Margherita Zorzi

Lambda Calculi and Logics for Quantum Computing

Ph.D. Thesis

June 11, 2009

Università degli Studi di Verona
Dipartimento di Informatica

Advisor:
prof. Andrea Masini

Series N°: **TD-07-09**

Università di Verona
Dipartimento di Informatica
Strada le Grazie 15, 37134 Verona
Italy

a Nicola Sacco
e a Bartolomeo Vanzetti

Contents

Part II Main Theme: Quantum Lambda Calculi

1

Introduction

1.1 Quantum Computing

Quantum Computing is one of the most promising fields of Computer Science. It is devoted to developing an alternative computational model, based on quantum physics rather than classical physics.

The original idea was by Richard Feynman, who put a simple but interesting questions: *is a classical computer able to simulate a generic physical system?*

Given a physical system of N interacting particles, it can be fully described by a function of the shape $\psi(x_1, \ldots, x_N, t)$ where t represents time. The answer to Feynman's question is that, if the system is classical, a classical computer can efficiently simulate the full description with polynomial slowdown and, in the case of a quantum system, *a classical computer can efficiently simulate the full description with exponential slowdown*.

The birth of quantum computing coincided therefore with the attempt to understand how a computational device could be in order to emulate an arbitrary physical system. The seminal ideas by Feynman was resumed by P. Benioff [20], but the first concrete proposal for a quantum abstract computer is due to Deutsch, who introduced quantum Turing machines [35]. Deutsch started from the probabilistic version of the Church Turing Thesis, attempting to understand which physical basis are required in order to define more and more stronger versions of Church Turing Thesis. Deutsch introduced also the first quantum algorithm (subsequently called *Deutsch's algorithm*).

Starting from the Deutsch's fundamental works, E. Bernstein and U. Vazirani defined in [22] the Quantum Universal Turing Machine, and developed a complexity theory for quantum computing, revisiting and extending the results from the classical and the probabilistic cases.

Other quantum computational models have been subsequently defined: Quantum Circuits Families, also by Deutsch in 1989 and subsequently developed by Yao [104], and the Quantum Random Access Machines (QRAM), by Knill [61], a classically controlled machine plus a quantum device. On the ground of a QRAM model, Peter Selinger, in [85] defined the first functional language based on the so called *quantum data-classical control* paradigm, a functional, statically typed language, whose semantics, given in term of superoperators, is fully abstract.

The quantum computing had a strong impact on the notion of problems "computational tractability". The most surprising result is due to Peter Shor [88,89], which prove that two classically intractable problems such as the *factorization of integers* and the *discrete logarithm* could be efficiently (namely polynomially) solved by a quantum computer. Shor's Algorithm on prime factorization catalyzed the interest of scientific community toward the quantum computing research.

Nowadays, quantum computing foundation have no stabilized yet, and so there are several foundational interesting problems to investigate. Focalizing on theoretical (non necessary algorithmic) aspects, it is possible to state three main subjects strongly related to the proposals of this thesis:

On quantum computability: quantum computability is quite underdeveloped compared to its classical counterpart. One of the most important challenges in quantum computing, is the necessity of developing suitable *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. Some quantum computational models, such as the quantum Turing machine [22, 35] and the quantum circuit families [73, 74, 104] have been defined, and nowadays they are universal accepted and used as reference for theoretical studies. But they are essentially "first-order", so it seem to be mandatory to give a contribution to the definition of a quantum computational model for higher–order–functions.
So, as for the classical case, it seem to be interesting to study a primitive formalism based on the concept of abstraction and application, i.e. a quantum version of λ-calculus. The first seminal proposals of quantum lambda calculi were by A. Van Tonder [97] and by P. Selinger and B. Valiron [87] (see Section 3.3) as a foundation of higher order quantum functional programming languages. A related, interesting problem is to study quantum λ-calculi from a computability point of view, focalizing for example on aspects such as the expressive power with a comparison with quantum computational models (quantum Turing machine and quantum circuit families).
Moreover, quantum computations have several interesting features and, as for the classical case, standardization and confluence result are interesting subjects of the research. This become more complex, but also more interesting, in presence of measurements during the computation. Moreover, the computational behavior of infinite quantum computations with measurement is yet relatively unexplored territory.

On quantum complexity: one of the main motivation for studying computational applications of quantum mechanics is the potentiality to exploit quantum parallelism in order to reduce (as Shor did) the computational complexity of classical hard problems [76].
The crucial attention on complexity problems naturally involved the necessity of a complexity theory ad hoc for quantum computing setting. Since seminal work by Bernstein and Vazirani [22], new important quantum complexity classes have been defined, with particular emphasis on the quantum polytime. As for the classical case, complexity classes are generally defined on a quantum computational model such as quantum Turing machine (as in Bernstein and Vazirani paper) or quantum circuit families [73, 74, 104]. Important steps are moved toward the develop of a general quantum complexity theory, but it is very far from the "completeness" reached in the classical case. This is due also to the intrinsic difficulties related to foundational issue of quantum computing. Let us consider the class of the quantum Turing machine à la Bernstein and Vazirani [22]: each computation evolves as a superposition in a

space of configuration (a suitable Hilbert space, as we will see), and each classical computation in superposition can evolve independently. So, the result of a quantum computation is obtained, at the end, with a measurement of the several superpositional result, and so it is irremediably probabilistic. This induces the definition of three distinct quantum polytime classes (EQP, BQP, ZQP), since different constraint can be imposed on success or errors [22].

On logical systems for quantum computing: since the work of Birkhoff and von Neumann in 1936 [23], various logics have been investigated as a means to formalize reasoning about propositions taking into account the principles of quantum theory, e.g. [4, 31, 32, 70]. In general, it is possible to view *quantum logic* as a logical axiomatization of the mathematical structures associated to quantum mechanics (e.g. orthomodular spaces).
In our opinion the definition of a logic *for* quantum computing appear to be a different topic, that is nowadays quite unexplored (one of the few good paper in this direction is [15] or [14]). But what does it mean "logical system for quantum computing"? Such a system should be a formal system that is able to describe quantum computations, not a further axiomatization of quantum mechanics spaces.

1.2 Original contributions of this thesis

This thesis deal mainly with the development of new quantum λ-calculi. As a kind of a variation on the theme, we propose also new deduction systems to deal with quantum computations.

1.2.1 Quantum λ-calculi

Q calculus

We start our investigation by proposing a quantum, type-free λ–calculus with classical control and quantum data that we call Q. The syntax for terms and configurations is inspired by the seminal work by Selinger and Valiron [87] and moreover, taking into account linearity, we implicitly use linear logic in a way similar to Van Tonder's calculus [97].
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 [100]). In order to be correct w.r.t. term reduction we have proved a suitable *subject reduction theorem*. The Q calculus is not endowed with a reduction strategy (it is neither call–by–value nor call–by–name), therefore we have studied the problem of confluence. Noticeably, confluence holds in a strong way (weak normalization implies strong normalization). Another remarkable feature of the calculus is given by the *(quantum) standardization theorem*. Roughly speaking: for each terminating computation there is another *standard*, equivalent, computational where computation steps are performed in the following order:

1. first, *classical* reductions: in this phase the quantum register is empty and all the computations steps are classical;

2. secondly, reductions that *build the quantum register*;
3. and finally *quantum* reductions, namely controlled applications of unitary transformations to the quantum register.

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

Subsequently, we will go along our investigation by proposing an expressiveness study of Q. We investigate the relationship between Q and one of the most important quantum computing systems, such as quantum circuit families [73,74,104]: we prove in a detailed and rigorous way the *equivalence between our calculus and quantum circuit families*.

The Q calculus is measurement free, in fact the absence of measurement is a feature of standard quantum computational models, such as quantum Turing machines and quantum circuits families.

We assume to make an unique *implicit measurement* at the end of computation. Cause the importance of measurement, we also investigate a measurement extension of Q (see later the Q* calculus).

This part is based on the paper: *On a Measurement-Free Quantum Lambda Calculus with Classical Control*, by Dal Lago, Masini, Zorzi, in Mathematical Structures in Computer Science, Volume 19, Issue 02, Cambridge University Press, 2009 [30].

SQ calculus

Starting from the Q calculus, we give an implicit characterization of polytime quantum complexity classes by means of a second λ-calculus, that we call SQ. SQ is an untyped quantum lambda calculus where the well formation rules are strongly related with Lafont's Soft Linear Logic [63]. The language is not built on the basis of an explicit notion of polynomial bounds, not even on any concrete polytime machine , in fact it is a *machine independent* and a *resource free* calculus; therefore it is completely in the spirit of the so called Implicit Computational Complexity approach.

The correspondence with quantum complexity classes is an extensional correspondence, proved by showing that:

- on one side, any term in the language can be evaluated in polynomial time (where the underlying polynomial depends on the *box depth* of the considered term);
- on the other side, any problem P decidable in polynomial time (in a quantum sense) can be represented in the language, i.e., there exists a term M which decides P.

SQ is sound and complete w.r.t. polynomial time quantum Turing machine [22, 73, 74] and so we will restrict our attention to the subclass of the so called computable operators (see Definition 2.21 and [22, 73, 74]). Showing polytime completeness requires a relatively non-standard technique based on the Yao's encoding of QTM into quantum circuit families [104]).

As a subsystem of Q (terms and configurations of SQ form subclasses of the ones of Q), SQ follows the classical control-quantum data paradigm and, as s for Q , SQ is a measurement free calculus.

This part is based on the paper *Quantum Implicit Computational Complexity* by Dal Lago, Masini, Zorzi, accepted with minor revision revision in Theoretical Computer Science [64].

Q^* **calculus** We propose an extension of Q , called Q*: in Q*, it is possible perform measurements of the quantum data, obtaining as result classical data that can be used in the subsequent steps of the computation. Then we provide a qualitative and quantitative study about computations with measurement.
As previously written, the possibility to perform a measurement during a computation is an useful feature for encoding quantum algorithms (such as Shor's algorithm), because measurement outputs are allowed to influence subsequently steps of the calculus.
The implementation of algorithms is outside the scope of this thesis. Our concern is on the theoretical study of computations with measurement. Measurement breaks the deterministic evolution of the calculus, forcing a probabilistic nature into the computation.
What it happen to good properties such as confluence when we add the measurement operator? Is it possible to combine a suitable notion of confluence with the probabilistic behavior imported by the measurement into the computation?
We will investigate the problems by developing a technical framework in which we give an innovative confluence proof where also the case of infinite computations is tackled.
We introduce the notion of *probabilistic computation*, for which we provide a strong confluence result, taking into account, in lieu of a single result, probability distributions of results called *mixed states* (an adaptation of the notion of mixed states of quantum mechanics). Then, the notion of computation is extended to mixed states too (we will call such a computations *mixed*), and also for mixed computations a confluence theorem is proved.
This part is based on the paper *Confluence Results for a Quantum Lambda Calculus*, by Dal Lago, Masini, Zorzi, in Proceeding of 6th QPL, Oxford, UK, 2009 (to appear in Electronics Note in Theoretical Computer Science).

1.2.2 Modal Deduction Systems

In the last part of the present work, we will present new results about modal logics and quantum computing. We call this last investigation a *variation on the main theme*, because we left higher-order characterization of computable functions, moving toward a qualitative logical analysis of quantum states (or equivalently quantum registers) transformations.

MSQS and **MSpQS**

We propose a modal labeled natural deduction system, called MSQS, which describe how a quantum state is transformed into another one by means of the fundamental operations as unitary transformations and total measurement.
We are not in interest in the internal structure of quantum state, but we represent it in an abstract, qualitative, way.

We propose a logical treatment of total measurement and unitary transformations by means of suitable modal operators, and we propose also a Kripke style semantics for the calculus.
The Kripke style semantics describes quantum states transformations in term of accessibility relations between worlds and then we prove that MSQS is *sound and complete* with respect to this semantics.
We also prove a normalization result for MSQS proofs, and as a consequence of normalization theorem, we prove that MSQS enjoys subformula property. By means of subformula property, it is possible to give a purely syntactical proof of consistency of the system.

We propose also a system called MSpQS, that is a variant of MSQS; it takes into account general measurement, rather than the total one. All the good properties of MSQS still hold. So, MSpQS is sound and complete with respect to the given Kripke style semantics; MSpQS enjoys normalization, subformula property, and as a consequence it is consistent.
This part is partially based on the paper *A Qualitative Modal Representation of Quantum Register Transformations*, by Masini, Viganò, Zorzi, in Proceedings of the 38th IEEE International Symposium on Multiple-Valued Logic (ISMVL 2008) [67].

Structure of the thesis

Part I - Background

In Chapter 2 we will recall some basic notion about Hilbert Spaces, Linear Logic (and its 'light' versions), Lambda Calculus and Modal Deduction Systems.
We will largely use Hilbert Space in Chapter 3, where we will give technical instrument about Quantum Computing, stressing on quantum computational models.

Part II - Main Theme: Quantum Lambda Calculi

In Chapter 4 we will develop the operational study of Q, carrying on in Chapter 5 with the expressiveness study.
In Chapter 6 we will propose the polytime system SQ .
In Chapter 7 we will study infinite quantum computation by Q^*, the quantum lambda calculus with explicit measurement.

Part III - A Variation on the Theme: Modal Deduction Systems

In Chapter 8 we will propose the two modal labeled deduction system MSQS and MSpQS.

Part I

Background

2

Mathematical Framework and Basic Logical Instruments

In this chapter we introduce some mathematical and logical notions. The aim of the following sections is to give a short account of the basic theoretical notions used in the advanced part of the present work.
Since we are working in the foundational approach of quantum computation, we will use several different technical framework. Firstly we will introduce the algebraic formulation of quantum mechanics, in terms of Hilbert Space and unitary operators; it provides a complete and precise mathematical description of quantum states, that will be largely used in our computational study.
Secondly, we will summarize fundamental issue about Linear Logic, Lambda Calculus and Modal Logic.

2.1 Spaces and Linear Operators

The results and the notions recalled in this section are mainly based on the following references: S. Roman, *Advanced in Linear Algebra* [81], S. Mac Lane, G. Birkoff, *Algebra* [65].

Inner Product Spaces

Definition 2.1 (Complex inner product space). *A complex inner product space is a vector space on the field $\mathbb{C}$ equipped with a function $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{C}$ that satisfies the following properties:*

1. $\langle \phi, \psi \rangle = \langle \psi, \phi \rangle^*$;
2. $\langle \psi, \psi \rangle$ *is a non negative real number;*
3. *if* $\langle \psi, \psi \rangle = 0$ *then* $\psi = \mathbf{0}$
4. $\langle c_1\phi_1 + c_2\phi_2, \psi \rangle = c_1^*\langle \phi_1, \psi \rangle + c_2^*\langle \phi_2, \psi \rangle$;
5. $\langle \phi, c_1\psi_1 + c_2\psi_2 \rangle = c_1\langle \phi, \psi_1 \rangle + c_2\langle \phi, \psi_2 \rangle$.

In the sequel we will often call *inner product space* a *complex inner product space*. Let V be an inner product space. Given $u, v \in V$, we say that u and v are **orthogonal**, written

$u \perp v$, if $\langle u, v \rangle = 0$; given X and Y subset of V, we say that X and Y are **orthogonal** if for all $x \in X$ and $y \in Y$, $x \perp y$.

Definition 2.2. *Given an inner product space V, a nonempty set U of vector is said* ***orthogonal set*** *if for all $u_i, u_j \in U$, if $u_i \neq u_j$ then $u_i \perp u_j$.*

If each u_i is also a normalized vector (see Definition2.6), then U is an ***orthonormal set****.*

We use the orthonormality in order to define the so called Hilbert Basis, that will be used in the discussion on Hilbert Space.

The notion of Hilbert basis must not be confused with the usual notion of basis for a vector space. The basis for a vector space is the **Hamel basis**, strongly related to the notion of *span*.

Definition 2.3 (Span). *Let P be an inner-product space and let be $S \subset P$, the span of S is the inner product subspace of P defined by*

$$span(S) = \left\{ \sum_{i=1}^{n} c_i s_i \; | n \in \mathbb{N}, c_i \in \mathbb{C}, s_i \in S \right\}.$$

If $V = span(S)$ we say that S **generate** V, and the elements of S are called *generators*. Note that even if P is an Hilbert space (see Definition 2.14), *span*(S) is not necessary an Hilbert Space.

We give now the formal definitions of Hamel basis:

Definition 2.4 (Hamel Basis). *Let V be an inner product space and let S be a linearly independent subset of V; S is an* Hamel basis *if and only if $V = span(S)$.*

So, an Hamel basis is a maximal linearly independent set of generators, that can be orthonormal or not at all.

Hilbert basis is defined in the following way:

Definition 2.5 (Hilbert basis).

Let V be an inner product space. A maximal orthonormal set in V is called a Hilbert Basis for V.

By Zorn's lemma, such maximal orthonormal set V always exists.

The Hamel basis and the Hilbert basis induce two different definition of *dimension* of the space. The dimension induced by the Hamel basis is the cardinality of the Hamel basis itself, whereas the *Hilbert dimension* is the cardinality of the Hilbert basis. In the finite dimensional case, the two definitions coincide, but this is not the case of infinite dimensional spaces. In Example 2.15 we will give an explanation of the differences between the two concepts.

Definition 2.6 (Norm and unit vectors). *Let V be a inner product space.*
For $v \in V$, the non negative real number

$$||v|| = \langle v, v \rangle^{1/2}$$

is called the norm of V.

A vector $v \in V$ is a unit *or* normalized *vector if its norm is equal to 1.*

Proposition 2.7 (Basic properties of the norm).
The norm $||\cdot||$ for a inner product space V enjoys the following properties:
1. $||v|| \geq 0$;
2. $||v|| = 0$ *if and only if* $v = 0$;
3. *for all* $u, v \in V$, $|\langle u, v\rangle| \leq ||u|| ||v||$;
4. *for all* $u, v \in V$, $||u+v|| \leq ||u|| + ||v||$;
5. *for all* $u, v, w \in V$, $||u-v|| \leq ||u-w|| + ||w-v||$;
6. *for all* $u, v \in V$, $|\, ||u|| - ||v|| \,| \leq ||u-v||$;
7. *for all* $u, v \in V$, $||u+v||^2 + ||u-v||^2 = 2||u||^2 + 2||v||^2$.

We use the norm in order to define the distance between two vectors in the inner product space:

Definition 2.8 (Distance). *Let V be a inner product space.*
We call distance *a binary relation on V with the following properties:*
1. $d(u,v) \geq 0$ *and* $d(u,v) = 0$ *if and only if* $u = v$;
2. $d(u,v) = d(v,u)$;
3. $d(u,v) \leq d(u,w) + d(w,v)$.

Proposition 2.9. *Let V be an inner product space. For any $u, v \in V$, the relation d defined by*

$$d(u,v) \equiv ||u-v||$$

is a distance, called distance induced by the inner product.

The distance d as defined in 2.8, is also called *metric*.

Tensor product

Let U and V be two finite dimensional inner product spaces, with inner products $\langle\cdot,\cdot\rangle_U$ and $\langle\cdot,\cdot\rangle_V$ respectively. Let $F_{U\times V}$ be the inner product space freely generated by linear combination of element in the basis $U \times V$ ($U \times V$ is exactly the cartesian product of sets).
We consider now a subspace S of $F_{U\times V}$ generated by all the vector in the form
$\alpha(u_1, v) + \beta(u_2, v) - (\alpha u_1 + \beta u_2, v)$ and $\alpha(u, v_1) + \beta(u, v_2) - (u, \alpha v_1 + \beta v_2)$
where $\alpha, \beta \in \mathbb{C}$, $u_1, u_2, u \in U$ and $v, v_1, v_2 \in V$.
We define the tensor product of the inner space U and V as

$$U \otimes V = F_{U\times V}/S$$

i.e. as the quotient space respect to the cosets $S + (u,v)$, with $(u,v) \in U \times V$.
We denote the coset $S + (u,v)$ by $u \otimes v$.
The inner product of $U \otimes V$ is defined by:

$$\langle u_1 \otimes v_1, u_2 \otimes v_2\rangle_{U\otimes V} = \langle u_1, u_2\rangle_U \langle v_1, v_2\rangle_V$$

with the properties of linearity and antilinearity (in the second and in the first component respectively) as in Definition 2.1.
Given a basis $\mathscr{B}$ for U and a basis $\mathscr{C}$ for V, the set $\{b \otimes c | b \in \mathscr{B}, c \in \mathscr{C}\}$ is a basis for $U \otimes V$.

Proposition 2.10. *The map* $\otimes : F_{U\times V} \to U \otimes V$ *defined by* $(u, v) \mapsto S + (u, v)$ *is a bilinear map.*

We have just give a so called *coordinate free* definition of tensor product.
In quantum computing is very common to define tensor product in a more intuitive but mathematically less precise way. We recall it in the following proposition:

Proposition 2.11. *Let* U *and* V *be two finite dimensional complex inner product spaces and let* $\mathscr{B}_1$ *and* $\mathscr{B}_2$ *be basis for* U *and* V *respectively.*
The tensor product space $P = U \otimes V$ *is the space generated by*
$\mathscr{B} = \{e_i^1 \otimes e_j^2 | e_i^1 \in \mathscr{B}, e_j^2 \in \mathscr{B}_2\}$ *with inner product defined by*
$\langle u_1 \otimes v_1, u_2 \otimes v_2\rangle_{U\otimes V} = \langle u_1, u_2\rangle_U \langle v_1, v_2\rangle_V$.

