of view. Moreover, we start our study with a measurement-free calculus, following Van Tonder.

— *We do not propose a programming language*. Our emphasis is on computability, not on the development of a more or less concrete programming languages (exactly as for pure λ-calculus, which 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 into the 'quantum data and classical control' paradigm in the absence 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 an *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 with respect to term reduction, we have proved a suitable *subject-reduction theorem*. The calculus we introduce here is not endowed with a reduction strategy (it is neither call-by-value nor call-by-name). We prove *confluence*, which holds in a strong sense. Notably, a configuration (the quantum generalisation of a term) is strongly normalising if and only if it is weakly normalising. Another remarkable feature of the calculus is given by the (*quantum*) *standardisation theorem*. Roughly speaking, for each terminating computation there is another 'canonical', equivalent, computation where computation steps are performed in the following order:

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

We think that standardisation 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 Bernstein) and quantum circuit families (remember that the two formalisms have been showed to be equivalent (Nishimura and Ozawa 2008))? In tackling the expressive power problem, we prove the *equivalence between our calculus and quantum circuit families*. As far as we are aware, this is the first time that such a study has been done in a detailed and rigorous way for a quantum λ-calculus. The equivalence proofs are based on the standardisation theorem and on suitable encodings.

On the absence of measurement. A few words on the absence of an explicit measurement operator are in order at this point – see Section 8 for a more detailed discussion. A property of the calculus under consideration is the absence of measurement, in the spirit of other quantum computational models such as quantum circuits and quantum Turing machines. This is obviously a limitation of our system, as 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 performing measurements is a key feature of any (quantum) programming language since it simplifies the writing of programs.

On the other hand, the absence of measurement is a feature of standard quantum computational models, such as quantum Turing machines and quantum circuits. In all such systems there is no indication of how to *directly* code algorithms where classical computational steps depend on measurements (see, for example, the Shor factoring algorithm), not to mention infinite quantum computations, that is, infinite sequences of quantum steps interleaved with 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 computed without performing any explicit measurement. In practice it is possible to make a unique *implicit measurement* at the end of computation – see, for example, Bernstein and Vazirani (1997, page 1420), Nielsen and Chuang (2000, pages 185–187) and the introductory sections of Selinger (2004) and Selinger and Valiron (2006). We have adopted this point of view by *assuming a unique implicit measurement at the end of the computation*. And we will prove that, even in the absence of measurements, a computationally complete calculus can be obtained.

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

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

1 Quantum computability and complexity theory are quite underdeveloped compared with their classical counterparts; in particular, there is almost no result relating classes of (first-order) functions definable in pure and typed λ-calculi to classes of functions from computability and complexity theory (in contrast with classical computability theory (Kleene 1936)).
2 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). These interactions are even stronger in the presence of measurements.

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, we introduce a λ-calculus, called Q, which has classical control and quantum data. The calculus is untyped, but is equipped with well-formation judgments for terms.
— In Section 4, we study Q operationally by means of a suitable formulation of subject reduction and confluence.
— In Section 5, we give some examples of terms, configurations and computations.
— In Section 6, we give a further result on the dynamics of Q by means of a standardisation theorem (as explained above).
— In Section 7, we study in detail the equivalence of Q to quantum circuit families.
— In Section 8, we expand on the above discussion of measurement operators.
— In Section 9, we give conclusions and some suggestions for further work.
— Finally, we recall the basic notions of Hilbert spaces in an Appendix.

2. Mathematical structures

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

2.1. *Quantum computing basics*

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

The states $|0\rangle$ and $|1\rangle$ of a qubit correspond to the boolean constants 0 and 1, which are the only possible values of a classical bit. A qubit, however, can assume other values, different from $|0\rangle$ and $|1\rangle$. In fact, every linear combination $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ where $\alpha, \beta \in \mathbb{C}$, and $|\alpha|^2 + |\beta|^2 = 1$, 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 cannot establish its quantum state with the same precision, 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 $|\beta|^2$.

When defining quantum computational models, we need a generalisation of the notion of a qubit, called a *quantum register* (Nishimura and Ozawa 2008; Selinger 2004; Selinger and Valiron 2006; van Tonder 2004). A quantum register of arity n is a normalised 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.

A key property of quantum registers is that it is not always possible to decompose an n-qubit register into n isolated qubits (mathematically, this means that we are not able to describe the global state as a tensor product of single qubits). These non-decomposable registers are said to be *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 (we lay out some basic notions on Hilbert spaces in the Appendix). As we will see, they allow us 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 such as p, r and q).

Usually, qubits of quantum registers are referred to by means of their ordinal position, but in the approach we propose here 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 $f \in \{0,1\}^{\mathcal{V}}$) such that $\sum_{f\in\{0,1\}^{\mathcal{V}}} |\phi(f)|^2 = 1$ (*normalisation condition*).

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

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

(i) An *inner sum* $+ : \mathscr{H}(\mathscr{V}) \times \mathscr{H}(\mathscr{V}) \to \mathscr{H}(\mathscr{V})$ defined by

$$(\varphi + \psi)(f) = \varphi(f) + \psi(f).$$

(ii) A *multiplication by a scalar* $\cdot : \mathbb{C} \times \mathscr{H}(\mathscr{V}) \to \mathscr{H}(\mathscr{V})$ defined by

$$(c \cdot \varphi)(f) = c \cdot (\varphi(f)).$$

(iii) An *inner product* $\langle \cdot, \cdot \rangle : \mathscr{H}(\mathscr{V}) \times \mathscr{H}(\mathscr{V}) \to \mathbb{C}$ defined by

$$\langle \varphi, \psi \rangle = \sum_{f \in \{0,1\}^{\mathscr{V}}} \varphi(f)^* \psi(f).$$

The space is equipped with the *orthonormal basis* $\mathscr{B}(\mathscr{V}) = \{|f\rangle : f \in \{0,1\}^{\mathscr{V}}\}$[†]. We call such a basis a *standard* basis. For example, the standard basis of the space $\mathscr{H}(\{p,q\})$ 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 $\mathscr{V}' \cap \mathscr{V}'' = \emptyset$. We use $\mathscr{H}(\mathscr{V}') \otimes \mathscr{H}(\mathscr{V}'')$ to denote the tensor product (defined in the usual way) of $\mathscr{H}(\mathscr{V}')$ and $\mathscr{H}(\mathscr{V}'')$. If $\mathscr{B}(\mathscr{V}') = \{|f_i\rangle : 0 \leqslant i < 2^n\}$ and $\mathscr{B}(\mathscr{V}'') = \{|g_j\rangle : 0 \leqslant j < 2^m\}$ are the orthonormal bases of $\mathscr{H}(\mathscr{V}')$ and $\mathscr{H}(\mathscr{V}'')$, respectively, then $\mathscr{H}(\mathscr{V}') \otimes \mathscr{H}(\mathscr{V}'')$ is equipped with the orthonormal basis

$$\{|f_i\rangle \otimes |g_j\rangle : 0 \leqslant i < 2^n, 0 \leqslant j < 2^m\}.$$

We will abbreviate $|f\rangle \otimes |g\rangle$ by $|f, g\rangle$. If $\mathscr{V}$ is a qvs, then $I_{\mathscr{V}}$ is the identity on $\mathscr{H}(\mathscr{V})$, which is clearly unitary. It is easy to show that if $\mathscr{V}' \cap \mathscr{V}'' = \emptyset$, there is a standard *isomorphism*

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

In the rest of this paper we will assume that we are working up to such an isomorphism[‡].

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

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

In particular, if $\mathscr{Q}' \in \mathscr{H}(\mathscr{V}')$ and $\mathscr{Q}'' \in \mathscr{H}(\mathscr{V}'')$ are two quantum registers, with a little abuse of language (authorised by the isomorphism defined above), we will say that $\mathscr{Q}' \otimes \mathscr{Q}''$ is a quantum register in $\mathscr{H}(\mathscr{V}' \cup \mathscr{V}'')$. In the rest of this paper we will use 1 to denote the empty quantum register (which belongs to $\mathscr{H}(\emptyset)$).

Quantum computing is essentially based on the application of unitary operators to quantum registers. A linear operator $\mathbf{U} : \mathscr{H}(\mathscr{V}) \to \mathscr{H}(\mathscr{V})$ is said to be *unitary* if for all

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

[‡] In particular, if $\mathscr{Q} \in \mathscr{H}(\mathscr{V})$, $r \notin \mathscr{V}$ and $|r \mapsto c\rangle \in \mathscr{H}(\{r\})$, we will use $\mathscr{Q} \otimes |r \mapsto c\rangle$ to denote the element $i_s(\mathscr{Q} \otimes |r \mapsto c\rangle) \in \mathscr{H}(\mathscr{V} \cup \{r\})$.

$\phi, \psi \in \mathscr{H}(\mathscr{V})$, $\langle \mathbf{U}(\phi), \mathbf{U}(\psi)\rangle = \langle \phi, \psi\rangle$. The tensor product of unitary operators is defined by $(\mathbf{U} \otimes \mathbf{V})(\phi \otimes \psi) = \mathbf{U}(\phi) \otimes \mathbf{U}(\psi)$.

But which unitary operators are available in Q? Here we assume that unitary operators can be chosen from $\mathscr{U}$, an arbitrary but denumerable fixed set of unitary operators, called *elementary operators*. Clearly, the expressive power of Q depends on this choice. If we want, for example, to capture quantum Turing machines in the style of Bernstein and Vazirani (Bernstein and Vazirani 1997), we could fix $\mathscr{U}$ 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 $\mathscr{U}$ are precisely those (finitely) generated by $\mathscr{U}$.