The previous construction of tensor product is useful but it is less general respect to the other one, because it is strongly related to the choice of the basis $\mathscr{B}_1$ and $\mathscr{B}_2$.

Hilbert Space

The mathematical framework of Quantum Mechanics involves particular inner product spaces, called Hilbert spaces. Hilbert spaces enjoy the property of *completeness*, induced by the distance defined in 2.8.

For this aim, we introduce the central notion of *Cauchy sequence*.

Definition 2.12 (Cauchy Sequence).

Let V *be an inner product space with metric* d*. A sequence* $(\phi_n)_{n<\omega}$ *is a* Cauchy sequence *if for all* $\epsilon > 0$*, there exists* $N > 0$ *such that for all* $n, m < N$ *we have* $d(\phi_n, \phi_m) < \epsilon$.

It is easy to show that any convergent sequence is a Cauchy sequence. The converse, if it holds, gives the following definition:

Definition 2.13 (Completeness).

Let V *be an inner product space;* V *is said to be* complete *if any Cauchy sequence in* V *converges in* V.

Inner product spaces complete respect to the distance plays a central rôle and are called Hilbert space.

Definition 2.14 (Hilbert Space). *An Hilbert space* $\mathcal{H}$ *is a complex inner product space that is complete respect to the distance induced by the inner product.*

One of the most important example of Hilbert space is ℓ^2, the set of the sequence $x = (x_n)_n$ on $\mathbb{C}$ such that $\sum_{n=1}^{\infty} |x_n|^2 < \infty$.

We give an example on ℓ^2 in order to exploit the difference between Hamel and Hilbert basis.

Example 2.15. Let M be the set of the vectors of ℓ^2 in the form
$e_i = (0 \dots 1 \dots 0 \dots)$ where e_i has a 1 in the ith component and 0 elsewhere.
M is an orthonormal set and can be easily showed that it is maximal too, then it his an

Hilbert basis of ℓ^2.
But it is not a Hamel basis for ℓ^2, in fact M is a minimal set of generators for $S = span(M) \subsetneq \ell^2$, the subspace of all sequence in ℓ^2 that have finite support, and $S \neq \ell^2$.

As a consequence of Zorn's Lemma, every nontrivial Hilbert space has an Hilbert basis, i.e a maximal orthonormal set.

Unitary operators

Very important class of operators in quantum theory is given by the so called *unitary operators*. Quantum Computing is essentially based on the application of unitary operators to normalized vectors of the space $\ell^2(\mathcal{S})$ (see below Section 2.1.1), i.e. on quantum registers.

Definition 2.16 (Unitary operators). *Let $\mathcal{H}$ be an Hilbert space, and let $\mathbf{U} : \mathcal{H} \to \mathcal{H}$ be a linear transform. The* adjoint *of $\mathbf{U}$ is the unique linear transform $\mathbf{U}^\dagger : \mathcal{H} \to \mathcal{H}$ such that for all ϕ, ψ*
$\langle \mathbf{U}\phi, \psi \rangle = \langle \phi, \mathbf{U}^\dagger \psi \rangle$. If $\mathbf{U}^\dagger \mathbf{U}$ is the identity, we say that $\mathbf{U}$ is a unitary operator.

The tensor product of unitary operators is defined as follows:

Definition 2.17. *Let $\mathbf{U} : \mathcal{H}_1 \to \mathcal{H}_1$ and $\mathbf{W} : \mathcal{H}_2 \to \mathcal{H}_2$ be two unitary operators. The linear unitary operator $\mathbf{U} \otimes \mathbf{W} : \mathcal{H}_1 \otimes \mathcal{H}_2 \to \mathcal{H}_1 \otimes \mathcal{H}_2$ is defined by*

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

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

2.1.1 The fundamental Hilbert space $\ell^2(\mathcal{S})$

We use now ℓ^2 to give the description of the Hilbert space $\ell^2(S)$, from which we can extract several useful spaces as particular cases.

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

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

equipped with:

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

[1] in order the inner product definition make sense we should prove that the sum $\sum_{s \in \mathcal{S}} \phi(s)^* \psi(s)$ converges, see [81].

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

In the thesis. we will call *quantum state* or *quantum register* any normalized vector in $\ell^2(\mathcal{S})$.

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

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

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

It is now interesting to distinguish two cases:

1. $\mathcal{S}$ is *finite*: in this case $\mathcal{B}(\mathcal{S})$ is also an orthonormal (Hamel) basis of $\ell^2(\mathcal{S})$ and consequently $span(\mathcal{B}(\mathcal{S})) = \ell^2(\mathcal{S})$. $\ell^2(\mathcal{S})$ is isomorphic to $\mathbb{C}^{|\mathcal{S}|}$. With a little abuse of language we say also that $\ell^2(\mathcal{S})$ is "generated" by $\mathcal{S}$.

2. $\mathcal{S}$ is *denumerable*: in this case it is easy to show that $\mathcal{B}(\mathcal{S})$ is an Hilbert basis of $\ell^2(\mathcal{S})$, but *it is not* an Hamel basis. In fact let us consider the subspace $span(\mathcal{B}(\mathcal{S}))$. We see immediately that $span(\mathcal{B}(\mathcal{S})) \subsetneq \ell^2(\mathcal{S})$ is an inner-product infinite dimensional space with $\mathcal{B}(\mathcal{S})$ as Hamel basis [2], but $span(\mathcal{B}(\mathcal{S}))$ *is not* an Hilbert space because it is not complete.
   There is a strong relationship between $span(\mathcal{B}(\mathcal{S}))$ and $\ell^2(\mathcal{S})$, in fact it is possible to show (this is a standard result [81]) that $span(\mathcal{B}(\mathcal{S}))$ is a dense subspace of $\ell^2(\mathcal{S})$, and that $\ell^2(\mathcal{S})$ is the (unique!) *completion* of $span(\mathcal{B}(\mathcal{S}))$. This fact is important because in the main literature on Quantum Turing Machines, unitary transforms are usually defined on spaces like $span(\mathcal{B}(\mathcal{S}))$, but this could be problematic because $span(\mathcal{B}(\mathcal{S}))$ *is not a quantum space*. This is not a concrete problem, in fact it is possible to show that each unitary operator $\mathbf{U}$ in $span(\mathcal{B}(\mathcal{S}))$ has a standard extension in $\ell^2(\mathcal{S})$.

2.1.2 Two important finite dimensional Hilbert Spaces

In the rest of the thesis, the following spaces are extensively used.

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

Let be $S = \{0,1\}^n$, i.e. S is the set of the finite binary strings of length n. The Hilbert space $\mathcal{H}(\mathcal{S})$ is the standard space used in the field of quantum computing. This kind of space is useful to describe bits, qubits, and quantum registers.

For example, let us consider $S = \{0,1\}^2$. The computational basis of $\ell^2(S)$ is $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, and a generic quantum register may be expressed, in the computational basis as $\alpha_1|00\rangle + \alpha_2|01\rangle + \alpha_3|10\rangle + \alpha_4|11\rangle$, where $\sum_i |\alpha_i|^2 = 1$.

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

Let $\mathcal{V}$ be a set of names and let be $S = \{f | f : \mathcal{V} \to \{0,1\}\}$ (S is the set of classical valuation into the set $\{0,1\}$). $\ell^2(S)$ is an Hilbert space of dimension $2^{\#\mathcal{V}}$. In the following we will shorten $\ell^2(\{0,1\}^{\mathcal{V}})$ with $\mathcal{H}(\mathcal{V})$.

[2] $span(\mathcal{B}(\mathcal{S}))$ contains all the functions of $\ell^2(\mathcal{S})$ that are almost everywhere 0.

As we will see, this space is useful to describe quantum registers when we want assigning names to qubits, when they are no referred by means of their ordinal position (as it usually happens in literature).
The set of quantum registers are *normalized vectors* of $\mathcal{H}(\mathcal{V})$: by definition, a quantum register will be a function $\phi : \{0,1\}^{\mathcal{V}} \to \mathbb{C}$ such that $\sum_{f\in\{0,1\}^{\mathcal{V}}} |\phi(f)|^2 = 1$ (*normalization condition*).
The space $\mathcal{H}(\mathcal{V})$ is equipped with the *orthonormal basis* $\mathcal{B}(\mathcal{V}) = \{|f\rangle : f \in \{0,1\}^{\mathcal{V}}\}$ We call *standard* or *computational* such a basis. For example, the standard basis of the space $\mathcal{H}(\{p,q\})$ is $\{|p \mapsto 0, q \mapsto 0\rangle, |p \mapsto 0, q \mapsto 1\rangle, |p \mapsto 1, q \mapsto 0\rangle,$
$|p \mapsto 1, q \mapsto 1\rangle\}$.
Let $\mathcal{V}' \cap \mathcal{V}'' = \emptyset$. With $\mathcal{H}(\mathcal{V}') \otimes \mathcal{H}(\mathcal{V}'')$ we denote the tensor product (defined in the usual way) of $\mathcal{H}(\mathcal{V}')$ and $\mathcal{H}(\mathcal{V}'')$. If $\mathcal{B}(\mathcal{V}') = \{|f_i\rangle : 0 \leq i < 2^n\}$ and $\mathcal{B}(\mathcal{V}'') = \{|g_j\rangle : 0 \leq j < 2^m\}$ are the orthonormal bases respectively of $\mathcal{H}(\mathcal{V}')$ and $\mathcal{H}(\mathcal{V}'')$ then $\mathcal{H}(\mathcal{V}') \otimes \mathcal{H}(\mathcal{V}'')$ is equipped with the orthonormal basis
$\{|f_i\rangle \otimes |g_j\rangle : 0 \leq i < 2^n, 0 \leq j < 2^m\}$. We will abbreviate $|f\rangle \otimes |g\rangle$ with $|f, g\rangle$. If $\mathcal{V}$ is a qvs, then $I_{\mathcal{V}}$ is the identity on $\mathcal{H}(\mathcal{V})$, which is clearly unitary. It is easy to show that if $\mathcal{V}' \cap \mathcal{V}'' = \emptyset$ then there is a standard *isomorphism* i_s:

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

In the rest of the paper we will assume to work up-to such an isomorphism[3].
In particular if $\mathcal{Q}' \in \mathcal{H}(\mathcal{V}')$ and $\mathcal{Q}'' \in \mathcal{H}(\mathcal{V}'')$ are two quantum registers, with a little abuse of language (authorized by the isomorphism defined above) we will say that $\mathcal{Q}' \otimes \mathcal{Q}''$ is a quantum register in $\mathcal{H}(\mathcal{V}' \cup \mathcal{V}'')$. In the rest of the paper we will denote with the scalar 1 the empty quantum register (that belongs to $\mathcal{H}(\emptyset)$).

Let $u \in \mathcal{H}(\{0,1\}^n)$ be the quantum register $u = \alpha_1|0\ldots0\rangle + \ldots +$
$\alpha_{2^n}|1\ldots1\rangle$ and let $\langle q_1,\ldots,q_n\rangle$ be a sequence of names. $u^{\langle q_1,\cdots,q_n\rangle}$ is the quantum register in $\mathcal{H}(\{q_1,\ldots,q_n\})$ defined by $u^{\langle q_1,\cdots,q_n\rangle} = \alpha_1|q_1 \mapsto 0,\ldots,q_n \mapsto 0\rangle + \ldots +$
$\alpha_{2^n}|q_1 \mapsto 1,\ldots,q_n \mapsto 1\rangle$.
Let $\mathbf{U} : \mathcal{H}(\mathcal{V}) \to \mathcal{H}(\mathcal{V})$ be an elementary operator and let $\langle q_1,\ldots,q_n\rangle$ be any sequence of distinguished names in $\mathcal{V}$. Considering the bijection between $\{0,1\}^n$ and $\{0,1\}^{\{q_1,\ldots,q_n\}}$, $\mathbf{U}$ and $\langle q_1,\ldots,q_n\rangle$ induce an operator
$\mathbf{U}_{\langle q_1,\ldots,q_n\rangle} : \mathcal{H}(\{q_1,\ldots,q_n\}) \to \mathcal{H}(\{q_1,\ldots,q_n\})$ defined as follows: if
$|f\rangle = |q_{j_1} \mapsto b_{j_1},\ldots,q_{j_n} \mapsto b_{j_n}\rangle$ is an element of the orthonormal basis of $\mathcal{H}(\{q_1,\ldots,q_n\})$, then

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

where $q_{j_i} \mapsto b_{j_i}$ means that to the qubit named q_{j_i} we associate the element b_{j_i} of the basis.
Let $\mathcal{V}' = \{q_{j_1},\ldots,q_{j_k}\} \subseteq \mathcal{V}$. We naturally extend (by suitable standard isomorphisms) the unitary operator $\mathbf{U}_{\langle q_{j_1},\ldots,q_{j_k}\rangle} : \mathcal{H}(\mathcal{V}') \to \mathcal{H}(\mathcal{V}')$ to the unitary operator $\mathbf{U}_{\langle\langle q_{j_1},\ldots,q_{j_k}\rangle\rangle} : \mathcal{H}(\mathcal{V}) \to \mathcal{H}(\mathcal{V})$ that acts as the identity on variables not in $\mathcal{V}'$ and as $\mathbf{U}_{\langle q_{j_1},\ldots,q_{j_k}\rangle}$ on variables in $\mathcal{V}'$.

[3] in particular, if $\mathcal{Q} \in \mathcal{H}(\mathcal{V})$, $r \notin \mathcal{V}$ and $|r \mapsto c\rangle \in \mathcal{H}(\{r\})$ then
$\mathcal{Q} \otimes |r \mapsto c\rangle$ will denote the element $i_s(\mathcal{Q} \otimes |r \mapsto c\rangle) \in \mathcal{H}(\mathcal{V} \cup \{r\})$

Example 2.18. Let us consider the standard operator $\mathbf{cnot} : \ell^2(\{0,1\}^2) \to \ell^2(\{0,1\}^2)$, defined by

$$\mathbf{cnot}|00\rangle = |00\rangle \qquad \mathbf{cnot}|10\rangle = |11\rangle$$
$$\mathbf{cnot}|01\rangle = |01\rangle \qquad \mathbf{cnot}|11\rangle = |10\rangle$$

The **cnot** operator is one of the most important quantum operator (see also Example 3.4).
Intuitively, the **cnot** operator complements the target bit (the second one) if the control bit is 1, and otherwise does not perform any action. Let us fix the sequence $\langle p, q\rangle$ of variables, $\mathbf{cnot}$ induces the operator

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

such that:

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

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

In this thesis, in order to prove an equivalence result between the one of the proposed calculi and the formalism of quantum circuits families (see Chapter 3 for the definition of quantum circuits families), we must restrict the class of possible unitary operators to an effectively enumerable class (see, e.g., the paper of Nishimura and Ozawa [74] on the perfect equivalence between quantum circuit families and Quantum Turing Machines).

A particular interesting effectively enumerable class of unitary operators is given by the so called *computable operators*.

Some definition on computability are in order now.

Definition 2.19.

1. *A real number $x \in \mathbb{R}$ is* computable *iff there is a Deterministic Turing Machine which on input 1^n computes a binary representation of an integer $m \in \mathbb{Z}$ such that $|m/2^n - x| \leq 1/2^n$. Let $\tilde{\mathbb{R}}$ be the set of computable real numbers.*
2. *A real number $x \in \mathbb{R}$ is* polynomial-time computable *iff there is a Deterministic Polytime Turing Machine which on input 1^n computes a binary representation of an integer $m \in \mathbb{Z}$ such that $|m/2^n - x| \leq 1/2^n$. Let $\mathbf{P}\mathbb{R}$ be the set of polynomial time real numbers.*

Definition 2.20.

1. *A complex number* $z = x + iy$ *is* computable *iff* $x, y \in \tilde{\mathbb{R}}$. *Let* $\tilde{\mathbb{C}}$ *be the set of computable complex numbers.*
2. *complex number* $z = x + iy$ *is* polynomial-time computable *iff* $x, y \in \mathbf{P}\mathbb{R}$. *Let* $\mathbf{P}\mathbb{C}$ *be the set of polynomial time computable complex numbers.*
3. *a normalized vector* ϕ *in any Hilbert space* $\ell^2(\mathcal{S})$ *is* computable *(*polynomial computable*) if the range of* ϕ *(a function from* $\mathcal{S}$ *to complex numbers) is* $\tilde{\mathbb{C}}$ *(*$\mathbf{P}\mathbb{C}$*).*

Now we can define the computable unitary operators:

Definition 2.21 (Computable Operators). *A unitary operator* $\mathbf{U} : \ell^2(\mathcal{S}) \to \ell^2(\mathcal{S})$ *is computable if for every computable normalized vector* ϕ *of* $\ell^2(\mathcal{S})$*,* $\mathbf{U}(\phi)$ *is computable.*

2.2 Logics and Lambda Calculi

2.2.1 Linear Logic

Linear logic (LL) [47] was introduced by Jean-Yves Girard in 1987: it allows to reason about the use of resources, it was a fundamental starting point for new perspective in Proof Theory studies and it has a lot of interesting applications in Computer Science. Linear Logic is both a decomposition and a refinement of classical logic. Starting from classical logic, the refinement procedure begins with the elimination of the structural rules (weakening, contraction and exchange), then, we re-enter the original expressive power by two modalities or *exponential* '!' and '?', which control the duplicability of resources and express the iterability of an action: in other word Linear Logic allows structural rules for a restricted class of exponential prefixed formulas.

Girard uses the modality ! to decompose intuitionistic implication in more primitive elements: the implication $A \to B$ is decomposed as $!(A) \multimap B$ where $\multimap$ is the *linear implication*.

In Linear Logic two different kinds of conjunctions, $\otimes$ and $\&$, coexist; $\otimes$ and $\&$ correspond to different meaning (multiplicative and additive) of the connective 'and'. Dually, the multiplicative disjunction $\parr$ corresponds to the conjunction $\otimes$ and the additive disjunction $\oplus$ corresponds to the conjunction $\&$; moreover LL is equipped with an involutive *linear negation* $(\cdot)^\perp$. In LL $(A \otimes B)^\perp = A^\perp \parr B^\perp$ and $(A \parr B)^\perp = A^\perp \otimes B^\perp$ holds.

Now we briefly recall in Figure 2.1 the so called multiplicative-exponential fragment based on $\otimes$, $\parr$, !, ?: it will be the basis for our quantum calculi proposed in Chapter 4, 5, 6, 7.

The symbols 1 and $\perp$ are the neutral element of $\otimes$ and $\parr$ respectively, and $1^\perp = \perp$, $\perp^\perp = 1$ holds; moreover, by means of linear negation and disjunction, the linear implication can be written as $A \multimap B = A^\perp \parr B$.

A complete explanation of Linear Logic can be found in [47].

In this thesis we will not use Linear Logic directly: we will propose some untyped calculi whose well formation rules are strongly based on a fragment of Linear Logic, such as fragment formulation introduced by Wadler in [100] (see Section 2.2.2) and the Soft Linear Logic introduced by Lafont [63] (see Section 2.2.3). Moreover, the implicit complexity calculus proposed in Chapter 6 is inspire to the *affine* version of Linear Logic, following the so called light logics implicitly complexity approach.

$$\frac{}{\vdash A, A^{\perp}}\,\mathsf{id} \qquad \frac{\vdash \Gamma, A \quad \vdash A^{\perp}, \Delta}{\vdash \Gamma, \Delta}\,\mathsf{cut} \qquad \frac{\vdash \Gamma_1, \Gamma_2}{\vdash \Gamma_2, \Gamma_1}\,\mathsf{ex}$$

$$\frac{}{\vdash 1}\,\mathsf{one} \qquad \frac{\vdash \Gamma}{\vdash \Gamma, \bot}\,\mathsf{false}$$

$$\frac{\Gamma, A \quad \vdash B, \Delta}{\vdash \Gamma, A \otimes B, \Delta}\,\mathsf{times} \qquad \frac{\vdash \Gamma, A, B}{\vdash \Gamma, A \parr B}\,\mathsf{par}$$

$$\frac{\vdash ?\Gamma, A}{\vdash ?\Gamma, !A}\,\mathsf{ofcourse} \qquad \frac{\vdash \Gamma}{\vdash \Gamma, ?A}\,\mathsf{weakening}$$

$$\frac{\vdash \Gamma, A}{\vdash \Gamma, ?A}\,\mathsf{dereliction} \qquad \frac{\vdash \Gamma, ?A, ?A}{\vdash \Gamma, ?A}\,\mathsf{contraction}$$

Fig. 2.1. Linear Logic Rules

2.2.2 The Lambda Calculus

The λ-calculus, introduced by Alonzo Church in the 1930's, is a formal system designed to investigate computability from a pure functional perspective.
The λ-calculus was born as instrument for the foundational study of mathematics but successively it had a great influence in theoretical Computer Science (see e.g. type theory and the so called Curry-Howard isomorphism) [53]. It can be considered a paradigmatic functional programming language.
The λ-calculus, in its pure formulation (the untyped one), is Turing-complete and its main features is that it is higher-order; that is, it gives a systematic notation whose "input" and "output" may be other functions.
We recall here the main notation, for an exhaustive treatment see e.g. [16].

Syntax

Given the alphabet $\{\lambda, (,), x, y, z \ldots\}$, where $x, y, z \ldots$ are variables in a given set $\mathcal{V}$, the set Λ of λ-terms is generated by the following grammar:

$$M := x | M_1(M_2) | \lambda x.N$$

where $x \in \mathcal{V}$, $M_1(M_2)$ is the *functional application* and $\lambda x.N$ is the λ-*abstraction*. The λ-abstraction construct puts in evidence the variable x, and says that the term M is a function of x; x is a formal parameter that can be substituted with an argument that the function M can takes in input.

Notation and basic definitions on λ-terms

We give in the following some standard syntactic notions and notation.

$M, N, L, P, Q, \ldots$ will denote arbitrary λ-terms. We will use the symbol $\equiv$ to denote syntactic equality.

Moreover, to avoid ambiguity, we refer to the following notation:
$\lambda x_1 x_2 \dots x_n.N \equiv \lambda x_1(\lambda x_2(\dots((\lambda x_n(N)))\dots))$ (λ-abstraction associates on the right) and $N_1 N_2 \dots N_n \equiv (\dots((N_1 N_2)\dots N_n)$ (application associate to the left).

Let $M \in \Lambda$. We say that a variable occurs *free* in M if it is not in the scope of a λ-abstraction, and we say that it occurs *bound* otherwise.
We define the set of free variable of a term as following:

Definition 2.22. *$FV(M)$ is the set of the free variable in M, inductively defined as*

$FV(x) = \{x\}$
$FV(\lambda x.N) = FV(N) - \{x\}$
$FV(N(L)) = FV(N) \cup FV(L)$

M is a *closed* term if $FV(M) = \emptyset$.
We write $M\{N/x\}$ for the *substitution* with N for the variable x in the term M.
Substitution of bound occurrence of variables induce a convention on terms.

Definition 2.23. *Let $M \in \Lambda$. A change of bound occurrences of a variable x in M is a substitution of a subterm $\lambda x.L$ of M by $\lambda y.(L\{y/x\})$, where $y \notin FV(L)$.*

If M_2 is obtained by M_1 by one ore more changes of bound variables we say that M_1 and M_2 are α-congruent ($M_1 \equiv M_2$). All α-congruent terms are considered syntactically equivalent. In this thesis we will work modulo α-congruence.

The notion of substitution of free variable is central in order to define the reduction relation.

Definition 2.24. *Let x a variable that occurs free in a term M. We define the substitution with the term N for free occurrences of x in M as*

$x\{N/x\} \equiv N$
$y\{N/x\} \equiv y$ *if* $x \neq y$
$(\lambda y.L)\{N/x\} \equiv (\lambda y.(L\{N/x\}))$ *if* $x \neq y$ *and* $y \notin FV(N)$
$(M_1 M_2\{N/x\} \equiv (M_1\{N/x\})(M_2\{N/x\})$

In the present work we will assume the following variable convention (see [16], p. 26)

Barendregt's Variable Convention
If $M_1, \dots, M_k$ occur in a certain mathematical context (such as definition, proof, ...), then in this terms all bound variables are chosen to be different from the free variables.

See [16] for other standard results on substitution.

Reduction and strategies

The computational step of lambda calculus, i.e. the evaluation process, consists in the plug of arguments into functions, and it is called β-reduction. β-reduction convert a λ-term in the form $(\lambda x.M)(N)$ (called *redex*), into the term $M\{N/x\}$ (called *reduct*); in this case we will write $(\lambda x.M)(N) \to_\beta M\{N/x\}$.
During the evaluation, we reduce lambda terms finding a redex and by replacing it with its reduct. More formally, we define β-reduction as following:

Definition 2.25 (One-step β-reduction). *The one-step β-reduction is the smallest relation $\rightarrow_\beta$ on term defined inductively by the following rules:*

$$\frac{}{(\lambda x.M)(N) \rightarrow_\beta M\{N/x\}} \qquad \frac{M \rightarrow_\beta M'}{\lambda x.M \rightarrow_\beta \lambda x.M'}$$

$$\frac{M \rightarrow_\beta M'}{MN \rightarrow_\beta M'N} \qquad \frac{M \rightarrow_\beta M'}{NM \rightarrow_\beta NM'}$$

The reflexive-transitive closure of the previous relation is defined in the following way:

Definition 2.26 (β-reduction).
The β-reduction is the smallest relation $\rightarrow_{\beta^*}$ on term defined inductively by the following rules:

$$\frac{}{M \rightarrow_\beta^* M} \qquad \frac{M \rightarrow_\beta N}{M \rightarrow_\beta^* N} \qquad \frac{M \rightarrow_\beta^* N \quad N \rightarrow_\beta^* L}{M \rightarrow_\beta^* L}$$

A term is said to be in *β-normal form* (respect to the β-reduction) if no application of the β rule is possible, i.e. the term does not contain any redex.
A term M *has* β-normal form if there exists N such that $M \rightarrow_\beta^* N$ and N a β-normal form.

β-reduction enjoys some important properties, such as the Church-Rosser property and the uniqueness of the normal form.
The first property says that the result of the computation is independent from the order in which the redexes of the lambda term are contracted.

Theorem 2.27 (Church-Rosser Theorem). *If $M \rightarrow_\beta^* N_1$ and $M \rightarrow_\beta^* N_2$, there exists a term L such that $N_1 \rightarrow_\beta^* L$ and $N_2 \rightarrow_\beta^* L$.*

As a corollary of the previous statement, we have the following

Corollary 2.28. *Each λ-term M has at most one β-normal form.*

Therefore, if a term has the β-normal form, then it must be unique.

Not all terms has normal form. In fact, there are some term that cannot be reduced to a normal form. For example, let be $\omega \equiv \lambda x.xx$ and $\mathbf{\Omega} \equiv \omega\omega$. Clearly, if $\mathbf{\Omega} \rightarrow_\beta M$, then $M \equiv \Omega$ and consequently $\mathbf{\Omega}$ has not β-nf.

Church-Rosser theorem and related results have some important consequences: (i) it is possible to prove, in a pure syntactically way, the consistency of the Lambda Calculus; (ii) in order to find the β-nf of a term M, if such β-nf exists, the various sub-expression of M can be reduced in different order.

See [16] for an complete and good treatment of the previous topics.

Lambda Calculus and Linearity

One of the main features of the calculi proposed in this thesis is *linearity*: this is not surprising, if we consider that linearity is a fundamental ingredient of many quantum computational models.
In the literature, several linear lambda calculi have been introduced, in order (via Curry-Howard isomorphism) to give a suitable syntax for Linear Logic proofs and to define an useful instrument 'resource conscious' inside the functional programming paradigm. Particularly interesting is the formulation of Linear Logic proposed by P. Wadler [100]. Such a formulation, that we give in Figure 2.2, will be the basis of the quantum lambda calculi proposed in this thesis.

$$\frac{}{x : A \vdash x : A}\ \mathsf{Id}$$

$$\frac{\Gamma \vdash t : A \quad x : A, \Delta \vdash u : B}{\Gamma, \Delta \vdash u : \{t/x\} : B}\ \mathsf{Cut} \qquad \frac{\Gamma, p : A \vdash u : B}{\Gamma, x : A \vdash (\text{let}\, p = x \text{ in } u) : B}\ \mathsf{Let}$$

$$\frac{\Gamma \vdash t : A \quad \Delta \vdash u : B}{\Gamma, \Delta \vdash (t, u) : (A \otimes B)}\ \otimes\mathsf{-R} \qquad \frac{\Gamma, p : A, q : B \vdash t : C}{\Gamma, (p, q) : (A \otimes B) \vdash t : C}\ \otimes\mathsf{-L}$$

$$\frac{\Gamma, p : A \vdash t : B}{\Gamma \vdash (\lambda p.t) : (A \multimap B)}\ (\multimap \mathsf{-R}) \qquad \frac{\Gamma \vdash t : A \quad y : B, \Delta \vdash u : C}{\Gamma, f : (A \multimap B), \Delta \vdash u : \{(ft)/y\} : C}\ \multimap \mathsf{-L}$$

$$\frac{!x_1 :!A_1, \dots, !x_n :!A_n \vdash t : B}{!x_1 :!A_1, \dots, !x_n :!A_n \vdash !t :!B}\ \mathsf{Promotion} \qquad \frac{\Gamma, z : A \vdash t : B}{\Gamma, !z :!A \vdash t : B}\ \mathsf{Dereliction}$$

Fig. 2.2. Wadler's formulation

Note that Wadler use a *let* rule, that has no logical content, and can be derived by other rules. An expression (let $p = x$ in u) can be considered as an abbreviation for $((\lambda p.u)x)$. Wadler's syntax is able to resolve some theoretical problem (the asymmetry of older formulation respect to LL semantics), using *pattern*. For each type of the language, there is a term to construct values of that type, and a pattern as a destructor of values of that type. The patterns are generate by the grammar

$$p, q := x|(p, q)|!x$$

and in the new formulation they are paired with types (instead of in the usual formulation types are paired with variables).
See the original paper [100] for a full discussion on the topic.

A type free linear λ-calculus

On the basis of the Wadler's calculus, it is possible to design a type free formulation of pure λ-calculus, where the notion of linearity is explicitly expressed in the syntax by using the modality '!'.

The syntax of terms is given by:

$$M := x|M(M)|\lambda x.M|\lambda !x.M|!(M)$$

where $\lambda x.M$ and $\lambda !x.M$ are the linear and the non linear lambda abstraction respectively. The well formation rules for untyped linear calculus are given in Figure 2.3:

$$\frac{}{!\Delta, x \vdash x} \qquad \frac{}{!\Delta, !x \vdash x} \qquad \frac{!\Delta \vdash M}{!\Delta \vdash !M}$$

$$\frac{\Lambda_1, !\Delta \vdash M \quad \Lambda_2, !\Delta \vdash N}{\Lambda_1, \Lambda_2, !\Delta \vdash M(N)} \qquad \frac{\Gamma, x \vdash M}{\Gamma \vdash \lambda x.M} \qquad \frac{\Gamma, !x \vdash M}{\Gamma \vdash \lambda !x.M}$$

Fig. 2.3. Linear λ-calculus well formation rules

An *environment* Γ is a (possibly empty) finite set in the form $\Lambda, !\Delta$, where Λ is a (possibly empty) set of variables, and $!\Delta$ denote a (possibly empty) set of patterns $!x_1, \ldots, !x_n$. We impose that in an environment, each classical variable x occurs at most once (either as $!x$ or as x). A *judgement* is an expression $\Gamma \vdash M$, where Γ is an environment and M is a term. We say that a judgement is *well-formed* if it is derivable by means of the *well-forming rules* in figure 2.3.
In the linear lambda calculus, as for Linear Logic, the bang (!) connective is used to control duplication and erasing of lambda terms, and we need to define beta-reduction and strategies taking in account this features.

The definition of the β-reduction have to deal with to presence of two kind of redex:

$$(\beta \multimap) \qquad (\lambda x.M)(N) \to M\{N/x\}$$
$$(\beta !) \qquad (\lambda !x.M)(!N) \to M\{N/x\}$$

with the following contextual closures:

$$\frac{M \to M'}{MN \to M'N} \qquad \frac{N \to N'}{MN \to MN'}$$
$$\frac{M \to M'}{\lambda x.M \to \lambda x.M'} \quad \frac{M \to M'}{\lambda !x.M \to \lambda !x.M'}$$

The reduction described is the so called *surface reduction* (see [91]): the remarkable point of such a kind of reduction is that no reduction occurs in the scope of a bang. We can say that a term in the form $!N$ represents a suspended computation: it will be eventually reduced after an evaluation by means of a $\beta !$ rule.

A term M is said to be in *surface normal form* if there is no surface reduction from M. In this thesis we will use surface reduction to evaluate quantum lambda terms in order to avoid violations of some important quantum computing principles (no cloning and no erasing properties).

2.2.3 Implicit Computational Complexity and Light Logics

The Computational Complexity is one of the fundamental topics of Computer Science: it analyzes the computation in terms of computational resources (time and space), with respect to well-know computational models such as Turing Machines, Circuits Families and so on.
Computational problems are classified in *complexity classes*, depending on the amount of computational resource required.

Many years ago, some researchers asked the following questions: is it possible to face the problem of Computational Complexity in a new way? Namely, is it possible, instead of to show that a problem or (more intensionally) an algorithm, belongs to a certain complexity class $\mathscr{C}$, to define calculi that guarantee the implicit belonging to $\mathscr{C}$ for the algorithms defined in it? This question opened the Implicit Computational Complexity approach (ICC) research area.
The main features of the ICC approach is that it is *machine independent*: there are no references to any computational models such as Turing Machine or Circuits Families. Another good property for an ICC system is the absence of any explicit reference to the complexity bound. A system which satisfies this second property is said to be *resource free*.

The seminal work of ICC approach (in the far 1965), was by A. Cobham [28], in the field of Recursion Theory; starting from the definition of *feasibility*, Cohbam gave the first recursion-theoretic characterization of FP (the class of polytime functions):

Theorem 2.29 (Cobham, 1964).
A function is in FP if and only if it is obtained by means of base function, composition and limited iteration on notation.

Cobham's computational formalism is the first machine-independent characterization of polytime functions, but it is no resource-free: in fact, it needs an explicit polynomial function (in order to bound the notation), in the syntax.
In 1992 Bellantoni and Cook [19], starting from Cobham's idea, developed the basis of *ramified arithmetic* with *safe recursion*, by defining a system that exactly capture the class FP an that is moreover resource free.

Another way to deal with Implicit Computational Complexity is based on proof-theoretic methods.
Jean-Yves Girard, in 1998 defined the Light Linear Logic [50], a subsystem of LL that captures exactly polytime function. LLL has a precursor system in the Bounded Linear Logic (1992) [52], developed by Girard himself; but if in Bound LL polynomials directly appear in the syntax, Light LL is a truly resource free system, and it should be considered the real founder of the (large) family of ICC logics.
Girard open his paper [50] with this claim: *"We are seeking a $\langle\langle$logic of polytime$\rangle\rangle$. Not yet one more axiomatization, but an intrinsically polytime system."*, where the expressive power of the system is given by the computational complexity of the cut elimination procedure. Girard's main breakthrough was to understand that the problem of exponential blowup (in time and space) of cut elimination is essentially caused by *structural rules* (in particular contraction, responsible, during the cut elimination process, of duplications of subproofs). In order to solve the problem, in the *light* version of Linear Logic, duplication is controlled by restricting exponential rules; consequently it is possible to master the expressive power (in the Curry-Howard sense) of the logical system.

Unfortunately, LLL has an heavy syntax and it is a quite complex system because it need extra modalities (besides ! and ?).
Then, few yeas ago other "light systems" have been defined, for example the Light Affine Logic by A. Asperti (with the unrestricted rule of weakening) [10], that is a suitable simplification and extension of Light LL and the Soft Linear Logic by Y. Lafont [63]. We focus our attention on the last one, because the intrinsically polytime quantum calculus defined in Chapter 6 is explicitly inspired to Lafont's work.

Soft LL has been developed on the basis of both Bound LL, and is given by the rules in Figure 2.4:

Identity and Cut

$$\frac{}{A \vdash A}\,\mathsf{Ax} \qquad \frac{\Gamma \vdash A \quad \Delta, A \vdash C}{\Gamma, \Delta \vdash C}\,\mathsf{Cut}$$

Logical Rules

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \multimap B}\,\vdash\multimap \qquad \frac{\Gamma \vdash A \quad \Delta, B \vdash C}{\Gamma, \Delta, A \multimap B \vdash C}\,\multimap\vdash \qquad \frac{}{\vdash 1}$$

$$\frac{\Gamma \vdash A \quad \Delta \vdash B}{\Gamma . \Delta \vdash A \otimes B}\,\vdash\otimes \qquad \frac{\Gamma, A, B \vdash C}{\Gamma, A \otimes B \vdash C}\,\otimes\vdash \qquad \frac{\Gamma \vdash C}{\Gamma, 1 \vdash C}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash \forall\alpha.A}\,\vdash\forall \qquad \frac{\Gamma, A[B/\alpha] \vdash C}{\Gamma, \forall\alpha.A \vdash C}\,\forall\vdash$$

Structural Rules

$$\frac{\Gamma, A, B, \Delta \vdash C}{\Gamma, A, B, \Delta \vdash C}\,\mathsf{exc}\vdash$$

Exponential Rules

$$\frac{\Gamma \vdash A}{!\Gamma \vdash !A}\,! \qquad \frac{\Gamma, A^{(n)} \vdash C}{\Gamma, !A \vdash C}\,\mathsf{mm}$$

Fig. 2.4. Soft Linear Logic

The right logical rule for $\forall$ has the usual side condition, i.e. there must be no free occurrences of α in Γ.

We write $A^{(n)}$ for the sequence $A, A, \ldots, A$ (n, possibly 0, times).

The key points of the syntax are the absence, with exception of exchange rule, of free structural rules and the strong control of the modalities. In the mm rule, $n \in \mathbb{N}$ (note that we have *weakening* for $n = 0$ and *dereliction* for $n = 1$) and it corresponds to the axiom $!A \multimap \underbrace{A \otimes \ldots \otimes A}_{\text{n}}$.

In the original work on SLL [63] , Lafont gives all the technical results with proof nets formalism. Since a proof net [47] can be view as an equivalence class of proof, we can identify the two concept [4]. [5]

A proof net u of a sequent $\Gamma \vdash A$ is a set of *cells* connected through oriented *wires*. There are *atomic* or *compound* cells, and the last type is also called *boxed*. We can emulate the application of each rule with the composition and the plug-in of different cells, and in [63] several rules are defined in order to built proof net. See the original paper for technical details about proof net built rules.
The standard logical normalization procedure has a well defined counterpart in the proof net setting: in fact cut elimination correspond to three different kind of reduction for proof net: *external*, *internal*, and *commutative*.

The techniques used in order to prove the polytime soundness theorems are quite standard.
Some 'ranks' are defined in order to control in a quantitative way the effectively weight of the normalization procedure of the system.

It is possible to prove the following *normalization property*:

Theorem 2.30. *A proof net u of rank n reduces to a unique normal form in at most $W_u(n)$ steps, where W_u is a polynomial.*

The system is also complete for polynomial time computations.
Soft Linear Logic is able to encode standard data structures such as natural numbers, e.g.:

$$\mathbf{N} = \forall\alpha.!(\alpha \multimap \alpha) \multimap \alpha \multimap \alpha$$

booleans,

$$\mathbf{B} = \forall\alpha.(\alpha\&\alpha) \multimap \alpha$$

boolean lists

$$\mathbf{S} = \forall\alpha.!((\alpha \multimap \alpha)\&(\alpha \multimap \alpha)) \multimap \alpha \multimap \alpha$$

and then polynomial expressions.
It is very important to remark that in Soft LL natural numbers are non-duplicable resource.

Soft LL is able to encode in an efficiently way (finite tape) Turing Machines. In fact a Turing Machines with k states and 3 symbols (including blank) is represented by:

$$\mathbf{M} = \forall\alpha.!((\alpha \multimap \alpha)\&(\alpha \multimap \alpha)) \multimap \mathbf{F}_k \otimes \mathbf{F}_3 \otimes (\alpha \multimap \alpha)$$

where $\mathbf{F}_k$ represents the current states and $\mathbf{F}_3$ stands for the current symbols.
The following lemma holds:

[4] In this thesis we do not use the proof-nets formalism, therefore we address the interested reader to the literature on the subject, e.g. [47] [34] [11]
[5] A proof of soundness and completeness of SLL on sequents formulation (and not on proof-nets, as in Lafont's paper) can be found in [43]

Lemma 2.31. *For each Turing Machine with three symbols and k states, there is a generic proof of the sequent* $\mathbf{M} \vdash \mathbf{M}$ *which correspond to the transition function of the Turing Machine.*

In a similar way we can build: (i) a proof net for a sequent $\mathbf{N} \vdash \mathbf{M}$ which transform a natural number into the description of a TM with n cells (initialized with 0) and the head in the first cell; (ii) a proof net for a sequent $\mathbf{M} \vdash \mathbf{M}$ which emulate the writing of symbols on the tape and the moves of the head; (iii) a proof net for a sequent $\mathbf{M} \vdash \mathbf{B}$ which says if the machine is in a accepting or in a non-accepting state.

Finally it is possible to state the following theorem, stating that any polynomial time algorithm is represented by a generic proof net:

Theorem 2.32. *If a predicate of boolean strings is computable by a Turing Machine in polynomial time* $P(n)$ *and in polynomial space* $Q(n)$*, there is a generic proof for the sequent* $\mathbf{S}^{(deg(p)+deg(Q)+1)} \vdash \mathbf{B}$ *which corresponds to this predicate.*

2.3 Modal Logics

This section is devoted to give the minimal background needed to understand our proposal on a modal characterization of quantum state transformations (Chapter 8). The reader is invited to refer to the main literature on modal logic [26] to have a better knowledge.