Let $\mathbf{U} : \mathbb{C}^{2^n} \to \mathbb{C}^{2^n}$ be an elementary operator and let $\langle q_1, \dots, q_n\rangle$ be any sequence of distinguished variables. $\mathbf{U}$ and $\langle q_1, \dots, q_n\rangle$ induce an operator

$$\mathbf{U}_{\langle q_1,\dots,q_n\rangle} : \mathscr{H}(\{q_1, \dots, q_n\}) \to \mathscr{H}(\{q_1, \dots, q_n\})$$

defined as follows: if $|f\rangle = |q_{j_1} \mapsto b_{j_1}, \dots, q_{j_n} \mapsto b_{j_n}\rangle$ is an element of the orthonormal basis of $\mathscr{H}(\{q_1, \dots, q_n\})$, then

$$\mathbf{U}_{\langle q_1,\dots,q_n\rangle}|f\rangle \overset{def}{=} \mathbf{U}|b_{j_1}, \dots, b_{j_n}\rangle.$$

Let $\mathscr{V}' = \{q_{i_1}, \dots, q_{i_k}\} \subseteq \mathscr{V}$. We naturally extend (by suitable standard isomorphisms) the unitary operator $\mathbf{U}_{\langle q_{j_1},\dots,q_{j_k}\rangle} : \mathscr{H}(\mathscr{V}') \to \mathscr{H}(\mathscr{V}')$ to the unitary operator $\mathbf{U}_{\langle\langle q_{j_1},\dots,q_{j_k}\rangle\rangle} : \mathscr{H}(\mathscr{V}) \to \mathscr{H}(\mathscr{V})$ that acts as the identity on variables not in $\mathscr{V}'$ and as $\mathbf{U}_{\langle q_{j_0},\dots,q_{j_k}\rangle}$ on variables in $\mathscr{V}'$.

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 performs no action:

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

Fixing the sequence $\langle p, q\rangle$ of variables, **cnot** induces the operator

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

such that

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

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

3. The syntax of Q

3.1. *A gentle introduction*

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

The proposed quantum λ-calculus is based on the notion of a *configuration* (a reformulation of the concept of a *program state* (Selinger and 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:

1 Configurations can evolve *classically*: the term M changes, but $\mathcal{Q}$ and $\mathcal{QV}$ will not be modified. In other words, reduction will take the 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. This class of reductions includes all the standard λ-reductions (for example, β-reduction).

2 Configurations, however, can also evolve *non-classically*: the term M and the quantum register interact. There are two ways to modify the underlying quantum register:

(a) The *creation of a new quantum bit* by reducing a term `new`(c) (where c is a classical bit). Such a reduction creates a *new quantum variable name* in the underlying term and a new qubit in the underlying 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 the redex `new`(c) with a (fresh) variable name r in M, $\mathcal{Q}'$ is the new quantum register with a new qubit referenced by r and $\mathcal{QV}'$ is simply $\mathcal{QV} \cup \{r\}$.

(b) The application of a *unitary transformation to the quantum register*. This computation step consists of reducing a term $U\langle r_1, \ldots, r_n\rangle$, where U is the name of a unitary operator and $r_1, \ldots, r_n$ are quantum variables. A unitary reduction has the shape

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

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

3.2. *On linearity*

One of the main features of our calculus (and of many other quantum computational models) is linearity, where we use linearity to mean that a term is neither 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. In this way we are able to guarantee that the 'non-cloning and non-erasing' property is 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, that is, terms that do not contain any quantum variable. To this end, the syntax of terms includes a modal operator ! (called the 'bang' operator). The bang operator was introduced in term calculi for linear logic (see, for example, Wadler's syntax (Wadler 1994)) and allows us 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 way to enforce the non-cloning and non-erasing properties. Other solutions have been proposed in the literature, see, for example, Arrighi and Dowek (2008), where it is possible to duplicate base vectors, and Altenkirch and Grattage (2005,) where duplication is modelled by means of sharing.

3.3. *The language of terms*

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

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

We assume we are working modulo variable renaming, that is, *terms are equivalence classes modulo α-conversion*. Substitution up to α-equivalence is defined in the usual way.

We use $\mathbf{Q}(M_1, \dots, M_k)$ to denote the set of quantum variables occurring in $M_1, \dots, M_k$. Note that:

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

— There are also two sorts of constants, namely *boolean constants* (0 and 1) and *unitary operators*: the first 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 $\mathtt{new}(\cdot)$ creates a new qubit when applied to a boolean constant.
— The syntax allows so-called pattern abstraction. A pattern is either a classical variable, a tuple of classical variables, or a 'banged' variable (namely, an expression of the kind $!x$, where x is a name of a classical variable). In order to allow an abstraction of the kind $\lambda !x.M$, the environment (see below) must be enriched with !-patterns, denoting duplicable or erasable variable.

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 no measurement operator in the language – see Section 8 for further discussion of this.

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 a common notation used 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 or x, y, z or the empty set $\emptyset$. Given a classic environment $\Delta = x_1,\ldots,x_n$, we use $!\Delta$ to denote the set of patterns $!x_1,\ldots,!x_n$.
— A *quantum environment* is a (possibly empty) set (denoted by Θ, which may be indexed) of quantum variables. Examples of quantum environments are r_1, r_2, r_3 and the empty set $\emptyset$.
— A *linear environment* is a (possibly empty) set (denoted by Λ, which may be 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, which may be 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, but $x_1, !x_1$ is *not* an environment.
— A *judgement* is an expression $\Gamma \vdash M$, where Γ is an environment and M is a term.

We say that a judgement $\Gamma \vdash M$ is *well formed* (*notation:* $\rhd \Gamma \vdash M$) 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 $i \neq j$, we have $\Lambda_i \cap \Lambda_j = \emptyset$ (note that Λ_i and Λ_j are sets of linear and quantum variables, being linear environments). We use $d \rhd \Gamma \vdash M$ to mean that d is a derivation of the well-formed judgement $\Gamma \vdash M$. If $\Gamma \vdash M$ is well formed, we say also that the term M *is well formed with respect to the environment* Γ. We say that a term M is *well formed* if the judgement $\mathbf{Q}(M) \vdash M$ is well formed.

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

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

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

Fig. 1. Well-forming rules

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

4. Computations

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

— M is a term;
— $\mathscr{QV}$ is a finite quantum variable set such that $\mathbf{Q}(M) \subseteq \mathscr{QV}$;
— $\mathscr{Q} \in \mathscr{H}(\mathscr{QV})$.

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

Definition 3 (Configurations). Two preconfigurations $[\mathscr{Q}, \mathscr{QV}, M]$ and $[\mathscr{Q}', \mathscr{QV}', M']$ are equivalent if and only if there is a bijection $\theta : \mathscr{QV} \to \mathscr{QV}'$ such that $\mathscr{Q}' = \theta(\mathscr{Q})$ and $M' = \theta(M)$. If a preconfiguration C is equivalent to D, we will write $C \equiv D$. The relation $\equiv$ is an equivalence relation. A *configuration* is an equivalence class of preconfigurations modulo the relation $\equiv$. Let $\mathscr{C}$ be the set of configurations.

Remark 1. The way configurations have been defined, namely quotienting preconfigurations over $\equiv$, is very reminiscent of the 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 $\to_\alpha \subseteq \mathscr{C} \times \mathscr{C}$ by means of the rules in Figure 2. Note the presence of two commutative reduction rules (namely $\mathsf{l.cm}$ and $\mathsf{r.cm}$). Since Q is untyped, the role of commutative reductions is not to guarantee that normal forms have certain properties, but rather to prevent quantum reductions from blocking classical ones (see Section 6).

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

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

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

Creation of a new qubit and quantum variable

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

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.

Note 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 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 we should stress that, even though Q is type-free, a set of admissible terms is isolated through well-forming rules, so we need to prove that the class of well-formed terms is closed under reduction.

Quantum 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, note 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.

The Subject Reduction theorem must be given in the following form in order to take into account the introduction of quantum variables during reduction: if $d \triangleright \Gamma \vdash M$ and $[\mathcal{Q}, \mathcal{QV}, M] \rightarrow [\mathcal{Q}', \mathcal{QV}', M']$, then $\triangleright\ \Gamma, \mathcal{QV}' - \mathcal{QV} \vdash M'$ where $\mathcal{QV}' - \mathcal{QV}$ is the (possibly empty) set of quantum variables generated along the reduction. In our example, we have $\triangleright \vdash \mathtt{new}(0)$ and $p \vdash p$ is indeed 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 but easy induction on the derivation of $[\mathcal{Q}, \mathcal{QV}, M] \rightarrow [\mathcal{Q}', \mathcal{QV}', M']$.

For the sake of clarity, we will just show one 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

$$\dfrac{\dfrac{\begin{array}{c} d_1 \\ \vdots \\ \Lambda_1, !\Delta, x \vdash P \end{array}}{\Lambda_1, !\Delta \vdash \lambda x.P}\ \mathsf{lam2} \qquad \begin{array}{c} d_2 \\ \vdots \\ \Lambda_2, !\Delta \vdash N \end{array}}{\Lambda_1, \Lambda_2, !\Delta \vdash (\lambda x.P)N}\ \mathsf{app}$$

Since the reduction does not modify $\mathscr{QV}$, we just have to apply the substitution Lemma (linear case) to d_1 and d_2, obtaining $\triangleright \Lambda_1, \Lambda_2, !\Delta \vdash P\{N/x\}$. $\square$

The following is an immediate consequence (which is trivially provable by induction).

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

The notion of a well-formed judgement can be extended to configurations.