Modal logic is a framework of a large number of logical system based on the notion of *necessity* and *possibility*. Roughly speaking the main idea of modal logic is to enrich classical propositional logic with two modal operators: $\Box$ and $\Diamond$. The intended meaning of a formula of the kind $\Box A$ is *it is necessary that A is true*, and of $\Diamond A$ is *it is possible that A is true*.

Even if modal logic has been for a lot of time studied only in the so called field of "philosophical logic" now days (starting from the end of seventy-th) is one of the main logical tools in computer science. In fact, it is possible to use modal operators to deal with *computational structures* (in a broad sense). In this respect, it is possible to rephrase the meaning of modal operators w.r.t. computational structures based on the notion of *state*: $\Box A$ is true w.r.t. the current state s iff A is true in each reachable state from s, and dually, $\Diamond A$ is true in the current state s iff A is true in at least a reachable state from the state s.

Definition 2.33 (language). *The alphabet of the modal language consists of: (i) a denumerable set* $\mathcal{P} = \{p_0, p_1, \ldots\}$ *of propositional symbols; (ii) the standard propositional connectives* $\bot$ *and* $\supset$*; and (iii) the unary modal operator* $\Box$*.*
The set MF of modal formulas *(or simply* formulas*) is the last set* X *s.t.: (a)* $\bot \in X$*; (b)* $p \in X$*, for* $p \in \mathcal{P}$*; (c) if* $A, B \in X$ *then* $A \supset B, \Box A \in X$*.*

The other modal and propositional connectives are defined in the standard way:
$\neg A \equiv A \supset \bot$,
$A \wedge B \equiv \neg(A \supset \neg B)$,
$A \leftrightarrow B \equiv (A \supset B) \wedge (B \supset A)$,
$\Diamond A \equiv \neg\Box\neg A$.

Possible worlds semantics

A pair $\mathscr{F} = \langle W, R\rangle$, with W a non empty set of *worlds*, $R \subseteq W \times W$ an *accessibility relation*, is called *frame*.
An *interpretation* (called also *valuation*) on a frame F is a function $V : W \to 2^{\mathcal{P}}$.
A *model* $\mathscr{M}$ is a frame $\mathscr{F}$ plus an interpretation V , namely $\mathscr{M} = \langle \mathscr{F}, V\rangle$.
Let Mod_K be the class of all the models.

It is possible to restrict the class of model by imposing particular conditions on the accessibility relation. In particular we have the following subclasses of Mod_K:

$$
\begin{array}{ll}
\mathsf{Mod}_T & = \{\langle W, R, V\rangle \ : \ \forall u \in W.\ uRu\} \\
\mathsf{Mod}_4 & = \{\langle W, R, V\rangle \ : \ \forall u, v, z \in W.\ uRv \ \&\ vRz \ \Rightarrow\ uRz\} \\
\mathsf{Mod}_B & = \{\langle W, R, V\rangle \ : \ \forall u, v \in W.\ uRv \ \Rightarrow\ vRu\} \\
\mathsf{Mod}_{S4} & = \{\langle W, R, V\rangle \ : \ \forall u, v, z \in W.\ uRv \ \&\ vRz \ \Rightarrow\ uRz, \\
& \quad \forall u \in W.\ uRu\} \\
\mathsf{Mod}_{S5} & = \{\langle W, R, V\rangle \ : \ \forall u, v, z \in W.\ uRv \ \&\ vRz \ \Rightarrow\ uRz, \\
& \quad \forall u, v \in W.\ uRv \ \Rightarrow\ vRu, \forall u \in W.\ uRu\}
\end{array}
$$

Definition 2.34 (Truth). *The truth relations:*
$\models^{\mathscr{M},w}$: *A is true at the world w of $\mathscr{M}$ ($\mathscr{M} = \langle W, R, V\rangle$ and $w \in W$);*
$\models^{\mathscr{M}}$: *A is true in $\mathscr{M}$;*
$\models_{\mu}$: *A is valid w.r.t. the class Mod_μ of models (μ is a name in $\{K, T, 4, B, S4, S5\}$);*
are defined in the following way:

$$
\begin{array}{lll}
\not\models^{\mathscr{M},w} \bot & & \\
\models^{\mathscr{M},w} p & \textit{iff} & p \in V(w) \qquad (p \in \mathcal{P}) \\
\models^{\mathscr{M},w} B \supset C & \textit{iff} & \models^{\mathscr{M},w} B \Rightarrow\ \models^{\mathscr{M},w} C \\
\models^{\mathscr{M},w} \Box B & \textit{iff} & \forall w' \in W.\ wRw' \Rightarrow\ \models^{\mathscr{M},w'} B \\
\models^{\mathscr{M}} A & \textit{iff} & \forall w \in W.\ \models^{\mathscr{M},w} A \\
\models_{\mu} A & \textit{iff} & \forall \mathscr{M} \in \mathsf{Mod}_\mu.\ \models^{\mathscr{M}} A
\end{array}
$$

The main modal systems

Differently form the case of first/second order logic, there is not a unique modal logic, and in fact it is possible to define infinite different modal logics.The following (model theoretically defined) logics are considered basic.

Definition 2.35 (basic modal logics).

$$
\begin{array}{ll}
\mathsf{K} & = \{A : \models_K A\}; \\
\mathsf{KT} & = \{A : \models_T A\}; \\
\mathsf{K4} & = \{A : \models_4 A\} \\
\mathsf{KB} & = \{A : \models_B A\} \\
\mathsf{S4} & = \{A : \models_{S4} A\} \\
\mathsf{S5} & = \{A : \models_{S5} A\}
\end{array}
$$

Roughly speaking, K is the logic of all the models, KT is the logic of models with reflexive accessibility relation, K4 is the logic of models with transitive accessibility relation, KB is the logic of models with symmetric accessibility relation, S4 is the logic of models with reflexive and transitive accessibility relation, S5 is the logic of models where the accessibility relation is an equivalence relation.

2.3.1 Labeled natural deduction systems for modal logics

In order to develop adequate proof theoretical foundations of modal logic some authors [42, 90] have developed the so called *labeled deduction approach*. In particular here we will refer to the approach of Viganò [98].

The main idea of a labeled natural deduction system is to join the well founded model theoretical approach of modal logic with the standard proof theory of classical/intuitionistic logic.

Let us consider the semantics of $\Box$ w.r.t. a model $\mathscr{M} = \langle W, R, V\rangle$,

$$\models^{\mathscr{M},w} \Box A \quad \text{iff} \quad \forall w' \in W.\, wRw' \Rightarrow \models^{\mathscr{M},w'} A$$

It is evident that the intended meaning of $\Box B$ is a that of a *first order quantification on possible worlds* constrained by the accessibility relation R.

The labeled approach is a consequence of the above observation.

The main ingredients of the labeled calculi are:

labels: a set of names that correspond, syntactically, to possible worlds;
relational formulas: expressions of the kind $x\mathsf{R}y$, formalizing, in the syntax, the accessibility between worlds;
labeled formulas: expressions of the kind $x : A$, where x is a label and A is a modal formulas. The intended meaning of $x : A$ is A *holds at world* x.

The introduction/elimination rules for $\Box$ are consequently the following:

$$\dfrac{\begin{array}{c}[x\mathsf{R}y]\\ \vdots\\ y : A\end{array}}{x : \Box A}\,\Box I \qquad\qquad \dfrac{x : \Box A \quad x\mathsf{R}y}{y : A}\,\Box E$$

(see [98] for the definition of semantics and the proof of soundness and completeness of the labeled calculi).

It is intuitive to observe that the rules $\Box I$ and $\Box E$ "mimic" respectively the introduction and elimination rules of a first order quantifier with respect to the variable (label) y, therefore it is not surprising the following constraint on the rule $\Box I$: *the label y is different from x and does not occur in any assumption on which $y : A$ depends other than $x\mathsf{R}y$.*

Definition 2.36 (The labeled language). *Given a denumerable set $Var = \{x_0, x_1, \ldots\}$, ranged by x, y, z, of labels and a binary symbol R, the set of relational formulas (r-formulas) is given by expressions of the form $x\mathsf{R}y$.*

A labelled formula *(*l-formula*) is an expression $x : A$, where x is a label and A is a modal formula. A* formula *is either an r-formula or an l-formula (formulas are ranged by α, β).*

Definition 2.37 (Natural deduction systems, derivations and proofs). *Let us call* minimal modal rules *the rules $\supset I, \supset E, RAA, \Box I, \Box E$ of figure 2.5.*
We define the following natural deduction systems with rules in figure 2.5.

$$\frac{\begin{array}{c}[x : A] \\ \vdots \\ x : B\end{array}}{x : A \supset B} \supset I \qquad \frac{x : A \supset B \quad x : A}{x : B} \supset E \qquad \frac{\begin{array}{c}[x : \neg A] \\ \vdots \\ y : \bot\end{array}}{x : A}\ RAA$$

$$\frac{\begin{array}{c}[x\mathsf{R}y] \\ \vdots \\ y : A\end{array}}{x : \Box A}\ \Box I \qquad \frac{x : \Box A \quad x\mathsf{R}y}{y : A}\ \Box E$$

$$\frac{}{x\mathsf{R}x}\ T \qquad \frac{x\mathsf{R}y \quad y\mathsf{R}z}{x\mathsf{R}z}\ 4 \qquad \frac{y\mathsf{R}x}{x\mathsf{R}y}\ B$$

In RAA, $A \neq \bot$. In $\Box I$, y is fresh: it is different from x and does not occur in any assumption on which $y : A$ depends other than $x\mathsf{R}y$.

Fig. 2.5. The modal rules

system name	***rules***
N_K	*minimal modal rules*
N_T	*minimal modal rules + rule T*
N_4	*minimal modal rules + rule 4*
N_B	*minimal modal rules + rule B*
N_{S4}	*minimal modal rules + rules $T, 4$*
N_{S5}	*minimal modal rules + rules $T, B, 4$*

Let μ be a name in $\{K, T, 4, S4, S5\}$, a derivation *in N_μ of a formula α from a set of formulas Γ is a tree formed using the rules in N_μ, ending with α and depending only on a finite subset of Γ.*
We write $\Gamma \vdash_{N_\mu} \alpha$ to denote that there exists an N_μ-derivation of α from Γ.
A derivation in N_μ of α depending on the empty set is called a proof *of α and we then write $\vdash_{N_\mu} \alpha$ as an abbreviation of $\emptyset \vdash_{N_\mu} \alpha$ and say that α is a theorem of N_μ.*

We need to enrich the notion of model in order to deal with labeled and relational formulas.

Definition 2.38. *A structure $\mathscr{S}$ is a model plus a valuation of labels, namely $\mathscr{S} = \langle W, R, V, \mathscr{I} : Var \to W\rangle$. The notion of truth defined for modal formulas is extended to the case of labeled and relational formulas w.r.t. structures.*

$$\begin{array}{lll}
\models^{\mathscr{M},\mathscr{I}} x\mathsf{R}y & \textit{iff} & \mathscr{I}(x)\ R\ \mathscr{I}(y) \\
\models^{\mathscr{M},\mathscr{I}} x : A & \textit{iff} & \models^{\mathscr{M},\mathscr{I}(x)} A \\
\models^{\mathscr{M},\mathscr{I}} \Gamma & \textit{iff} & \forall \alpha \in \Gamma.\ \models^{\mathscr{M},\mathscr{I}} \alpha
\end{array}$$

Let μ be a name in $\{K, T, 4, B, S4, S5\}$,

$$\begin{array}{lll}
\Gamma \models_{l_\mu} \alpha & \textit{iff} & \forall \mathscr{M} \in \mathsf{Mod}_\mu, \forall \mathscr{I}.\ \Gamma \models^{\mathscr{M},\mathscr{I}} \alpha \\
& \textit{iff} & \forall \mathscr{M} \in \mathsf{Mod}_\mu, \forall \mathscr{I}.\ \models^{\mathscr{M},\mathscr{I}} \Gamma \Longrightarrow\ \models^{\mathscr{M},\mathscr{I}} \alpha
\end{array}$$

Theorem 2.39 (soundness and completeness). *For each (finite or denumerable) set Γ of formulas and for each formula α,*

$$
\begin{array}{lcl}
\Gamma \vdash_{N_K} \alpha & \Leftrightarrow & \Gamma \models_{l_K} \alpha \\
\Gamma \vdash_{N_T} \alpha & \Leftrightarrow & \Gamma \models_{l_T} \alpha \\
\Gamma \vdash_{N_4} \alpha & \Leftrightarrow & \Gamma \models_{l_4} \alpha \\
\Gamma \vdash_{N_B} \alpha & \Leftrightarrow & \Gamma \models_{l_B} \alpha \\
\Gamma \vdash_{N_{S4}} \alpha & \Leftrightarrow & \Gamma \models_{l_{S4}} \alpha \\
\Gamma \vdash_{N_{S5}} \alpha & \Leftrightarrow & \Gamma \models_{l_{S5}} \alpha
\end{array}
$$

A direct consequence of the theorem is the following:

Corollary 2.40 (equivalence of modal systems). *The following equivalences hold:*

$$
\begin{array}{lcl}
\vdash_{N_K} x : A & \Leftrightarrow & A \in \mathsf{K} \\
\vdash_{N_T} x : A & \Leftrightarrow & A \in \mathsf{KT} \\
\vdash_{N_4} x : A & \Leftrightarrow & A \in \mathsf{K4} \\
\vdash_{N_B} x : A & \Leftrightarrow & A \in \mathsf{KB} \\
\vdash_{N_{S4}} x : A & \Leftrightarrow & A \in \mathsf{S4} \\
\vdash_{N_{S5}} x : A & \Leftrightarrow & A \in \mathsf{S5}
\end{array}
$$

3

Basic Concepts of Quantum Computing

In this Chapter we introduce some basic and universally accepted concepts of Quantum Computation, with emphasis on quantum computational models. We will give a description of Quantum Turing Machine [22, 73, 74] and Quantum Circuit Families [73, 74, 104] [58, 72], and we will give the basic definitions of the most useful quantum complexity classes (i.e. the quantum polytime classes); we will also recall the Yao's encoding of QTM with QCF [104].

3.1 Quantum Computing

The results and the notions recalled in this section are mainly based on the following references: C. Isham, *Lectures on quantum theory* [57], P. Kaye, R. Laflamme and M. Mosca *An introduction to quantum computing* [58], M. Nielsen and I, Chuang, *Quantum computation and quantum information* [72], M. Hirvensalo, *Quantum computing* [55], J. von Neumann, *Mathematical foundations of quantum mechanics* [99].

3.1.1 Quantum Bits, Quantum States and the Framework of Quantum Mechanics

Quantum Mechanics was born at the beginning of the 20th Century, when it was clear that the classical theories (such as Newton's and Maxwell's theories), had great problem in order to explain and understand the unexpected results of several physical experiments. Quantum Mechanics is the mathematical framework in which is possible to develop new physical theories as Quantum Physics, taking into account several surprising rules and postulates.
Paul Dirac wrote ⟨⟨*Quantum Mechanics is more suitable in order to understand atomic phenomena, and from several point of view, it appear a more elegant theory with respect to the classical one*⟩⟩ [38]. Nowadays, we can still say that we are able to understand some aspect of the world and the universe only accepting the unusual point of view of Quantum Mechanics.

We will recall the main ideas of Quantum Mechanics in a standard, intuitive way: we introduce the basic notion through four *postulate*, which capture the fundamental connections between the physical world and the mathematical formalism; the postulate give furthermore the basis of Quantum Computing . Then, we refers to standard, fundamental basic concept of Quantum Computing such as quantum bits, quantum states, quantum gates, in order to give some numerical examples.

Remark 3.1. In the following we will use the so called *Bra/Ket-notation*, introduced by Paul Dirac (we tacitly used the same notation in the previous chapter, giving some basic notions).
Given an Hilbert Space $\mathcal{H}$, a *ket* $|\psi\rangle$ indicate a generic elements (column vectors) of $\mathcal{H}$. Kets like $|\psi\rangle$ are typically use to describe quantum state.
The matching $\langle\psi|$ is called *bra*, and denotes the conjugate transpose of $|\psi\rangle$.

Postulates of Quantum Mechanics

Quantum Mechanics framework is able to interpret the structure, the evolution and the interaction of quantum systems, by means of suitable mathematical descriptions.

The first postulate of Quantum Mechanics assigns to quantum systems a mathematical representation in terms of Hilbert Spaces.

Postulate I

The state of a system is described by a unit vector in an Hilbert Space $\mathscr{H}$

The Hilbert Space $\mathscr{H}$ of a quantum system is called the *state space*, and the unit vector represents a *state vector*, which completely describes the system.
Let consider the Hilbert Space $\ell^2(S)$ as defined in 2.1.1, and take $S = \{0,1\}^n$, the set of the finite binary strings of length n.
The Hilbert space $\ell^2(S)$ is the standard space used in quantum computing and it is useful to describe quantum state in a simple, intuitive but rigorous way.

The most simple quantum system is a two-dimensional state space which elements are called *quantum bit* or *qubit* for short.
The more direct way to represent a quantum bit is a unitary vector in the 2-dimensional Hilbert space $\ell^2(\{0,1\})$. We will denote with $|0\rangle$ and $|1\rangle$ the elements of the computational basis of $\ell^2(\{0,1\})$ (see Chapter 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*.
The amplitudes α e β univocally represent the qubit with respect to the computational basis.

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

When defining quantum computational models, we need a generalization of the notion of a qubit, called a *quantum register* or, more commonly, *quantum state*. [73, 85, 87, 97]. A quantum register can be view as a system of n qubits and mathematically it is a normalized vector in the Hilbert space $\ell^2(\{0,1\}^n)$.
The standard orthonormal basis for $\ell^2(\{0,1\}^n)$ is $\mathcal{B} = \{|i\rangle \mid i \text{ is a binary stringof length } n\}$ (see Section 2.1.1), called computational basis. In literature, often it is written that $\ell^2(\{0,1\}^n)$ is the Hilbert Space $\mathbb{C}^{2^n}$. This is not completely correct, in fact we should say that $\ell^2(\{0,1\}^n)$ is isomorphic to $\mathbb{C}^{2^n}$, and in fact it is possible to prove the following proposition:

Proposition 3.2. *The map $\nu : \ell^2(\{0,1\}^n) \to \mathbb{C}^{2^n}$, such that for each element $|i\rangle \in \mathcal{B}$, $\nu(|i\rangle) = (0 \dots 1 \dots 0)^T$ (with 1 only in the $(i-1)$-th position) is an isomorphism of Hilbert Space.*

Note that ν maps the computational basis of $\ell^2(\{0,1\}^n)$ into the standard basis of $\mathbb{C}^{2^n}$.

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

The Hilbert Space $\ell^2(\{0,1\}^n)$ has dimension 2^n and consequently its Hilbert and its Hamel dimension coincide.
Note also that $\ell^2(\{0,1\}^n) \otimes \ell^2(\{0,1\}^m)$ is (up to isomorphism) $\ell^2(\{0,1\}^{n+m})$; the isomorphism is given by the map $|i\rangle \otimes |j\rangle \mapsto |ij\rangle$ (see also Postulate III).

Example 3.3. Let consider a 2-level quantum system, i.e. a system of two qubits. Each 2-qubit quantum register is a normalized vector in $\ell^2(\{0,1\}^2)$ and the computational basis is $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$ (see Postulate III for details about compose quantum system).
For example $1/\sqrt{2}|01\rangle + 1/\sqrt{2}|00\rangle \in \ell^2(\{0,1\}^n)$ is a quantum register of two qubits.

As normalized vectors represent physical systems, the (discrete) evolution of systems can be viewed as suitable transformation on Hilbert Spaces. The following postulate ensures that the evolution is linear and unitary[1]:

Postulate II

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

In quantum computing we refer to an unitary operator U acting on on a n qubits-quantum register as a n-qubit *quantum gate*. Via the isomorphism ν of Proposition 3.2

[1] Note that we refer to a closed system, i.e. to a system that non interacts in any way with other systems and with the rest of the world .
This is obviously an approximation of the reality, but it is a very common approximation in physical theories. We accept this terminology in order to distinguish unitary evolution from quantum measurement, which implies an explicit interaction of the system with the ambient.

we can represent operators on the 2^n-dimensional Hilbert Space $\ell^2(\{0,1\}^n)$ with respect to the standard basis of $\mathbb{C}^{2^n}$ as $2^n \times 2^n$ matrices, and it is possible to prove that to each unitary operator on Hilbert Space it is possible to associate univocally an algebraic representation.

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

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

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

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

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

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

Other important 1-qubit quantum gates are

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

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

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

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

The Hadamard gate is very useful when we want create a superposition starting from a classical state.

1-qubit quantum gate can be used in order to built gate acting on n-qubit quantum state.

A n-qubit quantum register with $n \geq 2$ can be view as a composite system. It is possible to combine two (or more) distinct physical systems into a composite one. In Chapter 2 we have introduced the tensor product $\otimes$. Third Postulate tell us how tensor product of Hilbert Space can describes the state space of a composite system.

Postulate III

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

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

If we have a 2-qubit quantum system, we can apply a 1-qubits quantum gate only to one component of the system, and we implicitly apply the identity operator to the other one. For example suppose we want to apply X to the first qubit. The 2-qubits input $|\psi_1 \otimes \psi_2\rangle$ gets mapped to $X|\psi_1\rangle \otimes I|\psi_2\rangle = (X \otimes I)|\psi_1\rangle \otimes |\psi_2\rangle$.
The linear operator $X \otimes I$ has the matrix representation

$$\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \otimes \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

An important 2-qubits quantum gate is the controlled-not, or CNOT, having the following matrix representation in $\mathbb{C}^{2^2}$

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

In term of its action on the computational basis $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$, the CNOT gate behaves as follow: $|00\rangle \mapsto |00\rangle, \quad |01\rangle \mapsto |01\rangle, \quad |10\rangle \mapsto |11\rangle, \quad |11\rangle \mapsto |10\rangle$ The CNOT gate flips the state of the second qubit (*target* qubit) if the first qubit (*control* qubit) is in the state $|1\rangle$, and nothing otherwise.

Entanglement

Not all quantum states can be viewed as composite systems. In other word, *if $|\psi\rangle$ is a state of a tensor product space $\mathcal{H}_1 \otimes \mathcal{H}_2$ it is not generally true that there exist $|\psi_1\rangle \in \mathcal{H}_1$ and $|\psi_2\rangle \in \mathcal{H}_2$ such that $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$.*
In fact a key property of quantum registers is the following: *it is not always possible to decompose an n-qubit register as the tensorial product of n qubits*.
These non-decomposable registers are called *entangled* and enjoy properties that we cannot find in any object of classical physics. If (the state of) n qubits are entangled, they behave as connected, independently of the real physical distance. The strength of quantum computation is essentially based on the existence of entangled states (see, for example the teleportation protocol [72]) .

Example 3.4. The 2-qubit state $|\psi\rangle = 1/\sqrt{2}|00\rangle + 1/\sqrt{2}|11\rangle$ is entangled.

Measurement

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

In a realistic perspective, a quantum system interacts with other one, and also with a measurement apparatus

The evolution of the state during a measurement is not unitary, so we need a new postulate in order to describe measurement.

Postulate IV

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

The described measurement is called *von Neumann measurement*, and it is a special kind of *projective measurement* (see e.g. [57, 58, 72]).

Projective measurement is very intuitive and it is commonly used to explain measurement Postulate.

In Chapter 7 we will use another type of measurement, the so called *general measurement*.

No-Cloning Theorem

No-Cloning Theorem states that Quantum Mechanics does not allow to make a copy of an unknown quantum states. It was discovered in the early 1980's [103] and it captures one of the fundamental property of quantum systems and of quantum information.

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

But, why cannot a qubit be duplicated? Let $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ be a 1-qubit quantum state. We should try to make a copy using a CNOT gate, as for the classical case[2]. The CNOT gate takes the state $|\psi\rangle$ as the control input, and a state initialized to $|0\rangle$ as the target input. The input state is therefore $\alpha|00\rangle + \beta|01\rangle$. As output, could the CNOT gate give the tensor state $|\psi\rangle \otimes |\psi\rangle$?

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

In general, we can prove the following:

Theorem 3.5 (No-Cloning Theorem). *There not exists an unitary transformation U such that, given a quantum state $|\phi\rangle$ and a quantum state*[3] $|s\rangle$

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

Proof. Suppose there exist the cloning operator U and suppose this copying procedure works for two particular state $|\psi_1\rangle$ and $|\psi_2\rangle$. We have

---

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

[3] The state $|s\rangle$ is assumed to be *pure*, i.e. a quantum state which is not a probabilistic distribution of other quantum states. In quantum mechanics, the notion of pure state is opposed to the notion of *mixed* state, see [58, 59] for detailed discussions.

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

If we take the inner product of the two equations we obtain

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

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

3.2 Quantum Computational Models

3.2.1 Quantum Turing Machine

Let Σ be is a finite alphabet with a blank symbols $\square$ and let Q be a finite set of states, with distinguished initial state q_0 and final state q_f ($q_0 \neq q_f$). As for the classical case, the quantum Turing machine is based on the reading/writing of the tape by an head.

Let us start with the definition of *tape configuration*:

Definition 3.6. *The set of tape configurations is the set of the functions* $\Sigma^{\#} = \{t : \mathbb{Z} \to \Sigma | t(m) \neq \square$ *only for a finite* $m \in \mathbb{Z}\}$.

Given $t \in \Sigma^{\#}$*, a symbol* $\sigma \in \Sigma$ *and an integer* $k \in \mathbb{Z}$*, a new tape configuration* t_k^σ *will be*

$$t_k^\sigma(m) = \begin{cases} \sigma & \text{if } m = k \\ t(m) & \text{if } m \neq k \end{cases}$$

We will call a *frame* the pair (Q, Σ).
To each frame, we can associate the *configuration space* $\mathscr{C}(Q,\Sigma) = Q \times \Sigma^{\#} \times \mathbb{Z}$, where $k \in \Sigma$ is the *head position*.
Each element $C = (q, t, k) \in \mathscr{C}(Q, \Sigma)$ is called *configuration*.

It is possible to construct an Hilbert Space generated by the configuration space: the *quantum state space* is $\ell^2(\mathscr{C}(Q,\Sigma))$ (see Chapter 2, Section 2.1.1), that we denote with $\mathcal{H}(Q, \Sigma)$:

$$\mathcal{H}(Q,\Sigma) = \{\varphi : \mathscr{C}(Q,\Sigma) \to \mathbb{C} \quad | \sum_{C \in \mathscr{C}(Q,\Sigma)} |\varphi(C)|^2 < \infty\}$$

Now, we can define the prequantum, the Quantum Turing Machine and the related properties. In the following we will use the notation $[a, b]_{\mathbb{Z}}$ ($a, b \in \mathbb{Z}$) in order to represent the integers between a and b (with a and b included).

Definition 3.7 (Prequantum Turing Machine).

A prequantum Turing machine *is a triple* (Q, Σ, δ) *where* (Q, Σ) *is a frame and* δ *is the quantum transition function* $\delta : Q \times \Sigma \times Q \times \Sigma \times [-1, 1]_{\mathbb{Z}} \to \mathbb{C}$

We limit the transition amplitudes to the polynomial computable complex numbers $\mathbf{P}\mathbb{C}$. This does not reduce the computational power of the Quantum Turing Machine [22, 73, 74].
The transition function δ induces the so called *time evolution operator* $U^{\delta}_{\mathscr{M}} : \mathcal{H}(Q,\Sigma) \to \mathcal{H}(Q,\Sigma)$, a linear operator defined as

$$U^{\delta}_{\mathscr{M}}|C\rangle = U^{\delta}_{\mathscr{M}}|q,t,k\rangle = \sum_{(p,\sigma,d)\in Q\times\Sigma\times[-1,1]_{\mathbb{Z}})} \delta(q,t(k),p,\sigma,d)\cdot|p,t^{\sigma}_{k},k+d\rangle$$

Definition 3.8 (Quantum Turing Machine). *A* quantum Turing machine *(QTM) is a prequantum Turing machine $\mathscr{M} = (Q,\Sigma,\delta)$ such that the time evolution operator $U^{\delta}_{\mathscr{M}}$ is unitary (i.e. ${U^{\delta}_{\mathscr{M}}}^{\dagger}U^{\delta}_{\mathscr{M}} = I = U^{\delta}_{\mathscr{M}}{U^{\delta}_{\mathscr{M}}}^{\dagger}$) and $\mathsf{range}(\delta) \subseteq \tilde{\mathbb{C}}$.*

Ozawa and Nishimura gave in [73] the following result on the unitarity of time evolution operator:

Theorem 3.9. *Given a prequantum Turing machine, $\mathscr{M} = (Q,\Sigma,\delta)$, the time evolution operator $U^{\delta}_{\mathscr{M}}$ is unitary if and only if the function δ satisfies the following conditions:*

- *for each $(q,\tau) \in Q\times\Sigma$,*

$$\sum_{(p,\sigma,d)\in Q\times\Sigma\times[-1,1]_{\mathbb{Z}})} |\delta(q,\tau,p,\sigma,d)|^2 = 1$$

- *for each $(q,\tau),(q',\tau') \in Q\times\Sigma$ with $(q,\tau)\neq(q',\tau')$*

$$\sum_{(p,\sigma,d)\in Q\times\Sigma\times[-1,1]_{\mathbb{Z}}} \delta(q',\tau',p,\sigma,d)^{*}\delta(q,\tau,p,\sigma,d) = 0$$

- *for each $(q,\tau,\sigma),(q',\tau',\sigma') \in Q\times\Sigma\times\Sigma$*

$$\sum_{(p,d)\in Q\times[-1,1]_{\mathbb{Z}})} \delta(q',\tau',p,\sigma',d-1)^{*}\delta(q,\tau,p,\sigma,d) = 0$$

- *for each $(q,\tau,\sigma),(q',\tau',\sigma') \in Q\times\Sigma\times\Sigma$*

$$\sum_{p\in Q} \delta(q',\tau',p,\sigma',-1)^{*}\delta(q,\tau,p,\sigma,1) = 0$$

Note that other than the unitarity property, we also require that the time evolution operator $U^{\delta}_{\mathscr{M}}$ must be (efficiently), computable; i.e. $U^{\delta}_{\mathscr{M}}$ belongs to the computable operators defined in 2.21.

We say that a QTM $\mathscr{M}$ is in $\mathbf{P}\mathbb{C}$ if the range of the function δ is included in $\mathbf{P}\mathbb{C}$.

Quantum Turing machines need some input/output conventions (see [22]).

We consider *final configuration* any configuration in a QTM $\mathscr{M}$ in the final state q_f.

Definition 3.10 (Polynomial-Time QTM).
We say that a QTM $\mathscr{M}$ halts with running time T *on input x if when $\mathscr{M}$ is run with input x, at time T the superposition contains only final configurations, and at any times $T_i < T$ the superposition contains no final configurations.*
A polynomial time *QTM $\mathscr{M}$ is a QTM which on every input x halts in time T with T polynomial in the length of x.*

Berstein and Vazirani in [22] give also careful definitions on the output of a QTM which halts as a superposition of the tape contents of the configurations in the machines's final position (see the definition of *stationariety*, *normal-form*... [22]).
In the present thesis we do not need to enter in the details of this important discussion which can be found in [22].

How we can verify that the QTM $\mathscr{M}$ effectively halts? Berstein and Vazirani in [22] write:"*This can be accomplished by performing a measurement to check whether the machine is in the final state q_f. Making this partial measurement does not have any other effect on the computation*".

Languages recognized by a QTM and quantum complexity classes

The computational power of quantum computing models has been studied from a complexity theoretic point of view and, as for the classical case, several quantum complexity classes have been defined.
Note that, in this thesis, we refer to the complexity of the so called *decision problems*. A decision problem is a function $Q : \{0,1\}^* \to \{0,1\}$ and, very informally, it performs a question on which the QTM has to give an answer of kind yes/no.

We focus now our attention on the quantum analogue of P, BPP and ZPP.

Definition 3.11. *We say that a QTM $\mathscr{M}$* accepts a language $\mathcal{L}$ with probability p*, if $\mathscr{M}$ accepts with probability at least p every string $x \in \mathcal{L}$, and rejects with probability at least p every string $x \notin \mathcal{L}$.*

Definition 3.12. *The class* EQP *is the set of the languages $\mathcal{L}$ accepted by polynomial QTM $\mathscr{M}$ with probability 1.*

EQP is the error-free (or exact) quantum polynomial-time complexity classes.

Definition 3.13. *The class* BQP *is the set of the languages $\mathcal{L}$ accepted by polynomial QTM $\mathscr{M}$ with probability 2/3.*

Definition 3.14. *The class* ZQP *is the set of the languages $\mathcal{L}$ accepted by polynomial QTM $\mathscr{M}$ such that, for every string x:*

- *if $x \in \mathcal{L}$, then $\mathscr{M}$ accepts x with probability $p > 2/3$ and rejects with probability $p = 0$;*
- *if $x \notin \mathcal{L}$, then $\mathscr{M}$ rejects x with probability $p > 2/3$ and accepts with probability $p = 0$.*

The class ZQP is the zero-error extension of the class BQP. In fact the QTM never gives a wrong answer, but in each case with probability 1/3 gives a "don't-know" answer (clearly, in this case we need to have three answers).

The inclusions $EQP \subseteq ZQP \subseteq BQP$ obviously hold.

The relationship with classical complexity classes is the following: $P \subseteq BPP \subseteq BQP \subseteq PSPACE$.

3.2.2 Quantum Circuits

We now need to introduce the notion of a (finitely generated) quantum circuit family. This is the computational model which will prove equivalent to Q in Chapter 5. We will use it in Chapter 6 too, when we will simulate the Yao's encoding of the Quantum Turing Machine with the calculus SQ.

Elementary classes of operators

We assume here, and for the rest of the thesis, that each considered class of unitary operators are the *elementary operators*.

We say that a class $\{U_i\}_{i\in I}$ of unitary operators is *elementary*, whether for each $j \in I$, the unitary operator U_j is realizable, either physically (i.e. by a laser or by other apparatus) or by means of a computable devices, such as a Turing Machine.

Remarkable classes of elementary operator are the class of computable operators (see Definition 2.21).

Note 3.15. We adopt the definitions given by Nishimura and Ozawa in [73] and [74].

In [74] Nishimura and Ozawa prove the *perfect computational equivalence* between the polynomial-time quantum Turing machine and the *finitely generated uniform quantum circuit families*.

In order to show this result, the authors make some restrictions. The amplitudes of the polynomial time quantum Turing machines has to be in the set $P\mathbb{C}$, and, by definition, the finitely generated quantum circuit family has to be based on a finite subset of the set of quantum gates $\mathcal{G}_u = \{\Lambda_1(N), R(\theta), P(\theta')|\theta, \theta' \in P\mathbb{C} \cap [0, 2\pi]\}$[4].

See [74] and the following section for the details.

It is important to remark that the restriction to a smaller class of quantum gates is also forced by calcolability problems, as remarked by Kitaev, Shen and Vyalyi in [59] The author say (remark 9.2, p. 90): *"the use of an arbitrary complete basis could lead to* pathologies". In fact they prove that the gate

$$X \equiv \begin{pmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{pmatrix}$$

where θ is a noncomputable number, " *enables us to solve the halting problem!*".

Our definition of elementary gates is consistent with the approach of Ozawa and Nishimura.

3.2.3 Quantum Circuit Families (QCF)

An *n-qubit quantum gate* is a unitary operator $\mathbf{U} : \ell^2(\{0,1\}^n) \to \ell^2(\{0,1\}^n)$. Formally, for any $n \in \mathbb{N}$, a $\{0,1\}^n$ quantum gate is an unitary operator on the corresponding Hilbert Space $\ell^2(\{0,1\}^n)$.
Given two unit vectors $|\phi\rangle, |\psi\rangle \in \ell^2(\{0,1\}^n)$, if $U|\phi\rangle = |\psi\rangle$, we call $|\psi\rangle$ the output state for the input state $|\phi\rangle$.

[4] where $\Lambda_1(N)$ is a controlled-not gate, $R(\theta)$ is a rotation gate by angle θ and $P(\theta')$ is a phase shift gate by angle θ'

A $\mathcal{V}$*-qubit gate* (where $\mathcal{V}$ is a set of name) is a unitary operator $\mathbf{G} : \mathcal{H}(\mathcal{V}) \to \mathcal{H}(\mathcal{V})$ (see Section 2.1.1, for the definition of Hilbert Space $\mathcal{H}(\mathcal{V})$).

If $\mathcal{G}$ is a set of qubit gates, a $\mathcal{V}$*-circuit* $\mathbf{K}$ based on $\mathcal{G}$ is a sequence

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

where, for every $1 \leq i \leq m$:

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

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

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

is the unitary operator

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

Once we have fixed an elementary class of operators, it is possible to have an effective encodings of circuits as natural numbers and, as a consequence, an effective enumerations of quantum circuits.

Definition 3.16 (Quantum Circuit Family).

Let $\mathcal{G}$ *be a denumerable set of elementary gates and let* $\{\mathbf{K}_i\}_{i\in\mathbb{N}}$ *be an effective enumeration of quantum circuits. A* family of circuits *generated by* $\mathcal{G}$ *is a triple* (f, g, h) *where:*

- $f : \mathbb{N} \to \mathbb{N}$*;*
- $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ *is such that* $0 \leq g(n, m) \leq n + 1$ *whenever* $1 \leq m \leq f(n)$*;*
- $h : \mathbb{N} \to \mathbb{N}$ *is such that for every* $n \in \mathbb{N}$, $\mathbf{K}_{h(n)}$ *is a* $\{r_1, \ldots, r_{f(n)}\}$*-circuit based on* $\mathcal{G}$*.*

Any family of circuits (f, g, h) *induces a function* $\Phi_{f,g,h}$ *(the* function induced by (f, g, h)*) which, given any finite sequence* $c_1, \ldots, c_n$ *in* $\{0, 1\}^*$*, returns an element of* $\mathcal{H}(\{r_1, \ldots, r_{f(n)}\})$*:*

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

where c_0, c_{n+1} *are assumed to be* 0 *and* 1*, respectively.*

Definition 3.17 (Uniformity for Quantum Circuit Families).

Uniform QCF

Given a quantum circuit family $\mathcal{K} = (f, g, h)$*, we say that* $\mathcal{K}$ *is* uniform *if the functions f, g, h are computable.*

Polynomial-size Uniform QCF

Given a quantum circuit family $\mathcal{K} = (f, g, h)$*, we say that* $\mathcal{K}$ *is* polynomial-size uniform *if the functions f, g, h are polytime.*

Finitely Generated QCF

Given a set $\mathcal{G}$ *of quantum gate, we say that a family of circuits* (f, g, h) *generated by* $\mathcal{G}$ *is* finitely generated *if* $\mathcal{G}$ *is a finite set.*

3.2.4 Encoding Polytime Quantum Turing Machine with Quantum Circuits Families

In [104], Yao has proposed an encoding of quantum Turing machines into quantum circuit families. We will use this result in Chapter 6 and now we recall its principal features.

From now on, we suppose to work with finite alphabets including a special symbol, called *blank* and denoted with $\Box$. Moreover, each alphabet comes equipped with a function $\sigma : \Sigma \to \{0,1\}^{\lceil \lg_2(|\Sigma|) \rceil}$. Σ^ω is the set of infinite strings on the alphabet Σ, i.e., elements of Σ^ω are functions from $\mathbb{Z}$ to Σ. $\Sigma^\#$ is a subset of Σ^ω containing string which are different from $\Box$ in finitely many positions.

Consider a polytime Quantum Turing Machine $\mathcal{M} = (Q, \Sigma, \delta)$ working in time bounded by a polynomial $t : \mathbb{N} \to \mathbb{N}$. The computation of $\mathcal{M}$ on input of length n can be simulated by a quantum circuit $L_{t(n)}$ built as follows:

- for each m, L_m has $\eta + k(\lambda + 2)$ inputs (and outputs), where $\eta = \lceil \log_2 |Q| \rceil$, $k = 2m + 1$ and $\lambda = \lceil \log_2 |\Sigma| \rceil$. The first η qubits correspond to a binary encoding q of a state in Q. The other inputs correspond to a sequence $\sigma_1 s_1, \ldots, \sigma_k s_k$ of binary strings, where each σ_i (with $|\sigma_i| = \lambda$) corresponds to the value of a cell of $\mathcal{M}$, while each s_i (with $|s_i| = 2$) encodes a value from $\{0, 1, 2, 3\}$ controlling the simulation.
- L_m is built up by composing m copies of a circuit K_m, which is depicted in Figure **??** and has $\eta + k(\lambda + 2)$ inputs (and outputs) itself.

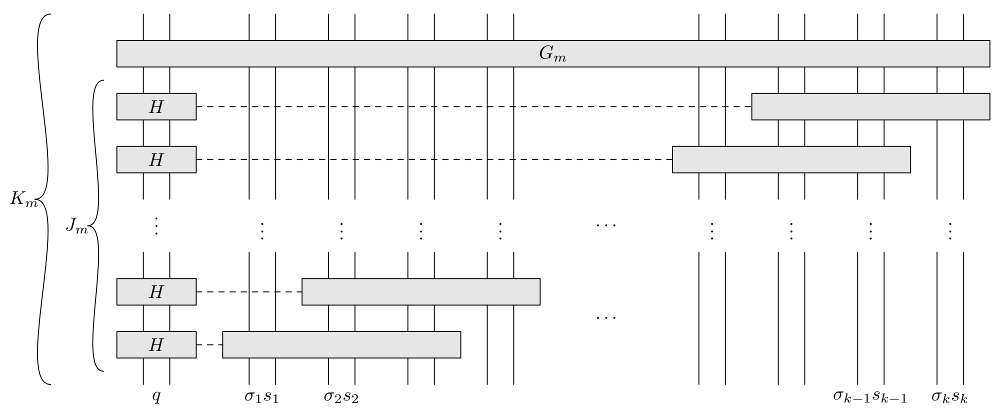


Fig. 3.1. The quantum circuit computing one step of the simulation.

- K_m is built up by composing G_m with J_m. G_m does nothing but switching the inputs corresponding to each s_i from 1 to 2 and vice-versa.
- J_m can be itself decomposed into $k-3$ instances of a single circuit H with $\eta+3(\lambda+2)$ inputs, acting on different qubits as shown in Figure 3.1. Notice that H can be assumed to be computable (in the sense of Definition 2.19), because $\mathcal{M}$ can be assumed to have amplitudes in $\mathbf{P}\mathbb{C}$ [73].