Definition 4. A configuration $[\mathscr{Q}, \mathscr{QV}, M]$ is said to be *well formed* if and only if 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 $[\mathscr{Q}, \mathscr{QV}, M] \overset{*}{\to} [\mathscr{Q}', \mathscr{QV}', M']$, then M' is well formed.

In the following, when we write *configuration* we will mean *well-formed configuration*. We will now define normal forms, configurations and computations.

Definition 5. A configuration $C \equiv [\mathscr{Q}, \mathscr{QV}, M]$ is said to be in *normal form* if and only if there is no D such that $C \to D$. We will use NF to denote the set of configurations in normal form.

We define a computation as a suitable sequence of configurations.

Definition 6. If C_0 is a configuration, a *computation* of length $\varphi \leqslant \omega$ starting with C_0 is a sequence of configurations $\{C_i\}_{i<\varphi}$ such that for all $0 < i < \varphi$, we have $C_{i-1} \to C_i$ and either $\varphi = \omega$ or $C_{\varphi-1} \in \mathsf{NF}$.

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

Proposition 2. Let $\{[\mathscr{Q}_i, \mathscr{QV}_i, M_i]\}_{i<\varphi}$ be a computation such that $\mathbf{Q}(M_0) = \emptyset$. Then for every $i < \varphi$, we have $\mathscr{QV}_i = \mathbf{Q}(M_i)$.

Proof. Observe that if $[\mathscr{Q}, \mathbf{Q}(M), M] \to [\mathscr{Q}', \mathscr{QV}', M']$, then, by inspection of the reduction rules, we immediately have that $\mathscr{QV}' = \mathbf{Q}(M')$ whenever $\mathscr{QV} = \mathbf{Q}(M)$, which gives the required result. $\square$

In the rest of the paper, we will use $[\mathscr{Q}, M]$ to denote the configuration $[\mathscr{Q}, \mathbf{Q}(M), M]$.

4.2. *On the linearity of the calculus: dynamics*

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

reductions cannot be performed under the scope of a bang.

Let us consider the following well-formed configuration: $[1, \emptyset, (\lambda!x.cnot\,\langle x, x\rangle)\,!(\mathtt{new}(1))]$. We can see immediately 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:

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

However, what happens if we are allowed to reduce under the scope of the bang (namely reducing $\mathtt{new}(1)$ before executing the $\mathsf{c}.\beta$-reduction)? We would then obtain the following computation:

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

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

4.3. *Confluence*

Commutative reduction steps behave very differently from 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 $\mathscr{K} = \{\mathsf{r.cm}, \mathsf{l.cm}\}$ and $\mathscr{N} = \mathscr{L} - \mathscr{K}$.

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

First we need to show that whenever $M \to_\alpha N$, the underlying quantum register evolves uniformly.

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 $U_{M,M'} : \mathscr{H}(\mathbf{Q}(M)) \to \mathscr{H}(\mathbf{Q}(M))$ such that $[\mathscr{Q}, \mathscr{QV}, M] \to_\alpha [\mathscr{Q}', \mathscr{QV}', M']$ if and only if $[\mathscr{Q}, \mathscr{QV}, M] \in \mathscr{C}$, $\mathscr{QV}' = \mathscr{QV}$ and $\mathscr{Q}' = (U_{M,M'} \otimes I_{\mathscr{QV}-\mathbf{Q}(M)})\mathscr{Q}$.
2. $\alpha = \mathsf{new}$ and there are a constant c and a quantum variable r such that we have $[\mathscr{Q}, \mathscr{QV}, M] \to_{\mathsf{new}} [\mathscr{Q}', \mathscr{QV}', M']$ if and only if $[\mathscr{Q}, \mathscr{QV}, M] \in \mathscr{C}$, $\mathscr{QV}' = \mathscr{QV} \cup \{r\}$ and $\mathscr{Q}' = \mathscr{Q} \otimes |r \mapsto c\rangle$.

Proof. We use induction on M. M cannot be a variable, a constant, a unitary operator or a term $!N$. If M is an abstraction $\lambda\psi.N$, then $M' \equiv \lambda\psi.N'$, $N \to_\alpha N'$ and the thesis follows from the induction hypothesis. If $M \equiv NL$, we distinguish a number of cases:

— $M' \equiv N'L$ and $N \rightarrow_\alpha N'$.
  The thesis follows from the induction hypothesis.
— $M' \equiv NL'$ and $L \rightarrow_\alpha L'$.
  The thesis follows from the induction hypothesis.
— $N \equiv U$, $L \equiv \langle r_{i_1}, \ldots, r_{i_n} \rangle$ and $M' \equiv \langle r_{i_1}, \ldots, r_{i_n} \rangle$.
  In this case Condition 1 holds. In particular, $\mathbf{Q}(M) = \{r_{i_1}, \ldots, r_{i_n}\}$ and $U_{M,M'} = \mathbf{U}_{\langle\langle r_{i_1},\ldots,r_{i_n}\rangle\rangle}$.
— $N \equiv \lambda x.P$ and $M' = P\{L/x\}$.
  In this case Condition 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 $M' \equiv P\{r_1/x_1, \ldots, r_n/x_n\}$.
  In this case Condition 1 holds and $U_{M,M'} = I_{\mathbf{Q}(M)}$.
— $N \equiv \lambda !x.P$, $L = !Q$ and $M' \equiv P\{Q/x\}$.
  In this case Condition 1 holds and $U_{M,M'} = I_{\mathbf{Q}(M)}$.
— $L \equiv (\lambda\pi.P)Q$ and $M' \equiv (\lambda\pi.NP)Q$.
  In this case Condition 1 holds and $U_{M,M'} = I_{\mathbf{Q}(M)}$.
— $N \equiv (\lambda\pi.P)Q$ and $M' \equiv (\lambda\pi.PL)Q$.
  In this case Condition 1 holds and $U_{M,M'} = I_{\mathbf{Q}(M)}$.

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

Note 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 $[\mathcal{Q}, \mathcal{QV}, M] \rightarrow_\alpha [\mathcal{Q}', \mathcal{QV}', M']$.
1. If $[\mathcal{Q}, \mathcal{QV}, M\{N/x\}] \in \mathcal{C}$, then

$$[\mathcal{Q}, \mathcal{QV}, M\{N/x\}] \rightarrow_\alpha [\mathcal{Q}', \mathcal{QV}', M'\{N/x\}].$$

2. If $[\mathcal{Q}, \mathcal{QV}, M\{r_1/x_1, \ldots, r_n/x_n\}] \in \mathcal{C}$, then

$$[\mathcal{Q}, \mathcal{QV}, M\{r_1/x_1, \ldots, r_n/x_n\}] \rightarrow_\alpha [\mathcal{Q}', \mathcal{QV}', M'\{r_1/x_1, \ldots, r_n/x_n\}].$$

3. If $\triangleright x, \Gamma \vdash N$ and $[\mathcal{Q}, \mathcal{QV}, N\{M/x\}] \in \mathcal{C}$, then

$$[\mathcal{Q}, \mathcal{QV}, N\{M/x\}] \rightarrow_\alpha [\mathcal{Q}', \mathcal{QV}', N\{M'/x\}].$$

Proof. Claims 1 and 2 can be proved by induction on the proof of $[\mathcal{Q}, \mathcal{QV}, M] \rightarrow_\alpha [\mathcal{Q}', \mathcal{QV}', M']$. 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 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 already been pointed out in the literature (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

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

and

$$\begin{aligned}[\mathcal{Q}, \mathcal{QV}, (\lambda\pi.M)((\lambda x.N)L)] &\rightarrow_{\mathcal{K}} [\mathcal{Q}, \mathcal{QV}, (\lambda x.(\lambda\pi.M)N)L] \\ &\rightarrow_{\mathcal{N}} [\mathcal{Q}, \mathcal{QV}, (\lambda\pi.M)(N\{L/x\})].\end{aligned}$$

However, this phenomenon can only occur because of the presence of commutative rules.

Proposition 3 (One-step confluence). Let C, D, E be configurations with $C \rightarrow_\alpha D$, $C \rightarrow_\beta E$ and $D \not\equiv E$. Then:
1. If $\alpha \in \mathcal{K}$ and $\beta \in \mathcal{K}$, then there is F with $D \rightarrow_{\mathcal{K}} F$ and $E \rightarrow_{\mathcal{K}} F$.
2. If $\alpha \in \mathcal{N}$ and $\beta \in \mathcal{N}$, then there is F with $D \rightarrow_{\mathcal{N}} F$ and $E \rightarrow_{\mathcal{N}} F$.
3. If $\alpha \in \mathcal{K}$ and $\beta \in \mathcal{N}$, then either $D \rightarrow_{\mathcal{N}} E$ or there is F with $D \rightarrow_{\mathcal{N}} F$ and $E \rightarrow_{\mathcal{K}} F$.

Proof. Let $C \equiv [\mathcal{Q}, QV, M]$. We use induction on M. We know that M cannot be a variable, a constant or 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

$$\begin{aligned}[\mathcal{Q}, \mathcal{QV}, N] &\rightarrow_\alpha [\mathcal{Q}', \mathcal{QV}', N'] \\ [\mathcal{Q}, \mathcal{QV}, N] &\rightarrow_\beta [\mathcal{Q}'', \mathcal{QV}'', N''].\end{aligned}$$

The induction hypothesis leads easily 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 $C \rightarrow_\alpha D$, $C \rightarrow_\beta E$:

— $D \equiv [\mathcal{Q}', \mathcal{QV}', N'L]$ and $E \equiv [\mathcal{Q}'', \mathcal{QV}'', NL']$
where $[\mathcal{Q}, \mathcal{QV}, N] \rightarrow_\alpha [\mathcal{Q}', \mathcal{QV}', N']$ and $[\mathcal{Q}, \mathcal{QV}, L] \rightarrow_\beta [\mathcal{Q}'', \mathcal{QV}'', L']$.
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 Lemma 1 again, we obtain

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

It is easy to check that $F \equiv G$.