Theorem 3.18 (Yao [104]). *The circuit family $\{L_m\}_{m \in N}$ simulates the Quantum Turing Machine $\mathcal{M}$.*

See the original articles by Yao [104] and by Nishimura and Ozawa [73, 74] for a full explanation of the result.

3.3 Quantum Higher Order Languages

One of the main topic in quantum computing is the investigation of the effective contribution that the new perspective can gives in the develop of efficient algorithm.
Nowadays, some ingenious computational model have been defined, and several researchers developed (the basis of) recursive and complexity theories for the quantum case, as done in the classical one.
In the quantum computing setting the situation is not easy as for the classical case. There are several technical problems related to the complexity of quantum computational model (and the quantum calculus is often very anti-intuitive), but in particular, it seem to be quite complicate move away from the first order models (such as QTM and QCF) toward higher order one. Then, there is the theoretical necessity of developing *calculi for quantum computable functions*, and specifically computational languages for higher order functions.

The first attempt to define a quantum higher–order language has been done in two unpublished papers by Maymin [68, 69]. Selinger in [85] rigorously defined a first-order quantum functional language. Another interesting proposal in the framework of first-order quantum functional languages is the language QML [7] by T. Altenchirk et all. Arrighi and Dowek have recently proposed an interesting extension of λ–calculus with potential applications in the field of quantum computing [9].

We restrict our attention to some distinct *foundational* proposals have already appeared in the literature: first of all the quantum lambda calculus with classical control by Selinger and Valiron [87] (see also an interesting extension proposed by Perdrix [77]). This work was very important for our research, because the notion of *configuration* of our quantum lambda calculi is strongly based on Selinger and Valiron concept of *program state*, and we follow exactly Selinger's paradigm called *quantum data + classical control*.

Subsequently we recall also other approach, such as the quantum lambda calculus by Van Tonder [97] (the first quantum lambda calculus) and other.

Selinger and Valiron's Approach.

The main goal of the work of Selinger and Valiron is to give the basis of a typed quantum functional language. The idea of Selinger and Valiron is to define a language where only data are superposed, and where programs live in a standard classical world. In particular, it is not necessary to have "exotic" objects such as λ–terms in superposition. The approach is well condensed by the slogan: *"classical control + quantum data"*. The proposed calculus, here dubbed λ_{sv}, is based on a call-by-value λ–calculus enriched with constants for unitary transformations and an explicit measurement operator allowing the program to observe the value of one of the quantum data.

Reductions are defined between *program state*: a program state is a triple $[Q, L, M]$, where Q is a normalized vector of an Hilbert Space which represent a quantum state, M is a λ-term and L is the *linking function*, which assign a quantum bit to free variables in M.
Cause the presence of measurement, the authors provide the operational semantics introducing a suitable probabilistic reduction system, in order to define a probabilistic call by value procedure for the evaluation.
λ_{sv} is a typed lambda calculus, and its type system is based on affine intuitionistic Linear Logic: the type system noticeably avoids run time errors and allows to control the linearity

of the system (linearity is extended to higher types), by distinguishing between duplicable and not duplicable resource. Selinger and Valiron develop the following type syntax:

$$A, B ::= \alpha | X | !A | A \multimap B | \top | A \otimes B$$

where α ranges over a set of type constants, X ranges over a countable set of type variables and $\top$ is the linear unit type.
The type system is also equipped with subtyping rules, which provide a more refined control of the resources. The calculus enjoys some good properties such as Subject Reduction and Progress, and a very strong notion of safety. Note that in the quantum setting one of the principal feature of typed calculi, i.e. the *principal type property* fails, for the presence of exponential "!". So the authors develop also a new interesting quantum type inference algorithm, based on the idea that a linear (quantum) type can be viewed as a decoration of an intuitionistic one.

Other Approach.

The calculus introduced by Van Tonder [97], called λ_q, has the same motivation and a number of *immediate similarities* with λ_{sv}.

But there is a couple of glaring differences between λ_q and λ_{sv}. In fact, at a first glance, it seems that λ_q *allows by design arbitrary superpositions of* λ*-terms.* In our opinion the essence of the approach of Van Tonder is in Lemma 5.1 of [97], 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 λ_q has an implicit classical control: it is not possible to superimpose terms differing in a remarkable way, i.e. terms with a different computational evolution. Moreover, in λ_q measurement is not internalized, i.e. there is not any measurement operator as in λ_{sv}.

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

We conclude recalling some important works about quantum higher order languages developed by T. Altenkirch et all. In [8], QML, a quantum languages for quantum computations in a typed setting is proposed, and its operational and denotational semantics are developed using quantum circuits and superoperators. An interesting complete equational theory for QML is successively developed in [7], and the authors proved soundness and completeness results with respect to the defined semantics.

Part II

Main Theme: Quantum Lambda Calculi

4

Q: a quantum lambda calculus with classical control

In this chapter we introduce the Q calculus and we propose a detailed operational study. The calculus is untyped, but the term formation is constrained by means of well forming rules. In order to be correct w.r.t. term reduction we will proved a suitable version of the *subject reduction theorem*.
We will proved a strong confluence result too, and a (quantum) *standardization theorem* for computations.

4.1 A note on the Unitary operators

In the syntax of Q we have constants that explicitly represent unitary operators.
But which unitary operators are available in Q? We here assume that unitary operators can be chosen from $\mathcal{U}$, an arbitrary but denumerable fixed set of unitary operators, called *elementary operators* (see Section 3.2.2). Clearly, the expressive power of Q depends on this choice. If one want, for example, to capture quantum Turing machines in the style of Bernstein and Vazirani [22], one could fix $\mathcal{U}$ to be the set of so-called computable operators. On the other hand, the expressivity results in this chapter relates Q and quantum circuit families; clearly, those that can be captured by Q terms with elementary operators in $\mathcal{U}$ are precisely those (finitely) generated by $\mathcal{U}$.

4.2 The Syntax of Q

This calculus is based on the *"quantum data and classical control"* paradigm, as developed by Selinger and Valiron [87] (see also Chapter 3).

The proposed quantum λ–calculus is based on the notion of *configuration* (a reformulation of the concept of *program state* [87]).

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

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

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

where the only relevant component of the step is the λ–term M. This class of reductions includes all the standard λ–reductions (e.g. β–reduction).

- Configurations, however, can evolve *non-classically*: the term M and the quantum register interact. There are two ways to modify the underlying quantum register:
  1. The *creation of a new quantum bit*, by reducing a term $\mathtt{new}(c)$ (where c is a classical bit). Such a reduction creates a *new quantum variable name* 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 $\mathtt{new}(c)$ 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 $\mathcal{QV} \cup \{r\}$.
  2. The application of a *unitary transformation to the quantum register.* This computation step consists in reducing a term $U\langle r_1, \ldots, r_n\rangle$, where U is the name of 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 $U_{\langle\langle r_1,\ldots,r_n\rangle\rangle}\mathcal{Q}$ 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.

4.2.1 On Linearity

One of the main features of our calculus (and of many other quantum computational models) is linearity, where by linearity we 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. This way we are able to guarantee that the "no cloning and no erasing" property is satisfied. Indeed, whenever $(\lambda x.M)N$ and x occurs (freely) 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, i.e., terms which do not contain any quantum variable. To this purpose, the syntax of terms includes a modal operator ! (called the "bang" operator). The bang operator has been introduced in term calculi for linear logic (see for example Wadler's syntax [100] and Section 2.2.2) and allows to distinguish between those syntactical objects (λ–terms) that can be duplicated or erased and those that cannot. Roughly speaking, a term is duplicable and erasable if and only if it is of the form $!M$ and, moreover, M does not contain quantum variables. This constraint is ensured "statically" by the well–forming rules below.

This is not the only possible way to enforce the no cloning and no erasing properties. Other solutions have been proposed in literature, see e.g. [9] where it is possible to duplicate base vectors, and [7] where duplication is modelled by means of sharing.

4.2.2 The Language of Terms

Let $\mathcal{U}$ be an elementary set (see Chapter 3, Section 3.2.2) of unitary operators. Let us associate to each elementary operator $\mathbf{U} \in \mathcal{U}$ a symbol U. The set of *term expressions*, or *terms* for short, is defined by the grammar in Figure 4.1:

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

Fig. 4.1. Syntax

We assume to work modulo variable renaming, i.e. *terms are equivalence classes modulo α-conversion*. For the linear patterns, we extend the definition by the following scheme: $\lambda\langle x_1, \ldots, x_n\rangle.M \equiv_\alpha \lambda\langle y_1, \ldots y_n\rangle.M\{y_1/x_1, \ldots y_n/x_n\}$, where $y_1, \ldots, y_n$ does not occur at all in M, and for all i, j, $x_i \neq y_j$.

Substitution up to α-equivalence is defined in the usual way.

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

- Variables are either *classical* or *quantum*: the first ones are the usual variables of lambda calculus (and can be bound by abstractions), while each quantum variable refers to a qubit in the underlying quantum register (to be defined shortly).
- There are two sorts of constants as well, namely *boolean constants* (0 and 1) and *unitary operators*: the first ones are useful for generating qubits and play no role in classical computations, while unitary operators are applied to (tuples of) quantum variables when performing quantum computation.
- The term constructor $\mathtt{new}(\cdot)$ creates a new qubit when applied to a boolean constant.
- The syntax allows the 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 [100]. Patterns (and, consequently, lambda abstractions) can only refer to classical variables.

There is not any measurement operator in the language. We will comment on that in Section 4.6.

In the rest of the thesis, a finite subset of quantum variables will be called *quantum variable set* (*qvs*).

4.2.3 Judgements and Well–Formed Terms

Judgements are defined from various notions of environments, that take into account the way the variables are used. Following common notations in type theory and proof theory, a set of variables $\{x_1, \ldots, x_n\}$ is often written simply as $x_1, \ldots, x_n$, with $x_1, \ldots, x_n$ distinct. 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 $\Delta = x_1, \ldots, x_n$, $!\Delta$ denotes the set of patterns $!x_1, \ldots, !x_n$.
- A *quantum environment* is a (possibly empty) set (denoted by Θ, possibly indexed) of quantum variables. Examples of quantum environments are r_1, r_2, r_3 and the empty set $\emptyset$.
- A *linear environment* is (possibly empty) set (denoted by Λ, possibly indexed) in the form Δ, Θ Where Δ is a classical environment and Θ is a quantum environment. The set x_1, x_2, r_1 is an example of a linear environment.
- An *environment* (denoted by Γ, possibly indexed) is a (possibly empty) set in the form $\Lambda, !\Delta$ where each classical variable x occurs at most once (either as $!x$ or as x) in Γ. For example, $x_1, r_1, !x_2$ is an environment, while $x_1, !x_1$ is *not* an environment.
- A *judgement* is an expression $\Gamma \vdash M$, where Γ is an environment and M is a term.

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

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

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

Fig. 4.2. Well–Forming Rules

Since we are working in an untyped setting, term formation is constrained by means of *well forming rules*. The structure of our terms is strongly based on the formulation of Linear Logic proposed by P. Wadler in [100]. We say that a judgement $\Gamma \vdash M$ is *well–formed* (*notation:* $\triangleright \Gamma \vdash M$) if it is derivable by means of the *well–forming rules* in Figure 4.2. The rules app and tens are subject to the constraint that for each $i \neq j$ $\Lambda_i \cap \Lambda_j = \emptyset$ (notice that Λ_i and Λ_j are sets of linear and quantum variables, being linear environments). With $d \triangleright \Gamma \vdash M$ we mean that d is a derivation of the well–formed judgement $\Gamma \vdash M$. If $\Gamma \vdash M$ is well–formed, we say also that the term M *is well–formed with respect to the environment* Γ. We say that a term M is *well–formed* if the judgement $\mathbf{Q}(M) \vdash M$ is well–formed.

Proposition 4.1. *If a term M is well–formed then all the classical variables in it are bound.*

More generally, if $\Lambda, !\Delta \vdash M$ is well-formed, then $\Lambda \subseteq FV(M) \subseteq \Lambda, \Delta$.

4.3 Computations

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

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

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

Definition 4.2 (Configurations). *Two preconfigurations $[\mathcal{Q}, \mathcal{QV}, M]$ and $[\mathcal{Q}', \mathcal{QV}', M']$ are equivalent iff there is a bijection $\theta : \mathcal{QV} \to \mathcal{QV}'$ such that $\mathcal{Q}' = \theta(\mathcal{Q})$ and $M' = \theta(M)$. If a preconfiguration C is equivalent to D, then we will write $C \equiv D$. The relation $\equiv$ is an equivalence relation. A* configuration *is an equivalence class of preconfigurations modulo the relation $\equiv$. Let $\mathcal{C}$ be the set of configurations.*

Remark 4.3. The way configurations have been defined, namely quotienting preconfigurations over $\equiv$, is very reminiscent of usual α-conversion in lambda-terms.

Let $\mathscr{L} = \{\mathsf{Uq}, \mathsf{new}, \mathsf{l}.\beta, \mathsf{q}.\beta, \mathsf{c}.\beta, \mathsf{l.cm}, \mathsf{r.cm}\}$. The set $\mathscr{L}$ will be ranged over by α, β, γ. For each $\alpha \in \mathscr{L}$, we can define a reduction relation $\to_\alpha \subseteq \mathcal{C} \times \mathcal{C}$ by means of the rules in Figure 4.3. Please notice the presence of two commutative reduction rules (namely $\mathsf{l.cm}$ and $\mathsf{r.cm}$). Since Q is untyped, the rôle of commutative reductions is not guaranteeing that normal forms have certain properties, but rather preventing quantum reductions to block classical ones (see Section 4.5).

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

Notice that $\to$ is not a strategy, (the only limitation is that we forbid reductions under the scope of a "!"), nevertheless, confluence holds.

4.3.1 Subject Reduction

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

First of all we stress that, even though Q is type-free, a set of admissible terms is isolated by way of well–forming rules. It is therefore necessary to prove that the class of well–formed terms is closed under reduction.

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

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

β–reductions

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

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

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

Unitary transform of quantum register

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

Creation of a new qubit and quantum variable

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

Commutative reductions

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

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

Context closure

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

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

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

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

Fig. 4.3. Reduction rules.

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

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

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

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 $\triangleright \vdash \texttt{new}(0)$ and, indeed, $p \vdash p$ is well–formed. In other words, we must guarantee that terms appearing during reduction are well-formed, taking into account the set of quantum variables created in the reduction itself.

Proposition 4.4 (Weakening). *For each derivation d, if $d \triangleright \Gamma \vdash M$ and x does not occur in Γ then $\triangleright \Gamma, !x \vdash M$.*

Proof. The proof is by induction on the derivation d. If d is an axiom, trivial. If the last rule r of d has $d_1, \ldots, d_k$ as premise(s), apply the IH to each d_i obtaining d'_i, apply the rule r and conclude.

In order to prove the Subject Reduction Theorem, we need to establish three substitution lemmas (*classical*, *linear* and *quantum*) in order to take into account the three different kind of β reductions.

Lemma 4.5 (Substitution Lemma (linear case)). *For all derivation d_1, d_2, if $d_1 \triangleright \Lambda_1, !\Delta, x \vdash M$ and $d_2 \triangleright \Lambda_2, !\Delta \vdash N$, with $\Lambda_1 \cap \Lambda_2 = \emptyset$, then $\triangleright \Lambda_1, \Lambda_2, !\Delta \vdash M\{N/x\}$.*

Proof. The proof is by induction on the height of d_1 and by cases on the last rule. Let r be the last rule of d_1.

- Let $\Lambda_1, !\Delta \vdash M$ the conclusion of d_1, if $x \notin \Lambda_1$ or $\Lambda_1 \cap \Lambda_2 \neq \emptyset$ the statement is trivially true.
- if d_1 is the axiom $\overline{!\Delta, x \vdash x}$, take d_2 and conclude;
- r is
$$\frac{\Lambda_{11}, !\Delta, x \vdash P_1 \quad \Lambda_{12}, !\Delta \vdash P_2}{\Lambda_{11}, \Lambda_{12}, !\Delta, x \vdash P_1 P_2}\ \textsf{app}.$$
By IH we have: $\triangleright \Lambda_{11}, \Lambda_2, !\Delta \vdash P_1\{N/x\}$, and by means of app:
$\triangleright \Lambda_{11}, \Lambda_{12}, \Lambda_2, !\Delta \vdash P_1\{N/x\}P_2 (\equiv P_1 P_2\{N/x\})$.
- r is
$$\frac{\Lambda_{11}, !\Delta \vdash P_1 \quad \Lambda_{12}, !\Delta, x \vdash P_2}{\Lambda_{11}, \Lambda_{12}, !\Delta, x \vdash P_1 P_2}\ \textsf{app}.$$
As in the previous case.
- r is
$$\frac{\Lambda_{11}, !\Delta \vdash P_1 \quad \cdots \quad \Lambda_{1i}, !\Delta, x \vdash P_i \quad \cdots \quad \Lambda_{1k}, !\Delta \vdash P_k}{\Lambda_{11}, \cdots, \Lambda_{1k}, !\Delta, x \vdash \langle P_1, \cdots, P_k \rangle}\ \textsf{tens}.$$
By IH we have:
$\triangleright \Lambda_{1i}, \Lambda_2, !\Delta \vdash P_i\{N/x\}$, and by means of tens:
$\triangleright \Lambda_{11}, \ldots, \Lambda_{1k}, \Lambda_2, !\Delta \vdash \langle P_1, \ldots, P_i\{N/x\}, \ldots, P_k \rangle (\equiv \langle P_1, \ldots, P_k \rangle \{N/x\})$.
- r is
$$\frac{\Lambda_1, !\Delta, x \vdash P}{\Lambda_1, !\Delta, x \vdash \texttt{new}(P)}\ \textsf{new}.$$
By IH we have: $\triangleright \Lambda_1, \Lambda_2, !\Delta \vdash P\{N/x\}$ and by means of new:
$\triangleright \Lambda_1, \Lambda_2, !\Delta \vdash \texttt{new}(P\{N/x\}) (\equiv \texttt{new}(P)\{N/x\})$.

- r is
$$\frac{\Lambda_1, !\Delta, x_1, \ldots, x_n, x \vdash P}{\Lambda_1, !\Delta, x \vdash \lambda\langle x_1, \ldots, x_n\rangle.P}\ \mathsf{lam1}.$$
By IH we have: $\triangleright \Lambda_1, \Lambda_2, !\Delta, x_1, \ldots, x_n \vdash P\{N/x\}$ and by means of lam1:
$\triangleright \Lambda_1, \Lambda_2, !\Delta \vdash \lambda\langle x_1, \ldots, x_n\rangle.P\{N/x\}(\equiv (\lambda\langle x_1, \ldots, x_n\rangle.P)\{N/x\})$.
- r is
$$\frac{\Lambda_1, !\Delta, x, y \vdash P}{\Lambda_1, !\Delta, x \vdash \lambda y.P}\ \mathsf{lam2}.$$
By IH we have: $\triangleright \Lambda_1, \Lambda_2, !\Delta, y \vdash P\{N/x\}$, and by means of lam2:
$\triangleright \Lambda_1, \Lambda_2, !\Delta \vdash \lambda y.P\{N/x\}(\equiv (\lambda y.P)\{N/x\})$.
- r is
$$\frac{\Lambda_1, !\Delta, x, !y \vdash P}{\Lambda_1, !\Delta, x \vdash \lambda !y.P}\ \mathsf{lam3}.$$
By proposition 4.4 we have $d_2' \triangleright \Lambda_2, !\Delta, !y \vdash N$, by IH we have: $\triangleright \Lambda_1, \Lambda_2, !\Delta, !y \vdash P\{N/x\}$, and by means of lam3:
$\triangleright \Lambda_1, \Lambda_2 !\Delta \vdash \lambda !y.P\{N/x\}(\equiv (\lambda !y.P)\{N/x\})$.

This concludes the proof.

Lemma 4.6 (Substitution (non linear case)). *For all derivation d_1, d_2, if $d_1 \triangleright \Lambda_1, !\Delta, !x \vdash M$ and $d_2 \triangleright !\Delta \vdash !N$, then $\triangleright \Lambda_1, !\Delta \vdash M\{N/x\}$.*

Proof. The proof is by induction on the height of d_1 and by cases on the last rule. Let r be the last rule of d_1.