– 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

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

It is easy to check that $F \equiv G$.

– If $\alpha \neq \mathsf{new}$ and $\beta = \mathsf{new}$, we can proceed as in the previous case.

– If $\alpha, \beta \neq \mathsf{new}$, then, by Lemma 1, there exist $\mathscr{QV}'' = \mathscr{QV}' = \mathscr{QV}$, $\mathscr{Q}' = (U_{N,N'} \otimes I_{\mathscr{QV}-\mathbf{Q}(N)})\mathscr{Q}$ and $\mathscr{Q}'' = (U_{L,L'} \otimes I_{\mathscr{QV}-\mathbf{Q}(L)})\mathscr{Q}$. Applying Lemma 1 again, we obtain

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

It is easy to check that $F \equiv G$.

— $D \equiv [\mathscr{Q}', \mathscr{QV}', N'L]$ and $E \equiv [\mathscr{Q}'', \mathscr{QV}'', N''L]$,
where $[\mathscr{Q}, QV, N] \to [\mathscr{Q}', QV', N']$ and $[\mathscr{Q}, \mathscr{QV}, N] \to [\mathscr{Q}'', \mathscr{QV}'', N'']$.
For this case we can apply the induction hypothesis.

— $D \equiv [\mathscr{Q}', \mathscr{QV}', NL']$ and $E \equiv [\mathscr{Q}'', \mathscr{QV}'', NL'']$,
where $[\mathscr{Q}, QV, L] \to [\mathscr{Q}', QV', L']$ and $[\mathscr{Q}, \mathscr{QV}, L] \to [\mathscr{Q}'', \mathscr{QV}'', L'']$.
For this case we can also apply the induction hypothesis.

— $N \equiv (\lambda x.P)$, $D \equiv [\mathscr{Q}, \mathscr{QV}, P\{L/x\}]$, $E \equiv [\mathscr{Q}', \mathscr{QV}', NL']$,
where $[\mathscr{Q}, \mathscr{QV}, L] \to_\beta [\mathscr{Q}', \mathscr{QV}', L']$.
Clearly, $[\mathscr{Q}, \mathscr{QV}, P\{L/x\}] \in \mathscr{C}$ and, by Lemma 2,

$$[\mathscr{Q}, \mathscr{QV}, P\{L/x\}] \to [\mathscr{Q}', \mathscr{QV}', P\{L'/x\}].$$

Moreover, $[\mathscr{Q}', \mathscr{QV}', NL'] \equiv [\mathscr{Q}', \mathscr{QV}', (\lambda x.P)L'] \to [\mathscr{Q}', \mathscr{QV}', P\{L'/x\}]$.

— $N \equiv (\lambda x.P)$, $D \equiv [\mathscr{Q}, \mathscr{QV}, P\{L/x\}]$ and $E \equiv [\mathscr{Q}', \mathscr{QV}', (\lambda x.P')L]$,
where $[\mathscr{Q}, \mathscr{QV}, P] \to_\beta [\mathscr{Q}', \mathscr{QV}', P']$.
Clearly, $[\mathscr{Q}, \mathscr{QV}, P\{L/x\}] \in \mathscr{C}$ and, by Lemma 2,

$$[\mathscr{Q}, \mathscr{QV}, P\{L/x\}] \to_\beta [\mathscr{Q}', \mathscr{QV}', P'\{L/x\}].$$

Moreover, $[\mathscr{Q}', \mathscr{QV}', (\lambda x.P')L] \to_\beta [\mathscr{Q}', \mathscr{QV}', P'\{L/x\}]$.

— $N \equiv (\lambda !x.P)$, $L \equiv !Q$, $D \equiv [\mathscr{Q}, \mathscr{QV}, P\{Q/x\}]$ and $E \equiv [\mathscr{Q}', \mathscr{QV}', (\lambda !x.P')L]$,
where $[\mathscr{Q}, \mathscr{QV}, P] \to_\beta [\mathscr{Q}', \mathscr{QV}', P']$.
Clearly, $[\mathscr{Q}, \mathscr{QV}, P\{Q/x\}] \in \mathscr{C}$ and, by Lemma 2,

$$[\mathscr{Q}, \mathscr{QV}, P\{Q/x\}] \to_\beta [\mathscr{Q}', \mathscr{QV}', P'\{Q/x\}].$$

Moreover, $[\mathscr{Q}', \mathscr{QV}', (\lambda x.P')!Q] \to_\beta [\mathscr{Q}', \mathscr{QV}', P'\{Q/x\}]$.

— $N \equiv (\lambda\langle x_1, \ldots, x_n\rangle.P)$, $L \equiv \langle r_1, \ldots, r_n\rangle$, $D \equiv [\mathscr{Q}, \mathscr{QV}, P\{r_1/x_1, \ldots, r_n/x_n\}]$, and $E \equiv [\mathscr{Q}', \mathscr{QV}', (\lambda\langle x_1, \ldots, x_n\rangle.P')L]$,
where $[\mathscr{Q}, \mathscr{QV}, P] \to_\beta [\mathscr{Q}', \mathscr{QV}', P']$.
Clearly, $[\mathscr{Q}, \mathscr{QV}, P\{r_1/x_1, \ldots, r_n/x_n\}] \in \mathscr{C}$ and, by Lemma 2,

$$[\mathscr{Q}, \mathscr{QV}, P\{r_1/x_1, \ldots, r_n/x_n\}] \to_\beta [\mathscr{Q}', \mathscr{QV}', P'\{r_1/x_1, \ldots, r_n/x_n\}].$$

Moreover, $[\mathscr{Q}', \mathscr{QV}', (\lambda\langle x_1, \ldots, x_n\rangle.P')L] \to_\beta [\mathscr{Q}', \mathscr{QV}', P'\{r_1/x_1, \ldots, r_n/x_n\}]$.

— $N \equiv (\lambda x.P)Q$, $D \equiv [\mathscr{Q}, \mathscr{QV}, (\lambda x.PL)Q]$ and $E \equiv [\mathscr{Q}, \mathscr{QV}, (P\{Q/x\})L]$ with $\alpha = \mathsf{r.cm}$, $\beta = \mathsf{l.}\beta$.
Clearly, $[\mathscr{Q}, \mathscr{QV}, (\lambda x.PL)Q] \to_{\mathsf{l.}\beta} [\mathscr{Q}, \mathscr{QV}, (P\{Q/x\})L]$.

— $N \equiv (\lambda\pi.P)Q$, $D \equiv [\mathscr{Q}, \mathscr{QV}, (\lambda\pi.PL)Q]$ and $E \equiv [\mathscr{Q}', \mathscr{QV}', ((\lambda\pi.P')Q)L]$ with $\alpha = \mathsf{r.cm}$,
where $[\mathscr{Q}, \mathscr{QV}, P] \to_\beta [\mathscr{Q}', \mathscr{QV}', P']$.
Clearly,

$$[\mathscr{Q}, \mathscr{QV}, (\lambda x.PL)Q] \to_{\mathsf{r.cm}} [\mathscr{Q}', \mathscr{QV}', (\lambda x.P'L)Q]$$

and

$$[\mathcal{Q}', \mathcal{QV}', ((\lambda\pi.P')Q)L] \rightarrow_{\beta} [\mathcal{Q}', \mathcal{QV}', (\lambda\pi.P'L)Q].$$

— $N \equiv (\lambda\pi.P)Q$, $D \equiv [\mathcal{Q}, \mathcal{QV}, (\lambda x.PL)Q]$ and $E \equiv [\mathcal{Q}', \mathcal{QV}', ((\lambda\pi.P)Q')L]$ with $\alpha = \mathsf{r.cm}$, where $[\mathcal{Q}, \mathcal{QV}, Q] \rightarrow_{\beta} [\mathcal{Q}', \mathcal{QV}', Q']$.
Clearly,

$$[\mathcal{Q}, \mathcal{QV}, (\lambda x.PL)Q] \rightarrow_{\mathsf{r.cm}} [\mathcal{Q}', \mathcal{QV}', (\lambda x.PL)Q']$$

and

$$[\mathcal{Q}', \mathcal{QV}', ((\lambda\pi.P)Q')L] \rightarrow_{\beta} [\mathcal{Q}', \mathcal{QV}', (\lambda\pi.PL)Q'].$$

— $N \equiv (\lambda\pi.P)Q$, $D \equiv [\mathcal{Q}, \mathcal{QV}, (\lambda x.PL)Q]$ and $E \equiv [\mathcal{Q}', \mathcal{QV}', ((\lambda\pi.P)Q)L']$ with $\alpha = \mathsf{r.cm}$, where $[\mathcal{Q}, \mathcal{QV}, L] \rightarrow_{\beta} [\mathcal{Q}', \mathcal{QV}', L']$.
Clearly,

$$[\mathcal{Q}, \mathcal{QV}, (\lambda x.PL)Q] \rightarrow_{\mathsf{r.cm}} [\mathcal{Q}', \mathcal{QV}', (\lambda x.PL')Q]$$

and

$$[\mathcal{Q}', \mathcal{QV}', ((\lambda\pi.P)Q)L'] \rightarrow_{\beta} [\mathcal{Q}', \mathcal{QV}', (\lambda\pi.PL')Q].$$

— $N \equiv (\lambda\pi.P)$, $L \equiv (\lambda x.Q)R$, $D \equiv [\mathcal{Q}, \mathcal{QV}, (\lambda x.NQ)R]$ and $E \equiv [\mathcal{Q}, \mathcal{QV}, N(Q\{R/x\})]$ with $\alpha = \mathsf{l.cm}$, $\beta = \mathsf{l.}\beta$.
Clearly,

$$[\mathcal{Q}, \mathcal{QV}, (\lambda x.NQ)R] \rightarrow_{\mathsf{l.}\beta} [\mathcal{Q}, \mathcal{QV}, N(Q\{R/x\})].$$

M cannot be in the form $\mathtt{new}(c)$ as that would give $D \equiv E$. □

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

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

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

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 to get a configuration F such that both $D \overset{*}{\rightarrow} F$ and $E \overset{*}{\rightarrow} F$, which gives a contradiction. □

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

Theorem 3. C is strongly normalising if and only if C is weakly normalising.