- Let $\Lambda_1, !\Delta \vdash M$ be the conclusion of d_1, if $!x \not\in !\Delta$ the statement is trivially true.
- Let r be
$$\frac{}{!\Delta, !x \vdash y}\ \mathsf{der}.$$
where $y \neq x$, the statement is trivially true: the result is given by
$$\frac{}{!\Delta \vdash y}\ \mathsf{der}.$$
- r is
$$\frac{}{!\Delta, !x \vdash x}\ \mathsf{der}.$$
We easily obtain a suitable derivation d of $!\Delta \vdash N$ by means of the immediate subderivation of d_2.
- r is
$$\frac{\Lambda_{11}, !\Delta, !x \vdash P_1 \quad \Lambda_{12}, !\Delta, !x \vdash P_2}{\Lambda_{11}, \Lambda_{12}, !\Delta, !x \vdash P_1 P_2}\ \mathsf{app}.$$
By IH we have: $\triangleright \Lambda_{11}, !\Delta \vdash P_1\{N/x\}$ and $\triangleright \Lambda_{12}, !\Delta \vdash P_2\{N/x\}$, and by means of app: $\triangleright \Lambda_{11}, \Lambda_{12}, !\Delta \vdash P_1\{N/x\}P_2\{N/x\}(\equiv (P_1 P_2)\{N/x\})$.
- r is
$$\frac{\Lambda_{11}, !\Delta, !x \vdash P_1 \quad \cdots \quad \Lambda_{1k}, !\Delta, !x \vdash P_k}{\Lambda_{11}, \ldots, \Lambda_{1k}, !\Delta, !x \vdash \langle P_1, \ldots, P_k\rangle}\ \mathsf{tens}.$$
By IH we have:
$\triangleright \Lambda_{1i}, !\Delta \vdash P_i\{N/x\}$ for $i \in [1, k]$, and by means of tens:
$\triangleright \Lambda_{11}, \ldots, \Lambda_{1k}, !\Delta \vdash \langle P_1\{N/x\}, \ldots, P_k\{N/x\}\rangle(\equiv \langle P_1, \ldots, P_k\rangle\{N/x\})$.

- r is
$$\frac{\Lambda_1, !\Delta, !x \vdash P}{\Lambda_1, !\Delta, !x \vdash \mathtt{new}(P)}\ \mathsf{new}.$$
By IH we have: $\rhd \Lambda_1, !\Delta \vdash P\{N/x\}$ and by means of new:
$\rhd \Lambda_1, !\Delta \vdash \mathtt{new}(P\{N/x\})(\equiv \mathtt{new}(P)\{N/x\})$.
- r is
$$\frac{\Lambda_1, !\Delta, x_1, \ldots, x_n, !x, \vdash P}{\Lambda_1, !\Delta, !x \vdash \lambda\langle x_1, \ldots, x_n\rangle.P}\ \mathsf{lam1}.$$
By IH we have: $\rhd \Lambda_1, !\Delta, x_1, \ldots, x_n \vdash P\{N/x\}$ and by means of lam1:
$\rhd \Lambda_1, !\Delta \vdash \lambda\langle x_1, \ldots, x_n\rangle.P\{N/x\}(\equiv (\lambda\langle x_1, \ldots, x_n\rangle.P)\{N/x\})$.
- r is
$$\frac{\Lambda_1, !\Delta, !x, y \vdash P}{\Lambda_1, !\Delta, !x \vdash \lambda y.P}\ \mathsf{lam2}.$$
By IH we have: $\rhd \Lambda_1, !\Delta, y \vdash P\{N/x\}$, and by means of lam2:
$\rhd \Lambda_1, !\Delta \vdash \lambda y.P\{N/x\}(\equiv (\lambda y.P)\{N/x\})$.
- r is
$$\frac{\Lambda_1, !\Delta, !x, !y \vdash P}{\Lambda_1, !\Delta, !x \vdash \lambda !y.P}\ \mathsf{lam3}.$$
By proposition 4.4 we have $d_2' \rhd !\Delta, !y \vdash N$, by IH we have: $\rhd \Lambda_1, !\Delta, !y \vdash P\{N/x\}$, and by means of lam3:
$\rhd \Lambda_1, !\Delta \vdash \lambda !y.P\{N/x\}(\equiv (\lambda !y.P)\{N/x\})$.
- r is prom. In order to apply the promotion rule, Λ_1 must be empty. Therefore r is
$$\frac{!\Delta, !x \vdash P}{!\Delta, !x \vdash !P}\ \mathsf{prom}.$$
By IH we have $\rhd !\Delta \vdash P\{N/x\}$ and by means of prom $\rhd !\Delta \vdash !P\{N/x\}$.

This concludes the proof.

Lemma 4.7 (Substitution (quantum case)). *For each derivation d, for every non empty sequence $x_1, \ldots, x_n$ if $d \rhd \Lambda, !\Delta, x_1, \ldots, x_n \vdash M$ and $r_1, \ldots, r_n \notin \Lambda$, then* $\rhd \Lambda, !\Delta, r_1, \ldots, r_n \vdash M\{r_1/x_1, \ldots, r_n/x_n\}$

Proof. By Lemma 4.5 applied to $d \rhd \Lambda, !\Delta, x_1, \ldots, x_n \vdash M$ and the axiom $!\Delta, r_1 \vdash r_1$ we obtain $\rhd \Lambda, !\Delta, , r_1, x_2, \ldots, x_n \vdash M[r_1/x_1]$, and by means of repeated applications of Lemma 4.5 with respect to axioms $!\Delta, r_2 \vdash r_2, \ldots, !\Delta, r_n \vdash r_n$ we obtain $\rhd \Lambda, !\Delta, , r_1, \ldots, r_n \vdash M[r_1/x_1, \ldots, r_n/x_n]$.

The previously stated substitution lemmas are the main technical tool in order to prove the *subject reduction theorem*:

Theorem 4.8 (Subject Reduction). *If $d \rhd \Gamma \vdash M$ and $[\mathcal{Q}, \mathcal{QV}, M] \to [\mathcal{Q}', \mathcal{QV}', M']$ then $\rhd \Gamma, \mathcal{QV}' - \mathcal{QV} \vdash M'$.*

Proof. The proof is by induction on the height of d and by cases on the last rule r of d.

- r is app and the reduction rule is

$$\frac{[\mathcal{Q}, \mathcal{QV}, P_1] \to_\alpha [\mathcal{Q}', \mathcal{QV}', P_1']}{[\mathcal{Q}, \mathcal{QV}, P_1P_2] \to_\alpha [\mathcal{Q}', \mathcal{QV}', P_1'P_2]}\ \mathsf{l.a}$$

  we have

$$\frac{\Lambda_1, !\Delta \vdash P_1 \quad \Lambda_2, !\Delta \vdash P_2}{\Lambda_1, \Lambda_2, !\Delta \vdash P_1P_2}\ \mathsf{app}.$$

  So by IH we have $\triangleright\ \Lambda_1, \mathcal{QV}' - \mathcal{QV}, !\Delta \vdash P_1'$, and by means of app we obtain $\triangleright\ \Lambda_1, \Lambda_2, \mathcal{QV}' - \mathcal{QV}, !\Delta \vdash P_1'P_2$.
- r is app and the reduction rule is

$$\frac{[\mathcal{Q}, \mathcal{QV}, P_2] \to_\alpha [\mathcal{Q}', \mathcal{QV}', P_2']}{[\mathcal{Q}, \mathcal{QV}, P_1P_2] \to_\alpha [\mathcal{Q}', \mathcal{QV}', P_1P_2']}\ \mathsf{r.a}.$$

  Symmetric to previous case.
- r is app and the reduction rule is

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

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

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

  Let $d_1 \triangleright\ \Lambda_1, !\Delta, x \vdash P$ and $d_2 \triangleright\ \Lambda_2, !\Delta \vdash N$.
  Let us consider the reduction $[\mathcal{Q}, \mathcal{QV}, (\lambda x.P)N] \to_{\mathsf{l.}\beta} [\mathcal{Q}, \mathcal{QV}, P\{N/x\}]$. We note that the reduction does not modify the $\mathcal{QV}$ set, so we have just to apply Lemma 4.5 to d_1 and d_2: $\triangleright \Lambda_1, \Lambda_2, !\Delta \vdash P\{N/x\}$.
- r is app and the reduction rule is q.β or c.β. Similar to previous case. If r is q.β, apply Lemma 4.7, if r is c.β, apply Lemma 4.6.
- r is app and the reduction rule is

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

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

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

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

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

- r is app and the reduction rule is

$$[\mathcal{Q}, \mathcal{QV}, ((\lambda\langle x_1, \dots, x_n\rangle.P)N)L] \to_{\mathsf{r.cm}} [\mathcal{Q}, \mathcal{QV}, (\lambda\langle x_1, \dots, x_n\rangle.PL)N] \ \ \mathsf{r.cm}.$$

As in the previous case,

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

then

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

- r is lam1:

$$\dfrac{\begin{array}{c}d_1\\ \vdots\\ \Gamma, x_1, \dots, x_n \vdash P\end{array}}{\Gamma \vdash \lambda\langle x_1, \dots, x_n\rangle.P}\ \mathsf{lam1}$$

If we have

$$\dfrac{[\mathcal{Q}, \mathcal{QV}, P] \to [\mathcal{Q}', \mathcal{QV}', P']}{[\mathcal{Q}, \mathcal{QV}, \lambda\langle x_1, \dots, x_n\rangle.P] \to [\mathcal{Q}', \mathcal{QV}', \lambda\langle x_1, \dots, x_n\rangle.P']}$$

by IH on d_1

$$\rhd\ \Gamma, \mathcal{QV}' - \mathcal{QV}, x_1, \dots, x_n \vdash P'$$

and we conclude

$$\rhd\ \Gamma, \mathcal{QV}' - \mathcal{QV} \vdash \lambda\langle x_1, \dots, x_n\rangle.P'$$

- r is lam2:

$$\dfrac{\begin{array}{c} d_1 \\ \vdots \\ \Gamma, x \vdash P \end{array}}{\Gamma \vdash \lambda x.P}\ \mathsf{lam1}$$

If we have

$$\frac{[\mathcal{Q}, \mathcal{QV}, P] \to [\mathcal{Q}', \mathcal{QV}', P']}{[\mathcal{Q}, \mathcal{QV}, \lambda x.P] \to [\mathcal{Q}', \mathcal{QV}', \lambda x.P']}$$

by IH on d_1

$$\triangleright\ \Gamma, \mathcal{QV}' - \mathcal{QV}, x \vdash P'$$

and we conclude

$$\triangleright\ \Gamma, \mathcal{QV}' - \mathcal{QV} \vdash \lambda x.P'$$

- r is lam3:

$$\dfrac{\begin{array}{c} d_1 \\ \vdots \\ \Gamma, !x \vdash P \end{array}}{\Gamma \vdash \lambda !x.P}\ \mathsf{lam3}$$

If we have

$$\frac{[\mathcal{Q}, \mathcal{QV}, P] \to [\mathcal{Q}', \mathcal{QV}', P']}{[\mathcal{Q}, \mathcal{QV}, \lambda !x.P] \to [\mathcal{Q}', \mathcal{QV}', \lambda !x.P']}$$

by IH on d_1

$$\triangleright\ \Gamma, \mathcal{QV}' - \mathcal{QV}, !x \vdash P'$$

and we conclude

$$\triangleright\ \Gamma, \mathcal{QV}' - \mathcal{QV} \vdash \lambda !x.P'$$

- r is new

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

We have the following reduction rule:

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

By means of

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

we obtain the result;

- r is new

$$\frac{\Gamma \vdash N}{\Gamma \vdash \mathtt{new}(N)}\ \mathsf{new}$$

and the reduction rule is

$$\frac{[\mathcal{Q}, \mathcal{QV}, N] \to_\alpha [\mathcal{Q}', \mathcal{QV}', N']}{[\mathcal{Q}, \mathcal{QV}, \mathtt{new}(N)] \to_\alpha [\mathcal{Q}', \mathcal{QV}', \mathtt{new}(N')]}$$

In this case the proof is identical to the case of application.

- r is

$$\frac{\Lambda_1, !\Delta \vdash P_1 \quad \cdots \quad \Lambda_k, !\Delta \vdash P_k}{\Lambda_1, \ldots, \Lambda_k, !\Delta \vdash \langle P_1, \ldots, P_k \rangle} \ \mathsf{tens}$$

  and the reduction rule is:

$$\frac{[\mathcal{Q}, \mathcal{QV}, P_i] \rightarrow_\alpha [\mathcal{Q}', \mathcal{QV}', P_i']}{[\mathcal{Q}, \mathcal{QV}, \langle P_1, \ldots, P_i, \ldots, P_k \rangle] \rightarrow_\alpha [\mathcal{Q}', \mathcal{QV}', \langle P_1, \ldots, P_i', \ldots, P_k \rangle]}$$

  For each $j \in \{1, \ldots, k\} - \{i\}$ $d_j \rhd \Lambda_j, !\Delta \vdash P_j$, moreover by IH we have $\rhd \Lambda_i, \mathcal{QV}' - QV, !\Delta, \vdash P_i$ therefore by means of tens we obtain $\rhd \Lambda_1, \ldots \Lambda_k, !\Delta, \mathcal{QV}' - \mathcal{QV} \vdash \langle P_1, \ldots, P_i', \ldots, P_k \rangle$

This concludes the proof.

An immediate corollary (provable by induction) is:

Corollary 4.9. *If* $\rhd \Gamma \vdash M$ *and* $[\mathcal{Q}, \mathcal{QV}, M] \xrightarrow{*} [\mathcal{Q}', \mathcal{QV}', M']$ *then* $\rhd\ \Gamma, \mathcal{QV}' - \mathcal{QV} \vdash M$.

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

Definition 4.10. *A configuration* $[\mathcal{Q}, \mathcal{QV}, M]$ *is said to be well–formed iff there is a context* Γ *such that* $\Gamma \vdash M$ *is well-formed.*

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

Corollary 4.11. *If* M *is well–formed and* $[\mathcal{Q}, \mathcal{QV}, M] \xrightarrow{*} [\mathcal{Q}', \mathcal{QV}', M']$ *then* M' *is well–formed.*

In the following, with *configuration* we mean *well–formed configuration*. Now, let us define normal forms and computations.

Definition 4.12. *A configuration* $C \equiv [\mathcal{Q}, \mathcal{QV}, M]$ *is said to be in normal form iff there is no* D *such that* $C \rightarrow D$*. Let us denote by* NF *the set of configurations in normal form.*

We define a computation as a suitable sequence of configurations:

Definition 4.13. *If* C_0 *is a configuration, a* computation *of length* $\varphi \leq \omega$ *starting with* C_0 *is a sequence of configurations* $\{C_i\}_{i<\varphi}$ *such that for all* $0 < i < \varphi$, $C_{i-1} \rightarrow C_i$ *and either* $\varphi = \omega$ *or* $C_{\varphi-1} \in \mathsf{NF}$.

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

Proposition 4.14. *Let* $\{[\mathcal{Q}_i, \mathcal{QV}_i, M_i]\}_{i<\varphi}$ *be a computation, such that* $\mathbf{Q}(M_0) = \emptyset$*. Then for every* $i < \varphi$ *we have* $\mathcal{QV}_i = \mathbf{Q}(M_i)$*.*

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

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

4.3.2 On the Linearity of the Calculus: Dynamics

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

reductions cannot be performed under the scope of a bang.

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

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

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

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

Notice 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). This is not compatible with the basic principles of quantum computing.

4.3.3 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 4.15. *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 write $M \rightarrow_\alpha N$ meaning that there are $\mathcal{Q}$, $\mathcal{QV}$, $\mathcal{Q}'$ and $\mathcal{QV}'$ such that $[\mathcal{Q}, \mathcal{QV}, M] \rightarrow_\alpha [\mathcal{Q}', \mathcal{QV}', N]$. Similarly for the notation $M \rightarrow_{\mathscr{S}} N$ where $\mathscr{S}$ is a subset of $\mathscr{L}$.

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

Lemma 4.16 (Uniformity). *For every M, M' such that $M \rightarrow_\alpha M'$, exactly one of the following conditions holds:*

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

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

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

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

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

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

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

A property similar to one-step confluence holds in Q. This is a consequence of having adopted the so-called *surface reduction*: it is not possible to reduce inside a subterm in the form $!M$ and, as a consequence, it is not possible to erase a diverging term. This has been already pointed out in the literature [91].

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

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

and

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

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

Proposition 4.18 (One-step Confluence). *Let C, D, E be configurations with $C \to_\alpha D$, $C \to_\beta E$. Then:*

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

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

$$[\mathcal{Q}, \mathcal{QV}, N] \to_\alpha [\mathcal{R}, \mathcal{RV}, P]$$
$$[\mathcal{Q}, \mathcal{QV}, N] \to_\beta [\mathcal{S}, \mathcal{SV}, Q]$$

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

- $D \equiv [\mathcal{R}, \mathcal{RV}, PL]$ and $E \equiv [\mathcal{S}, \mathcal{SV}, NR]$ where $[\mathcal{Q}, \mathcal{QV}, N] \to_\alpha [\mathcal{R}, \mathcal{RV}, P]$ and $[\mathcal{Q}, \mathcal{QV}, L] \to_\beta [\mathcal{S}, \mathcal{SV}, R]$. We need to distinguish four sub-cases:
  - If $\alpha, \beta = \mathsf{new}$, then, by Lemma 7.13, there exist two quantum variables $s, q \notin \mathcal{QV}$ and two constants d, e such that $\mathcal{RV} = \mathcal{QV} \cup \{s\}$, $\mathcal{SV} = \mathcal{QV} \cup \{q\}$, $\mathcal{R} = \mathcal{Q} \otimes |s \mapsto d\rangle$ and $\mathcal{S} = \mathcal{Q} \otimes |q \mapsto e\rangle$. Applying 7.13 again, we obtain

    $$D \to_{\mathsf{new}} [\mathcal{Q} \otimes |s \mapsto d\rangle \otimes |v \mapsto e\rangle, \mathcal{QV} \cup \{s, v\}, PR\{v/q\}] \equiv F$$
    $$E \to_{\mathsf{new}} [\mathcal{Q} \otimes |q \mapsto e\rangle \otimes |u \mapsto d\rangle, \mathcal{QV} \cup \{q, u\}, P\{u/s\}R] \equiv G$$

    As can be easily checked, $F \equiv G$.
  - If $\alpha = \mathsf{new}$ and $\beta \neq \mathsf{new}$, then, by Lemma 7.13 there exists a quantum variable r and a constant c such that $\mathcal{RV} = \mathcal{QV} \cup \{r\}$, $\mathcal{R} = \mathcal{Q} \otimes |r \mapsto c\rangle$, $\mathcal{SV} = \mathcal{QV}$ and $\mathcal{S} = (\mathbf{U}_{L,R} \otimes \mathbf{I}_{\mathcal{QV}-\mathbf{Q}(L)})\mathcal{Q}$. As a consequence, applying Lemma 7.13 again, we obtain

    $$D \to_\beta [(\mathbf{U}_{L,R} \otimes \mathbf{I}_{\mathcal{QV}\cup\{r\}-\mathbf{Q}(L)})(\mathcal{Q} \otimes |r \mapsto c\rangle), \mathcal{QV} \cup \{r\}, PR] \equiv F$$
    $$E \to_{\mathsf{new}} [((\mathbf{U}_{L,R} \otimes \mathbf{I}_{\mathcal{QV}-\mathbf{Q}(L)})\mathcal{Q}) \otimes |r \mapsto c\rangle, \mathcal{QV} \cup \{r\}, PR] \equiv G$$

    As can be easily checked, $F \equiv G$.
  - If $\alpha \neq \mathsf{new}$ and $\beta = \mathsf{new}$, then we can proceed as in the previous case.
  - If $\alpha, \beta \neq \mathsf{new}$, then by Lemma 7.13, there exist $\mathcal{SV} = \mathcal{RV} = \mathcal{QV}$, $\mathcal{R} = (\mathbf{U}_{N,P} \otimes \mathbf{I}_{\mathcal{QV}-\mathbf{Q}(N)})\mathcal{Q}$ and $\mathcal{S} = (\mathbf{U}_{L,R} \otimes \mathbf{I}_{\mathcal{QV}-\mathbf{Q}(L)})\mathcal{Q}$. Applying 7.13 again, we obtain

    $$D \to_\beta [(\mathbf{U}_{L,R} \otimes \mathbf{I}_{\mathcal{QV}-\mathbf{Q}(L)})((\mathbf{U}_{N,P} \otimes \mathbf{I}_{\mathcal{QV}-\mathbf{Q}(N)})\mathcal{Q}), \mathcal{QV}, PR] \equiv F$$
    $$E \to_\alpha [(\mathbf{U}_{N,P} \otimes \mathbf{I}_{\mathcal{QV}-\mathbf{Q}(L)})((\mathbf{U}_{L,R} \otimes \mathbf{I}_{\mathcal{QV}-\mathbf{Q}(L)})\mathcal{Q}), \mathcal{QV}, PR] \equiv G$$

    As can be easily checked, $F \equiv G$.
- $D \equiv [\mathcal{R}, \mathcal{RV}, PL]$ and $E \equiv [\mathcal{S}, \mathcal{SV}, QL]$, where $[\mathcal{Q}, \mathcal{QV}, N] \to [\mathcal{R}, \mathcal{RV}, P]$ and $[\mathcal{Q}, \mathcal{QV}, N] \to [\mathcal{S}, \mathcal{SV}, Q]$. Here we can apply the inductive hypothesis.