5. Examples

In this section we will give some simple examples showing 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 obtained easily by means of **cnot** and Hadamard's unitary operator **H**. The general schema of the term is

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

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

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

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

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

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

is the so-called β_{00} EPR state.

Deutsch's algorithm

Deutsch's algorithm was the first quantum algorithm to be defined. It has interesting applications: for example, it allows us to compute a global property of a function by combining results from two components of a superposition. We refer here to the presentation of *Deutsch's algorithm* given in Nielsen and Chuang (2000, pages 32–33) – a detailed explanation of the algorithm is beyond the scope of the current paper.

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

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

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

Deutsch's algorithm makes use of *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

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

as follows:

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

We have two cases:

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

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

— f is not a constant function; that is, $f(0) \oplus f(1) = 1$.
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:

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

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

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

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

6. Standardising computations

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

1 Perform classical reductions.
2 Perform reductions that build the underlying quantum register.
3 Perform quantum reductions.

In this section we prove a *standardisation theorem* that strengthens the common idea that a universal quantum computer should consist of a classical device 'setting up' a quantum circuit, which 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 $C \to_\alpha D$ and M be the relevant redex in C: if $\alpha \in \mathscr{Q}$, the redex M is said to be *quantum*; if $\alpha \in \mathscr{C}$, the redex M is said to be *classical*.

Definition 8. A configuration C is said to be *non-classical* if $\alpha \in n\mathscr{C}$ whenever $C \to_\alpha D$. Let NCL be the set of non-classical configurations. A configuration C is said to be *essentially quantum* if $\alpha \in \mathscr{Q}$ whenever $C \to_\alpha D$. Let EQT be the set of essentially quantum configurations.

Before stating the standardisation 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 $C_0 \equiv C$, $\varphi \leqslant \omega$ and:
1. for every $1 < i+1 < \varphi$, if $C_{i-1} \to_{n\mathscr{C}} C_i$, then $C_i \to_{n\mathscr{C}} C_{i+1}$;
2. for every $1 < i+1 < \varphi$, if $C_{i-1} \to_{\mathscr{Q}} C_i$, then $C_i \to_{\mathscr{Q}} C_{i+1}$.

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 reductions, while EQT is closed under quantum reductions.

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

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

In this way we are able to state and prove the Standardisation Theorem.

Theorem 4 (Standardisation). For every computation $\{C_i\}_{i<\varphi}$ such that $\varphi \in \mathbb{N}$, there is a CNQ computation $\{D_i\}_{i<\xi}$ such that $C_0 \equiv D_0$ and $C_{\varphi-1} \equiv D_{\xi-1}$.

Proof. We build a CNQ computation in three steps:

1 We start by reducing $D_0 \equiv C_0$ *using* $\mathscr{C}$ reductions as much as possible. By Theorem 3, we must obtain a finite reduction sequence $D_0 \to_{\mathscr{C}} \ldots \to_{\mathscr{C}} D_k$ such that $0 \leqslant k$ and no $\mathscr{C}$ reductions are applicable to D_k

2 Reduce D_k *using* new reductions as much as possible. By Theorem 3 we must obtain a finite reduction sequence $D_k \to_{\mathsf{new}} \cdots \to_{\mathsf{new}} D_j$ such that $k \leqslant j$ and no new reductions are applicable to D_j. Note that by Lemma 4 such reduction steps cannot generate classical redexes and, in particular, no classical redex can appear in D_j.

3 Reduce D_j *using* $\mathscr{Q}$ reductions as much as possible. By Theorem 3, we must obtain a finite reduction sequence $D_j \to_{\mathscr{Q}} \ldots \to_{\mathscr{Q}} D_m$ such that $j \leqslant m$ and no $\mathscr{Q}$ reductions are applicable to D_m. Note that by Lemma 5, such reduction steps can generate neither $\mathscr{C}$ redexes nor new redexes, and, in particular, neither $\mathscr{C}$ nor new reductions are applicable to D_m. Therefore, D_m *is in normal form.*

The reduction sequence $\{D_i\}_{i<m+1}$ is such that $D_0 \to_{\mathscr{C}} \ldots \to_{\mathscr{C}} D_k \to_{\mathsf{new}} \cdots \to_{\mathsf{new}} D_j \to_{\mathscr{Q}} \ldots \to_{\mathscr{Q}} D_m$ is a CNQ computation. By Theorem 2, we observe that $C_{\varphi-1} \equiv D_m$, which implies the thesis. □

The *intuition* behind a CNQ computation is that 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 then builds the quantum register without introducing any superposition. Finally, 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 for proving one side of the equivalence between Q and the 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 since there is no final measurable quantum state, and, consequently, the transformations of the quantum register are inaccessible (see also Section 8 for a discussion of the absence of measurements in Q).

The extension of standardisation to the infinite case makes this observation explicit. First note that we cannot have an infinite sequence of $n\mathscr{C}$ reductions.

Lemma 6. The relation $\to_{n\mathscr{C}}$ is strongly normalising (that is, there cannot be any infinite sequence $C_1 \to_{n\mathscr{C}} C_2 \to_{n\mathscr{C}} C_3 \to_{n\mathscr{C}} \ldots$).

Proof. Define the size $|M|$ of a term M as the number of symbols within it. Observe that if $[\mathscr{Q}, \mathscr{QV}, M] \to_{n\mathscr{C}} [\mathscr{Q}, \mathscr{QV}, N]$, we have $|N| < |M|$, which gives the result. □

As a trivial consequence of the Lemma we have the following result.

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

Finally, we can state the following theorem.

Theorem 5 (Standardisation for infinite computations). There is for every non-terminating computation $\{C_i\}_{i<\omega}$, a CNQ computation $\{D_i\}_{i<\omega}$ such that $C_0 \equiv D_0$.

Proof. We build the CNQ computation by first reducing $D_0 \equiv C_0$ *using* $\mathscr{C}$ reductions as much as possible. However, 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 (through the result of Ozawa and Nishimura (Nishimura and Ozawa 2008)), we have equivalence with quantum Turing machines as defined by Bernstein and Vazirani (Bernstein and Vazirani 1997). The fact that the class of circuit families we consider only contains finitely generated ones is not accidental: if we want to represent an entire family by one single lambda term (which is, by definition, a finite object), we must restrict consideration to families that are generated by a discrete set of gates.

Before going into the details, we will give an informal description of how our encoding works. Data will be encoded using some variations on Scott numerals (Wadsworth 1980). These can be used for both classical and quantum data. In the latter case, a more strict linear discipline (in general, quantum bits cannot be erased) 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. The crucial feature from a computational point of view is the way a quantum relevant term may modify 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 is true that Girard's encodings of intuitionistic logic into linear logic can be generalised in some way 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. For example, consider the lambda term $M \equiv x((\lambda y.yy)(\lambda y.yy))$: it is not normalisable, 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. However, 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 normalisable in 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, such as weak head reduction?

Even if there is a positive answer to this 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 just 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* Q.

Natural numbers are encoded as Q terms as follows:

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

In this way we can compute the successor and predecessor of a natural number as follows:

$$\begin{aligned} \mathsf{succ} &= \lambda z.!\lambda !x.\lambda !y.xz \\ \mathsf{pred} &= \lambda !z.z!(\lambda x.x)!\lceil 0 \rceil. \end{aligned}$$

Indeed,

$$\begin{aligned} \mathsf{succ}\,\lceil n \rceil &\to_{\mathscr{C}} !\lambda !x.\lambda !y.x\lceil n \rceil \equiv \lceil n+1 \rceil \\ \mathsf{pred}\,\lceil 0 \rceil &\to_{\mathscr{C}} (\lambda !x.\lambda !y.y)!(\lambda x.x)!\lceil 0 \rceil \overset{*}{\to}_{\mathscr{C}} \lceil 0 \rceil \\ \mathsf{pred}\,\lceil n+1 \rceil &\to_{\mathscr{C}} (\lambda !x.\lambda !y.x\lceil n \rceil)!(\lambda x.x)!\lceil 0 \rceil \to_{\mathscr{C}} (\lambda x.x)\lceil n \rceil \\ &\to_{\mathscr{C}} \lceil n \rceil. \end{aligned}$$

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

$$\begin{aligned} \mathsf{case}_0^{\mathsf{nat}} &\equiv \lambda !x.\lambda !y_0.\lambda !z.x!(\lambda !w.z)!y_0 \\ \mathsf{case}_{n+1}^{\mathsf{nat}} &\equiv \lambda !x.\lambda !y_0.\ldots.\lambda !y_{n+1}.\lambda !z.x!(\lambda !w.\mathsf{case}_n^{\mathsf{nat}} w!y_1\ldots!y_{n+1}!z)!y_0. \end{aligned}$$

They behave as follows:

$$\begin{aligned}
\forall m \leqslant n \quad \mathsf{case}^{\mathsf{nat}}_n\ \lceil m\rceil\,!M_0 \ldots !M_n !N &\xrightarrow{*}_{\mathscr{C}} M_m\\
\forall m > n \quad \mathsf{case}^{\mathsf{nat}}_n\ \lceil m\rceil\,!M_0 \ldots !M_n !N &\xrightarrow{*}_{\mathscr{C}} N.
\end{aligned}$$

We can capture *linear lists*, too: given any sequence $M_1, \ldots, M_n$ of terms (where $n \geqslant 0$), we can build a term $[M_1, \ldots, M_n]$ encoding the sequence by induction on n:

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

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

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

They behave on lists as follows:

$$\begin{aligned}
\mathsf{cons}\ M[M_1, \ldots, M_n] &\xrightarrow{*}_{\mathscr{C}} [M, M_1, \ldots, M_n]\\
\mathsf{sel}\ []\,!N!L &\xrightarrow{*}_{\mathscr{C}} L\\
\mathsf{sel}\ [M, M_1, \ldots, M_n]\,!N!L &\xrightarrow{*}_{\mathscr{C}} NM[M_1, \ldots, M_n].
\end{aligned}$$

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

$$\begin{aligned}
\mathsf{append}_n[N_1, \ldots, N_m]M_1 \ldots M_n &\xrightarrow{*}_{\mathscr{C}} [M_1, \ldots, M_n, N_1, \ldots, N_m]\\
\forall m \leqslant n \quad \mathsf{extract}_n M[N_1, \ldots, N_m] &\xrightarrow{*}_{\mathscr{C}} M[]N_m N_{m-1} \ldots N_1\\
\forall m > n \quad \mathsf{extract}_n M[N_1, \ldots N_m] &\xrightarrow{*}_{\mathscr{C}} M[N_{n+1} \ldots N_m]N_n N_{n-1} \ldots N_1.
\end{aligned}$$

Terms append_n can be built by induction on n:

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

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

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

The encodings of natural numbers and lists are similar, and are both in the style of the so-called Scott 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 $[M_1,\dots,M_n]$ that are part of $[M, M_1,\dots,M_n]$ do not lie in the scope of any bang operator.

We need *recursion and iteration* so that we can 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,

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

This will help us in encoding algorithms using 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,

$$\begin{aligned}
\mathsf{rec}^{\mathsf{nat}}\ \lceil 0\rceil!M!N &\overset{*}{\rightarrow}_{\mathscr{C}} \mathsf{rec}^{\mathsf{nat}}_{\mathsf{aux}}\ !(\mathsf{rec}^{\mathsf{nat}})\lceil 0\rceil!M!N\\
&\overset{*}{\rightarrow}_{\mathscr{C}} (\lambda!x.\lambda!y.y)!(\lambda!v.M!(\mathsf{rec}^{\mathsf{nat}}\ !v!M!N)!v)!N\\
&\overset{*}{\rightarrow}_{\mathscr{C}} N\\
\mathsf{rec}^{\mathsf{nat}}\ \lceil n+1\rceil!M!N &\overset{*}{\rightarrow}_{\mathscr{C}} (\lambda!x.\lambda!y.x\lceil n\rceil)!(\lambda!v.M!(\mathsf{rec}^{\mathsf{nat}}\ !v!M!N)!v)!N\\
&\overset{*}{\rightarrow}_{\mathscr{C}} (\lambda!v.M!(\mathsf{rec}^{\mathsf{nat}}\ !v!M!N)!v)!\lceil n\rceil\\
&\overset{*}{\rightarrow}_{\mathscr{C}} M!(\mathsf{rec}^{\mathsf{nat}}\ \lceil n\rceil!M!N)!\lceil n\rceil.
\end{aligned}$$

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,

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

Definition 10. A (partial) function $f : \mathbb{N}^n \rightarrow \mathbb{N}$ is *representable* if and only if there is a term M_f such that:

— Whenever $M_f\lceil m_1\rceil\dots\lceil m_n\rceil$ has a normal form N (with respect to $\overset{*}{\rightarrow}_{\mathscr{C}}$), we have $N \equiv \lceil m\rceil$ for some natural number m.

— $M_f \lceil m_1 \rceil \ldots \lceil m_n \rceil \overset{*}{\rightarrow}_{\mathscr{C}} \lceil m \rceil$ if and only if $f(m_1, \ldots, m_n)$ is defined and equal to m.

As we mentioned at the beginning of this section, the following result 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 encoded easily.

— The composition $f : \mathbb{N}^m \to \mathbb{N}$ of $h : \mathbb{N}^n \to \mathbb{N}$ and $g_1, \ldots, g_n : \mathbb{N}^m \to \mathbb{N}$ can be represented as follows:

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

— 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:

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

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

$$M_f \equiv \lambda x_1.\ldots.\lambda x_n.\mathsf{rec}!(N_g)\lceil 0 \rceil x_1 \ldots x_n$$

where

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

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

Quantum relevant terms. In this section, we will introduce the class of quantum relevant terms. In the next sections, we will prove that the class of functions that are captured by quantum relevant terms coincides with the class of functions that 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 $C \overset{*}{\rightarrow}_{\mathscr{S}} D$ and D is in normal form with respect to the relation $\rightarrow_{\mathscr{S}}$. We will write $C \Downarrow D$ to stand for $C \Downarrow_{\mathscr{L}} D$.

Confluence and the equivalence between weakly normalising and strongly normalising configurations allow the following definition.

Definition 12. A term M is said to be *quantum relevant* (written *qrel* for short) if it is well formed and for each list $![!c_1, \ldots, !c_n]$ there are a quantum register $\mathscr{Q}$ and a natural number m such that $[1, \emptyset, M![!c_1, \ldots, !c_n]] \Downarrow [\mathscr{Q}, \{r_1, \ldots, r_m\}, [r_1, \ldots, r_m]]$.

In other words, a quantum relevant term is the analogue of a pure λ-term representing a function on natural numbers. One can see immediately that the class of qrel terms is not recursively enumerable.

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

An *n-qubit gate* (or, simply, a *qubit gate*) is a unitary operator $\mathbf{U} : \mathbb{C}^{2^n} \rightarrow \mathbb{C}^{2^n}$, and a $\mathscr{V}$*-qubit gate* (where $\mathscr{V}$ is a qvs) is a unitary operator $\mathbf{G} : \mathscr{H}(\mathscr{V}) \rightarrow \mathscr{H}(\mathscr{V})$. We will work here with elementary operators only. If $\mathscr{G}$ is a set of qubit gates, a $\mathscr{V}$*-circuit* $\mathbf{K}$ based on $\mathscr{G}$ is a sequence

$$\mathbf{U}_1, r_1^1, \dots, r_{n_1}^1, \dots, \mathbf{U}_m, r_1^m, \dots, r_{n_m}^m$$

where, for every $1 \leqslant i \leqslant m$, we have:

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

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

$$\mathbf{K} = \mathbf{U}_1, r_1^1, \dots, r_{n_1}^1, \dots, \mathbf{U}_m, r_1^m, \dots, r_{n_m}^m$$

is the unitary operator

$$U_{\mathbf{K}} = (\mathbf{U}_m)_{\langle\langle r_1^m, \dots, r_{n_m}^m \rangle\rangle} \circ \dots \circ (\mathbf{U}_1)_{\langle\langle r_1^1, \dots, r_{n_1}^1 \rangle\rangle}.$$

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 $\mathscr{G}$ is a triple (f, g, h) where:

— $f : \mathbb{N} \rightarrow \mathbb{N}$ is a computable function.
— $g : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ is a computable function such that $0 \leqslant g(n, m) \leqslant n + 1$ whenever $1 \leqslant m \leqslant f(n)$.
— $h : \mathbb{N} \rightarrow \mathbb{N}$ is a computable function such that for every $n \in \mathbb{N}$, we have $\mathbf{K}_{h(n)}$ is a $\{r_1, \dots, r_{f(n)}\}$-circuit based on $\mathscr{G}$.

Any family of circuits (f, g, h) induces a function $\Phi_{f,g,h}$ (the *function induced by* (f, g, h)) that, when given any finite sequence $c_1, \dots, c_n$ in $\{0, 1\}^*$, will return an element of $\mathscr{H}(\{r_1, \dots, r_{f(n)}\})$,

$$\Phi_{f,g,h}(c_1, \dots, c_n) = U_{\mathbf{K}_{h(n)}}(|r_1 \mapsto c_{g(n,1)}, \dots, r_{f(n)} \mapsto c_{g(n,f(n))}\rangle)$$

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 $\mathscr{G}$ 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 consequence, we can assume that f, g and h are representable whenever (f, g, h) is a family of circuits.

The n-th elementary permutation of m elements (where $1 \leqslant n < m$) is the function that maps n to $n + 1$, $n + 1$ to n and any other elements in the interval $1, \dots, m$ 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* if and only if for every list $[N_1, \dots, N_m]$ with $m > n$, we have

$$M[N_1, \dots, N_m] \overset{*}{\to}_{\mathscr{C}} [N_1, \dots, N_{n-1}, N_{n+1}, N_n, N_{n+2}, \dots, N_m].$$

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

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

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

where

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

Indeed,

$$\begin{aligned} M_{el}\lceil 0 \rceil &\to_{\mathscr{C}} \mathsf{rec}^{\mathsf{nat}}\ \lceil 0 \rceil !N!L \to_{\mathscr{C}} L \\ L[M_1, \dots, M_m] &\to_{\mathscr{C}} \mathsf{extract}_2(\lambda z.\lambda w.\lambda q.\mathsf{append}_2 zwq)[M_1, \dots, M_m] \\ &\overset{*}{\to}_{\mathscr{C}} (\lambda z.\lambda w.\lambda q.\mathsf{append}_2 zwq)[M_3, \dots, M_m]M_2M_1 \\ &\overset{*}{\to}_{\mathscr{C}} \mathsf{append}_2[M_3, \dots, M_m]M_2M_1 \\ &\overset{*}{\to}_{\mathscr{C}} [M_2, M_1, M_3, \dots, M_m] \equiv [M_{\rho^1_m(1)}, \dots, M_{\rho^1_m(m)}] \\ M_{el}\lceil n+1 \rceil &\to_{\mathscr{C}} \mathsf{rec}^{\mathsf{nat}}\ \lceil n+1 \rceil !N!L \to_{\mathscr{C}} L \\ &\overset{*}{\to}_{\mathscr{C}} N!(\mathsf{rec}^{\mathsf{nat}}\ \lceil n \rceil !N!L)\lceil n \rceil \\ &\to_{\mathscr{C}} \lambda w.\mathsf{extract}_1(\lambda q.\lambda s.\mathsf{append}_1((\mathsf{rec}^{\mathsf{nat}}\ \lceil n \rceil !N!L)q)s)w \\ &\to_{\mathscr{C}} \lambda w.\mathsf{extract}_1(\lambda q.\lambda s.\mathsf{append}_1(Pq)s)w \equiv Q \\ Q[M_1, \dots, M_n] &\to_{\mathscr{C}} \mathsf{extract}_1(\lambda q.\lambda s.\mathsf{append}_1(Pq)s)[M_1, \dots, M_n] \\ &\overset{*}{\to}_{\mathscr{C}} (\lambda q.\lambda s.\mathsf{append}_1(Pq)s)[M_2, \dots, M_m]M_1 \\ &\overset{*}{\to}_{\mathscr{C}} \mathsf{append}_1(P[M_2, \dots, M_m])M_1 \\ &\overset{*}{\to}_{\mathscr{C}} \mathsf{append}_1([M_{\rho^n_{m-1}(1)+1}, \dots, M_{\rho^n_{m-1}(m-1)+1}])M_1 \\ &\overset{*}{\to}_{\mathscr{C}} [M_1, M_{\rho^n_{m-1}(1)+1}, \dots, M_{\rho^n_{m-1}(m-1)+1}] \equiv [M_{\rho^{n+1}_m(1)}, \dots, M_{\rho^{n+1}_m(m)}], \end{aligned}$$

which completes the proof. □

Lemma 9. There is a term M_{length} such that, for every list $[!N_1, \dots, !N_n]$, we have $M_{length}[!N_1, \dots, !N_n] \overset{*}{\to}_{\mathscr{C}} \lceil n \rceil$.

Proof. M_{length} is the term

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

Indeed,

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

which completes the proof. □

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

$$\begin{array}{rrcl} & M_{choose}\lceil 0\rceil\,[!N_1,\ldots,!N_m] & \overset{*}{\to}_{\mathscr{C}} & !\lceil 0\rceil \\ \forall\ 1\leqslant n\leqslant m & M_{choose}\lceil n\rceil\,[!N_1,\ldots,!N_m] & \overset{*}{\to}_{\mathscr{C}} & !N_n \\ & M_{choose}\lceil m+1\rceil\,[!N_1,\ldots,!N_m] & \overset{*}{\to}_{\mathscr{C}} & !\lceil 1\rceil. \end{array}$$

Proof. M_{choose} is the term

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

where

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

Indeed,

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

where

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

Now,

$$\begin{aligned}
\lceil 0\rceil !Q!(L_{=0})S &\overset{*}{\to}_{\mathscr{C}} L_{=0}S \overset{*}{\to}_{\mathscr{C}} S\lceil 0\rceil \overset{*}{\to}_{\mathscr{C}} !\lceil 0\rceil \\
\lceil 1\rceil !Q!(L_{=0})S &\overset{*}{\to}_{\mathscr{C}} (\lambda!s.\lambda r.(s!L_{\geqslant 2}!L_{=1}\{N/w\})r)\lceil 0\rceil S \\
&\overset{*}{\to}_{\mathscr{C}} (\lambda!x.\lambda!y.y)!L_{\geqslant 2}!(L_{=1}\{N/w\})S \\
&\overset{*}{\to}_{\mathscr{C}} L_{=1}\{N/w\}S \\
&\overset{*}{\to}_{\mathscr{C}} (\lambda!u.!N)(S!\lceil 0\rceil) \\
&\overset{*}{\to}_{\mathscr{C}} (\lambda!u.!N)!\lceil 0\rceil \overset{*}{\to}_{\mathscr{C}} !N \\
\lceil n+2\rceil !Q!(L_{=0})S &\overset{*}{\to}_{\mathscr{C}} (\lambda!s.\lambda r.(s!L_{\geqslant 2}!L_{=1}\{N/w\})r)\lceil n+1\rceil S \\
&\overset{*}{\to}_{\mathscr{C}} (\lambda!x.\lambda!y.x\lceil n\rceil)!L_{\geqslant 2}!(L_{=1}\{N/w\})S \\
&\overset{*}{\to}_{\mathscr{C}} L_{\geqslant 2}\lceil n\rceil S \\
&\overset{*}{\to}_{\mathscr{C}} S\lceil n+1\rceil,
\end{aligned}$$

which completes the proof. □

We now have all the required ingredients for proving that any finitely generated family of circuits can be represented in 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:

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

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

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

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

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

are distinct and in $\{r_1, \ldots, r_{f(h(i))}\}$. So, there are permutations π^i_j of $\{1, \ldots, f(h(i))\}$ such that $\pi^i_j(x) = y$ if and only if $r^{h(i),x}_j = r_y$ for every $1 \leqslant x \leqslant p(h(i), k(h(i)))$. Let ρ^i_j be the inverse of π^i_j. Clearly, both π^i_j and ρ^i_j can be effectively computed from i and j.

As a consequence, the following functions are partial recursive (in the 'classical' sense):

— A function $r : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ that, given (i, j), returns the number of elementary permutations of $\{1, \dots, f(h(i))\}$ in which π^i_j can be decomposed (using Lemma 7).

— A function $q : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that $q(i, j, x) = y$ if and only if the x-th elementary permutation of $\{1, \dots, f(h(i))\}$ in which π^i_j can be decomposed is the y-th elementary permutation.

— A function $s : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ that, given (i, j), returns the number of elementary permutations of $\{1, \dots, f(h(i))\}$ in which ρ^i_j can be decomposed (using Lemma 7).

— A function $t : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that $t(i, j, x) = y$ if and only if the x-th elementary permutation of $\{1, \dots, f(h(i))\}$ in which ρ^i_j can be decomposed is the y-th elementary permutation.

We will now build up a term M_{init} that, given a list L of boolean constants and a natural number $\lceil n \rceil$, computes the input list for $\mathbf{K}_{h(n)}$ from L.

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

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 $\lceil n \rceil$, 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

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

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. In 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 $![!c_1, \dots, !c_n], ![!d_1, \dots, !d_n]$ be two lists of bits (with the same length) and suppose that $[1, M![!c_1, \dots, !c_n]] \Downarrow_{n\mathcal{Q}} [\mathcal{Q}, N]$. Clearly, N cannot contain any boolean constants since M is assumed to be qrel. By applying exactly the same computation steps that lead from $[1, M![!c_1, \dots, !c_n]]$ to $[\mathcal{Q}, N]$, we can prove that $[1, M![!d_1, \dots, !d_n]] \Downarrow_{n\mathcal{Q}} [\mathcal{Q}', N]$, where $\mathcal{Q}$ and $\mathcal{Q}'$ live in the same Hilbert Space $\mathscr{H}(\mathbf{Q}(N))$ and are both elements of the computational basis. Moreover, any computation step leading from $[1, M![!c_1, \dots, !c_n]]$ to $[\mathcal{Q}, N]$ is effective, that is, it is intuitively computable (in the classical sense). Therefore, by the Church–Turing Thesis, we obtain the following proposition.

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 $c_1, \dots, c_n$, we have $[1, M![!c_1, \dots, !c_n] \Downarrow_{n\mathcal{Q}} [|r_1 \mapsto c_{g(n,1)}, \dots, r_{f(n)} \mapsto c_{g(n,f(n))}\rangle, N]$, where we conventionally set $c_0 \equiv 0$ and $c_{n+1} \equiv 1$.

Consider $[\mathcal{Q}, M] \in \mathsf{EQT}$ and assume that $[\mathcal{Q}, M] \Downarrow_{\mathcal{Q}} [\mathcal{Q}', [r_1, \dots, r_m]]$. Then $\mathcal{Q}$ and $\mathcal{Q}'$ live in the same Hilbert space:

$$\mathscr{H}(\mathbf{Q}(M)) = \mathscr{H}(\mathbf{Q}([r_1, \dots, r_m])) = \mathscr{H}(\{r_1, \dots, r_m\}).$$

The sequence of reductions in this computation allows us to build in an effective way a unitary transformation $\mathbf{U}$ such that $\mathcal{Q}' = \mathbf{U}_{\langle r_1, \dots, r_m\rangle}(\mathcal{Q})$. Summarising, we have the following proposition.

Proposition 7. Let M be a term containing quantum redexes only. Then there is a circuit $\mathbf{K}$ such that $\mathcal{Q}' = U_{\mathbf{K}}(\mathcal{Q})$ whenever $[\mathcal{Q}, M] \Downarrow_{\mathcal{Q}} [\mathcal{Q}', M']$. Moreover, $\mathbf{K}$ is generated by gates appearing in M. Furthermore, $\mathbf{K}$ can be effectively computed from M.

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

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

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

Note that the standardisation theorem is a great help 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

It is not possible in Q to make a classical observation of the content of the quantum register. More specifically, the language of terms does not include any measurement operator that, applied to a quantum variable, has the effect of observing the value of the related qubit. This is in contrast 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 $\texttt{meas}(\cdot)$ (in the style of λ_{sv}) would not be particularly problematic. However, some of the properties we have proved here would no longer be true. In particular:

— The reduction relation would be probabilistic, since observing a qubit can have different outcomes. As a consequence, confluence would no longer be true.
— The standardisation theorem would not hold in the form given here. In particular, the application of unitary transformations to the underlying quantum register may not necessarily be postponed until the end of a computation.

The main reason for restricting our attention to a calculus without any explicit measurement operator is that the (extensional) expressive power of the calculus obtained (that is, the extensional class of quantum computable functions) would presumably be the same. More precisely, in Bernstein and Vazirani (1997, page 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'* in Nielsen and Chuang (2000, page 186), and the discussion in Section 1 of the present paper.

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

However, note that the possibility of measuring qubits internally (for example, by a construct like $\texttt{meas}(\cdot)$ could allow us to solve certain problems more efficiently by exploiting the inherent non-determinism involved in measurements. Indeed, it is not known whether measurement-based quantum computation can be *efficiently* (with a polynomial overhead) simulated by measurement-free quantum computation. This interesting question goes well beyond the scope of this paper.

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

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

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

9. Conclusions and further work

In this paper we have studied Q, a quantum lambda calculus based on the paradigm of 'quantum data and classical control'. Unlike most of the related literature, which focuses on semantical issues, we have tackled 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 standardisation theorem, which should help in clarifying the interaction between the classical and quantum worlds (at least in a λ-calculus setting). We have also considered operational properties of the calculus, such as subject reduction and confluence.

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

Another possible direction for our future research is the study of measurement so that we can move from a calculus of computable functions towards a more concrete functional programming language.

Appendix A. Hilbert spaces

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

1 An *inner product* $\langle\cdot,\cdot\rangle_{\mathscr{H}} : \mathscr{H} \times \mathscr{H} \to \mathbb{C}$ such that:
   (a) $\langle\phi,\psi\rangle_{\mathscr{H}} = \langle\psi,\phi\rangle^*_{\mathscr{H}}$.
   (b) $\langle\psi,\psi\rangle_{\mathscr{H}}$ is a non-negative real number.
   (c) If $\langle\psi,\psi\rangle_{\mathscr{H}} = 0$, then $\psi = \mathbf{0}$.
   (d) $\langle c_1\phi_1 + c_2\phi_2,\psi\rangle_{\mathscr{H}} = c_1^*\langle\phi_1,\psi\rangle_{\mathscr{H}} + c_2^*\langle\phi_2,\psi\rangle_{\mathscr{H}}$.
   (e) $\langle\phi, c_1\psi_1 + c_2\psi_2\rangle_{\mathscr{H}} = c_1\langle\phi,\psi_1\rangle_{\mathscr{H}} + c_2\langle\phi,\psi_2\rangle_{\mathscr{H}}$.
2 A *norm* $||\cdot||_{\mathscr{H}} : \mathscr{H} \to \mathbb{R}^+$ defined by $||v||_{\mathscr{H}} = \langle v,v\rangle^{1/2}_{\mathscr{H}}$.

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

Proposition 8. Each finite dimensional complex vector space $\mathscr{H}$ equipped with an inner product $\langle\cdot,\cdot\rangle_{\mathscr{H}}$ is a Hilbert space with respect to the metric $d(\psi,\phi) = ||\psi - \phi||_{\mathscr{H}}$.

When it is clear from the context, we will simply write $\langle\cdot,\cdot\rangle$ and $||\cdot||$ instead of $\langle\cdot,\cdot\rangle_{\mathscr{H}}$ and $||\cdot||_{\mathscr{H}}$.

Let $\mathscr{H}$ be a Hilbert space and ϕ,ψ be generic vectors of $\mathscr{H}$. Then, ϕ is *normalised* if $||\phi|| = 1$, and ϕ and ψ are *orthogonal* if $\langle\phi,\psi\rangle = 0$. A set $\mathbf{b} = \{\phi_1,\ldots,\phi_n\} \subseteq \mathscr{H}$ is an orthonormal basis if:

1 If $\phi \in \mathscr{H}$, then $\phi = \sum_{i=1}^n d_i\phi_i$ (where each d_i is in $\mathbb{C}$).
2 Each $\phi \in \mathbf{b}$ is normalised.
3 If $\phi,\psi \in \mathbf{b}$ and $\phi \neq \psi$, then $\langle\phi,\psi\rangle = 0$.

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

Definition 15 (Tensor of Hilbert spaces). Let $\mathcal{H}', \mathcal{H}''$ be two Hilbert spaces with inner products $\langle \cdot, \cdot \rangle_{\mathcal{H}'}, \langle \cdot, \cdot \rangle_{\mathcal{H}''}$. The *tensor product* of $\mathcal{H}'$ and $\mathcal{H}''$ is the Hilbert space $\mathcal{H}' \otimes \mathcal{H}''$ built as follows.
Let $\mathcal{H}' \bullet \mathcal{H}''$ be the space freely generated by the set $\mathcal{H}' \times \mathcal{H}''$. Now 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 $(\mathcal{H}' \bullet \mathcal{H}'')/S$ be the quotient space[†]. The tensor product of $\mathcal{H}'$ and $\mathcal{H}''$ is $\mathcal{H} = \mathcal{H}' \otimes \mathcal{H}'' \overset{def}{=} (\mathcal{H}' \bullet \mathcal{H}'')/S$, where the inner product in $\mathcal{H}$ is defined by

$$\begin{aligned}
\langle \phi_1 \otimes \psi_1, \phi_2 \otimes \psi_2 \rangle_{\mathcal{H}} &= \langle \phi_1, \phi_2 \rangle_{\mathcal{H}'} \langle \psi_1, \psi_2 \rangle_{\mathcal{H}''} \\
\langle c_1\phi_1 + c_2\phi_2, \psi \rangle_{\mathcal{H}} &= c_1^* \langle \phi_1, \psi \rangle_{\mathcal{H}} + c_2^* \langle \phi_2, \psi \rangle_{\mathcal{H}} \\
\langle \phi, c_1\psi_1 + c_2\psi_2 \rangle_{\mathcal{H}} &= c_1 \langle \phi, \psi_1 \rangle_{\mathcal{H}} + c_2 \langle \phi, \psi_2 \rangle_{\mathcal{H}}.
\end{aligned}$$

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 basis of $\mathcal{H}' \otimes \mathcal{H}''$.

Definition 16. Let $\mathcal{H}'$ and $\mathcal{H}''$ be two Hilbert spaces and let U and V be unitary operators in $\mathcal{H}'$ and $\mathcal{H}''$, respectively. 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 $(U \otimes V)(\sum_{i=1}^{k} b_i\phi_i) = \sum_{i=1}^{k} b_i(U \otimes V)\phi_i$.

References

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

Altenkirch, T, and Grattage, J. (2005) A functional quantum programming language. *Proc. of the 20th Annual IEEE Symposium on Logic in Computer Science* 249–258.

Arrighi, P. and Dowek, G. (2008) Linear-algebraic lambda-calculus: higher-order, encodings and confluence. *Springer-Verlag Lecture Notes in Computer Science* **5117** 17–81.

Basdevant, J. L. and Dalibard, J. (2005) *Quantum mechanics*, Springer-Verlag.

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

Danos, V. and Kashefi, E. and Panangaden, P. (2007) The measurement calculus. *Journal of the ACM* **54** (2).

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

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

Feynman, R. P. (1982) Simulating physics with computers. *Internat. J. Theoret. Phys.* **21** (6–7) 467–488.

Grover, L. K. (1999) Quantum search on structured problems. In: Quantum computing and quantum communications (Palm Springs, CA, 1998). *Springer-Verlag Lecture Notes in Computer Science* **1509** 126–139.

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

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

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

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

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

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

Perdrix, S. (2007) Quantum Patterns and Types for Entanglement and Separability. *Electr. Notes Theor. Comput. Sci.* **170** 125–138.

Selinger, P. (2004) Towards a Quantum Programming Language. *Mathematical Structures in Computer Science* **14** (4) 527–586.

Selinger, P. and Valiron, B. (2006) A lambda calculus for quantum computation with classical control. *Mathematical Structures in Computer Science* **16** (3) 527–552.

Shor, P. W. (1994) Algorithms for quantum computation: discrete logarithms and factoring. In: *Proc. 35th Annual Symposium on Foundations of Computer Science* 124–134.

Simpson, A. (2005) Reduction in a linear lambda-calculus with applications to operational semantics. In: Proc. of Proc. of the 16th Annual Conference on Term Rewriting and Applications. *Springer-Verlag Lecture Notes in Computer Science* **3467** 219–234.

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

Wadler, P. (1994) A syntax for linear logic. In: Proc. of the 9th International Conference on Mathematical Foundations of Programming Semantics. *Springer-Verlag Lecture Notes in Computer Science* **802** 513–529.

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

Yao, A. (1993) Quantum Circuit Complexity. In: *Proc. 34th Annual Symposium on Foundations of Computer Science*, IEEE 352–360.