- $D \equiv [\mathcal{R}, \mathcal{RV}, NR]$ and $E \equiv [\mathcal{S}, \mathcal{SV}, NS]$, where $[\mathcal{Q}, QV, L] \to [\mathcal{R}, \mathcal{RV}, R]$ and $[\mathcal{Q}, \mathcal{QV}, L] \to [\mathcal{S}, \mathcal{SV}, S]$. Here we can apply the inductive hypothesis as well.
- $N = (\lambda x.T)$, $D \equiv [\mathcal{Q}, \mathcal{QV}, T\{L/x\}]$, $E \equiv [\mathcal{R}, \mathcal{RV}, NR]$, where $[\mathcal{Q}, \mathcal{QV}, L] \to_\beta [\mathcal{R}, \mathcal{RV}, R]$. Clearly $[\mathcal{Q}, \mathcal{QV}, T\{L/x\}] \in \mathcal{C}$ and, by Lemma 7.14, $[\mathcal{Q}, \mathcal{QV}, T\{L/x\}] \to [\mathcal{R}, \mathcal{RV}, T\{R/x\}]$. Moreover, $[\mathcal{R}, \mathcal{RV}, NR] \equiv [\mathcal{R}, \mathcal{RV}, (\lambda x.T)R] \to [\mathcal{R}, \mathcal{RV}, T\{R/x\}]$
- $N = (\lambda x.T)$, $D \equiv [\mathcal{Q}, \mathcal{QV}, T\{L/x\}]$, $E \equiv [\mathcal{R}, \mathcal{RV}, (\lambda x.V)L]$, where $[\mathcal{Q}, \mathcal{QV}, T] \to_\beta [\mathcal{R}, \mathcal{RV}, V]$. Clearly $[\mathcal{Q}, \mathcal{QV}, T\{L/x\}] \in \mathcal{C}$ and, by Lemma 7.14, $[\mathcal{Q}, \mathcal{QV}, T\{L/x\}] \to_\beta [\mathcal{R}, \mathcal{RV}, V\{L/x\}]$. Moreover, $[\mathcal{R}, \mathcal{RV}, (\lambda x.V)L] \to_\beta [\mathcal{R}, \mathcal{RV}, V\{L/x\}]$
- $N = (\lambda !x.T)$, $L = !Z$, $D \equiv [\mathcal{Q}, \mathcal{QV}, T\{Z/x\}]$, $E \equiv [\mathcal{R}, \mathcal{RV}, (\lambda !x.V)L]$, where $[\mathcal{Q}, \mathcal{QV}, T] \to_\beta [\mathcal{R}, \mathcal{RV}, V]$. Clearly $[\mathcal{Q}, \mathcal{QV}, T\{Z/x\}] \in \mathcal{C}$ and, by Lemma 7.14, $[\mathcal{Q}, \mathcal{QV}, T\{Z/x\}] \to_\beta [\mathcal{R}, \mathcal{RV}, V\{Z/x\}]$. Moreover, $[\mathcal{R}, \mathcal{RV}, (\lambda x.V)!Z] \to_\beta [\mathcal{R}, \mathcal{RV}, V\{Z/x\}]$
- $N = (\lambda\langle x_1, \ldots, x_n\rangle.T)$, $L = \langle r_1, \ldots, r_n\rangle$, $D \equiv [\mathcal{Q}, \mathcal{QV}, T\{r_1/x_1, \ldots, r_n/x_n\}]$, $E \equiv [\mathcal{R}, \mathcal{RV}, (\lambda\langle x_1, \ldots, x_n\rangle.V)L]$, where $[\mathcal{Q}, \mathcal{QV}, T] \to_\beta [\mathcal{R}, \mathcal{RV}, V]$. Clearly $[\mathcal{Q}, \mathcal{QV}, T\{r_1/x_1, \ldots, r_n/x_n\}] \in \mathcal{C}$ and, by Lemma 7.14, $[\mathcal{Q}, \mathcal{QV}, T\{r_1/x_1, \ldots, r_n/x_n\}] \to_\beta [\mathcal{R}, \mathcal{RV}, V\{r_1/x_1, \ldots, r_n/x_n\}]$. Moreover, $[\mathcal{R}, \mathcal{RV}, (\lambda\langle x_1, \ldots, x_n\rangle.V)L] \to_\beta [\mathcal{R}, \mathcal{RV}, V\{r_1/x_1, \ldots, r_n/x_n\}]$.
- $N = (\lambda x.T)Z$, $D \equiv [\mathcal{Q}, \mathcal{QV}, (\lambda x.TL)Z]$, $E \equiv [\mathcal{Q}, \mathcal{QV}, (T\{Z/x\})L]$, $\alpha = \mathsf{r.cm}$, $\beta = \mathsf{l.}\beta$. Clearly, $[\mathcal{Q}, \mathcal{QV}, (\lambda x.TL)Z] \to_{\mathsf{l.}\beta} [\mathcal{Q}, \mathcal{QV}, (T\{Z/x\})L]$.
- $N = (\lambda\pi.T)Z$, $D \equiv [\mathcal{Q}, \mathcal{QV}, (\lambda\pi.TL)Z]$, $E \equiv [\mathcal{R}, \mathcal{RV}, ((\lambda\pi.V)Z)L]$, $\alpha = \mathsf{r.cm}$, where $[\mathcal{Q}, \mathcal{QV}, T] \to_\beta [\mathcal{R}, \mathcal{RV}, V]$. Clearly, $[\mathcal{Q}, \mathcal{QV}, (\lambda x.TL)Z] \to_{\mathsf{r.cm}} [\mathcal{R}, \mathcal{RV}, (\lambda x.VL)Z]$ and $[\mathcal{R}, \mathcal{RV}, ((\lambda\pi.V)Z)L] \to_\beta [\mathcal{R}, \mathcal{RV}, (\lambda\pi.VL)Z]$.
- $N = (\lambda\pi.T)Z$, $D \equiv [\mathcal{Q}, \mathcal{QV}, (\lambda x.TL)Z]$, $E \equiv [\mathcal{R}, \mathcal{RV}, ((\lambda\pi.T)X)L]$, $\alpha = \mathsf{r.cm}$, where $[\mathcal{Q}, \mathcal{QV}, Z] \to_\beta [\mathcal{R}, \mathcal{RV}, X]$. Clearly, $[\mathcal{Q}, \mathcal{QV}, (\lambda x.TL)Z] \to_{\mathsf{r.cm}} [\mathcal{R}, \mathcal{RV}, (\lambda x.TL)X]$ and $[\mathcal{R}, \mathcal{RV}, ((\lambda\pi.T)X)L] \to_\beta [\mathcal{R}, \mathcal{RV}, (\lambda\pi.TL)X]$.
- $N = (\lambda\pi.T)Z$, $D \equiv [\mathcal{Q}, \mathcal{QV}, (\lambda x.TL)Z]$, $E \equiv [\mathcal{R}, \mathcal{RV}, ((\lambda\pi.T)Z)R]$, $\alpha = \mathsf{r.cm}$, where $[\mathcal{Q}, \mathcal{QV}, L] \to_\beta [\mathcal{R}, \mathcal{RV}, R]$. Clearly, $[\mathcal{Q}, \mathcal{QV}, (\lambda x.TL)Z] \to_{\mathsf{r.cm}} [\mathcal{R}, \mathcal{RV}, (\lambda x.TR)Z]$ and $[\mathcal{R}, \mathcal{RV}, ((\lambda\pi.T)Z)R] \to_\beta [\mathcal{R}, \mathcal{RV}, (\lambda\pi.TR)Z]$.
- $N = (\lambda\pi.T)$, $L = (\lambda x.Z)Y$, $D \equiv [\mathcal{Q}, \mathcal{QV}, (\lambda x.NZ)Y]$, $E \equiv [\mathcal{Q}, \mathcal{QV}, N(Z\{Y/x\})]$, $\alpha = \mathsf{l.cm}$, $\beta = \mathsf{l.}\beta$. Clearly, $[\mathcal{Q}, \mathcal{QV}, (\lambda x.NZ)Y] \to \mathsf{l.}\beta[\mathcal{Q}, \mathcal{QV}, N(Z\{Y/x\})]$.

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

The following definition is useful when talking about reduction lengths, and takes into account both commuting and non-commuting reductions:

Definition 4.19. *Let $C_1, \ldots, C_n$ be a sequence of configurations such that $C_1 \to \ldots \to C_n$. The sequence is called an m-*sequence of length n from C_1 to C_n *iff m is a natural number and there is $A \subseteq \{2, \ldots, n\}$ with $|A| = m$ and $C_{i-1} \to_{\mathscr{N}} C_i$ iff $i \in A$. If there is a m-sequence of length n from C to D, we will write $C \xrightarrow{m,n} D$ or simply $C \xrightarrow{m} D$.*

This way we can generalize Proposition 4.18 to another one talking about reduction sequences of arbitrary length:

Theorem 4.20 (Confluence). *Let C, D_1, D_2 be configurations with $C \xrightarrow{m_1} D_1$ and $C \xrightarrow{m_2} D_2$. Then, there is a configuration E with $D_1 \xrightarrow{n_1} E$ and $D_2 \xrightarrow{n_2} E$ with $n_1 \leq m_2$, $n_2 \leq m_1$ and $n_1 + m_1 = n_2 + m_2$.*

Proof. We prove the following, stronger statement: suppose there are C, D_1, D_2, a m_1-sequence of length l_1 from C to D_1 and an m_2-sequence of length l_2 from C to D_2. Then, there are a configuration E, a n_1-sequence of length k_1 from D_1 to E and n_2-sequence of length k_2 from D_2 to E with $n_1 \leq m_2$, $n_2 \leq m_1$, $k_1 \leq l_2$, $k_2 \leq l_1$ and

$n_1 + m_1 = n_2 + m_2$. We go by induction on $l_1 + l_2$. If $l_1 + l_2 = 0$, then $C \equiv D_1 \equiv D_2$, $E \equiv D_1 \equiv D_2$ and all the involved natural numbers are 0. If $l_1 = 0$, then $D_1 \equiv C$ and $E \equiv D_2$. Similarly when $l_2 = 0$. So, we can assume $l_1, l_2 > 0$. There are G_1, G_2, two integers $h_1, h_2 \leq 1$ with $C \to_\alpha G_2$ and $C \to_\beta G_2$, an $(m_1 - h_1)$-sequence of length $l_1 - 1$ from G_1 to D_1 and an $(m_2 - h_2)$-sequence of length $l_2 - 1$ from G_2 to D_2. We can distinguish four cases, depending on the outcome of Proposition 4.18:

- $\alpha \in \mathscr{K}$, $\beta \in \mathscr{K}$ with $G_1 = G_2$, or $\alpha \in \mathscr{N}$, $\beta \in \mathscr{N}$ with $G_1 = G_2$. By applying one time the the induction hypothesis we have the following diagram:

$$\begin{tikzcd}
 & & C \arrow[dl, "h_1{,}1"'] \arrow[dr, "h_1{,}1"] & & \\
 & G_1 \arrow[dl, "m_1-h_1{,}l_1-1"'] & = & G_2 \arrow[dr, "m_2-h_1{,}l_2-1"] & \\
D_1 \arrow[drr, "n_1{,}s_1"'] & & & & D_2 \arrow[dll, "n_2{,}s_2"] \\
 & & E & &
\end{tikzcd}$$

with the equations:

$$\begin{gathered}
n_1 \leq m_2 - h_1 \\
n_2 \leq m_1 - h_1 \\
s_1 \leq l_2 - 1 \\
s_2 \leq l_1 - 1 \\
n_1 + (m_1 - h_1) = n_2 + (m_2 - h_1)
\end{gathered}$$

from which $n_1 \leq m_2$, $n_2 \leq m_1$, and $n_1 + m_1 = n_2 + m_2$.

- $\alpha \in \mathscr{K}$, $\beta \in \mathscr{K}$ with $G_1 \neq G_2$, or $\alpha \in \mathscr{N}$, $\beta \in \mathscr{N}$ with $G_1 \neq G_2$ and there is H with $G_1 \to_\beta H$ and $G_2 \to_\alpha H$. By applying several times the induction hypothesis, we end up with the following diagram

$$\begin{tikzcd}
 & & C \arrow[dl, "h_1{,}1"'] \arrow[dr, "h_2{,}1"] & & \\
 & G_1 \arrow[dl, "m_1-h_1{,}l_1-1"'] \arrow[dr, "h_2{,}1"] & & G_2 \arrow[dl, "h_1{,}1"'] \arrow[dr, "m_2-h_2{,}l_2-1"] & \\
D_1 \arrow[dr, "q_1{,}t_1"'] & & H \arrow[dl, "u_1{,}v_1"'] \arrow[dr, "u_2{,}v_2"] & & D_2 \arrow[dl, "q_2{,}t_2"] \\
 & J \arrow[dr, "w_1{,}z_1"'] & & K \arrow[dl, "w_2{,}z_2"] & \\
 & & E & &
\end{tikzcd}$$

together with the equations:

$$
\begin{array}{lll}
q_1 \le h_2 & q_2 \le h_1 & w_1 \le u_2 \\
t_1 \le 1 & t_2 \le 1 & z_1 \le v_2 \\
u_1 \le m_1 - h_1 & u_2 \le m_2 - h_2 & w_2 \le u_1 \\
v_1 \le l_1 - 1 & v_2 \le l_2 - 1 & z_2 \le v_1
\end{array}
$$

and

$$
m_1 - h_1 + q_1 = u_1 + h_2 \qquad h_1 + u_2 = m_2 - h_2 + q_2 \qquad w_1 + u_1 = w_2 + u_2
$$

from which

$$
\begin{aligned}
q_1 + w_1 &\le h_2 + u_2 \le h_2 + m_2 - h_2 = m_2 \\
t_1 + z_1 &\le 1 + v_2 \le 1 + l_2 - 1 = l_2 \\
q_2 + w_2 &\le h_1 + u_1 \le h_1 + m_1 - h_1 = m_1 \\
t_2 + z_2 &\le 1 + v_1 \le 1 + l_1 - 1 = l_1 \\
q_1 + w_1 + m_1 &= h_1 + h_2 + u_1 + w_1 = h_1 + h_2 + u_2 + w_2 = m_2 + w_2 + q_2
\end{aligned}
$$

So we can just put $n_1 = q_1 + w_1$, $n_2 = q_2 + w_2$, $k_1 = t_1 + z_1$, $k_2 = t_2 + z_2$.

- $\alpha \in \mathscr{K}$, $\beta \in \mathscr{N}$ and there is H with $G_1 \equiv H$ and $G_2 \to_\beta H$. By applying several times the induction hypothesis, we end up with the following diagram:

$$
\begin{tikzcd}
 & & C \arrow[dl, "{h_1,1}"'] \arrow[dr, "{0,1}"] & & \\
 & G_1 \arrow[dl, "{m_1-h_1,l_1-1}"'] & & G_2 \arrow[ll, "{h_1,1}"] \arrow[dr, "{m_2,l_2-1}"] & \\
D_1 \arrow[drr, "{n_1,k_1}"'] & & & & D_2 \arrow[dll, "{n_2,k_2}"] \\
 & & E & &
\end{tikzcd}
$$

together with the equations:

$$
\begin{array}{l}
n_1 \le m_2 \\
k_1 \le l_2 - 1 \\
n_2 \le m_1 \\
k_2 \le l_1
\end{array}
$$

and

$$
m_1 + n_1 = m_2 + n_2
$$

from which the desired equations can be easily obtained.

- The last case is similar to the previous one.

This concludes the proof.

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

Lemma 4.21. *The relation $\to_{\mathscr{K}}$ is strongly normalizing. In other words, there cannot be any infinite sequence $C_1 \to_{\mathscr{K}} C_2 \to_{\mathscr{K}} C_3 \to_{\mathscr{K}} \ldots$.*

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

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

Theorem 4.22 (Uniqueness 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 4.18 obtaining a configuration F such that both $D \xrightarrow{*} F$ and $E \xrightarrow{*} F$. This however, is a contradiction.

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

Theorem 4.23. *A configuration C is strongly normalizing iff C is weakly normalizing.*

Proof. Strong normalization implies weak normalization. Suppose, by way of contradiction, that C is weakly normalizing but not strongly normalizing. This implies there is a configuration D in normal form and an m-sequence from C to D. Since C is not strongly normalizing, there is an infinite sequence $C \equiv C_1, C_2, C_3, \ldots$ with $C_1 \to C_2 \to C_3 \to \ldots$ From this infinite sequence, we can extract an $m+1$-sequence, due to Lemma 7.17. Applying Proposition 4.20, we get a configuration F and a 1-sequence from D to F. However, such a 1-sequence cannot exist, because D is normal.

4.4 Examples

We give now some simple examples showing how to compute with Q when the length of the input is fixed. In Section 5.2 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) [18].

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

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

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

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

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

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 $\mathbf{cnot}$ and $\mathbf{H}$ 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 is the first quantum algorithm that has been defined. It has interesting applications: for example it allows to compute a global property of a function by combining results from two components of a superposition. We refer here to the *Deutsch's Algorithm* as presented in [72], pages 32 and 33 (a detailed explanation of the algorithm is outside the scope of this paper).

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

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

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

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

In order to perform such a task, we first evaluate the normal form of:

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

$$
\begin{array}{ll}
 & [1, \mathsf{D}\langle \mathtt{new}(0), \mathtt{new}(1)\rangle] \\
\to^2_{\mathsf{new}} & [|p\mapsto 0\rangle \otimes |q\mapsto 1\rangle, (\lambda\langle x,y\rangle(\lambda\langle w,z\rangle.\langle Hw, z\rangle)(W_f\langle Hx, Hy\rangle))\langle p,q\rangle] \\
\to_{\mathsf{q}.\beta} & [|p\mapsto 0\rangle \otimes |q\mapsto 1\rangle, (\lambda\langle w,z\rangle.\langle Hw, z\rangle)(W_f\langle Hp, Hq\rangle)] \\
\to_{\mathsf{Uq}} & [\frac{|p\mapsto 0\rangle + |p\mapsto 1\rangle}{\sqrt{2}} \otimes |q\mapsto 1\rangle, (\lambda\langle w,z\rangle.\langle Hw, z\rangle)(W_f\langle p, Hq\rangle)] \\
\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 Hw, 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 Hw, 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 Hw, 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 Hp, 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; i.e. $f(0)\oplus f(1) = 0$.
  In this case the normal form may be rewritten as (by means of simple algebraic manipulations):
  $$[(-1)^{f(0)}|p\mapsto 0\rangle \otimes \frac{|q\mapsto 0\rangle - |q\mapsto 1\rangle}{\sqrt{2}}, \langle p, q\rangle]$$
- f is not a constant function; i.e. $f(0)\oplus f(1) = 1$.
  In this case the normal form may be rewritten as:
  $$[(-1)^{f(0)}|p\mapsto 1\rangle \otimes \frac{|q\mapsto 0\rangle - |q\mapsto 1\rangle}{\sqrt{2}}, \langle p, q\rangle]$$

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

Exchange

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

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

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

$$
\begin{array}{ll}
 & [1, \mathsf{L}\ \langle \mathtt{new}(1), \mathtt{new}(0)\rangle] \\
\xrightarrow{2} & [|p\mapsto 1\rangle \otimes |q\mapsto 0\rangle, (\lambda\langle x,y\rangle.(\lambda\langle a,b\rangle.cnot\langle b,a\rangle)((\lambda\langle w,z\rangle.cnot\langle z,w\rangle)(cnot\langle x,y\rangle))\langle p,q\rangle] \quad (4.1) \\
\to_{\mathsf{q}.\beta} & [|p\mapsto 1\rangle \otimes |q\mapsto 0\rangle, (\lambda\langle a,b\rangle.cnot\langle b,a\rangle)((\lambda\langle w,z\rangle.cnot\langle z,w\rangle)cnot\langle p,q\rangle]) \\
\to_{\mathsf{Uq}} & [|p\mapsto 1\rangle \otimes |q\mapsto 1\rangle, (\lambda\langle a,b\rangle.cnot\langle b,a\rangle)((\lambda\langle w,z\rangle.cnot\langle z,w\rangle)\langle p,q\rangle)] \\
\to_{\mathsf{q}.\beta} & [|p\mapsto 1\rangle \otimes |q\mapsto 1\rangle, (\lambda\langle a,b\rangle.cnot\langle b,a\rangle)cnot\langle q,p\rangle] \\
\to_{\mathsf{Uq}} & [|p\mapsto 0\rangle \otimes |q\mapsto 1\rangle, (\lambda\langle a,b\rangle.cnot\langle b,a\rangle)\langle q,p\rangle] \\
\to_{\mathsf{q}.\beta} & [|p\mapsto 0\rangle \otimes |q\mapsto 1\rangle, (cnot\langle p,q\rangle)] \\
\to_{\mathsf{Uq}} & [|p\mapsto 0\rangle \otimes |q\mapsto 1\rangle, \langle p,q\rangle] \quad (4.2)
\end{array}
$$

Please notice that the values attributed to p and q in the underlying quantum register are exchanged between configurations (4.1) and (4.2).

4.5 Standardizing Computations

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

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

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

We distinguish three particular subsets of $\mathscr{L}$, namely $\mathscr{Q} = \{\mathsf{Uq}, \mathsf{q}.\beta\}$, $n\mathscr{C} = \mathscr{Q} \cup \{\mathsf{new}\}$, and $\mathscr{C} = \mathscr{L} - n\mathscr{C}$. Let $C \to_\alpha D$ and let M be the relevant redex in C; if $\alpha \in \mathscr{Q}$ the redex M is called *quantum*, if $\alpha \in \mathscr{C}$ the redex M is called *classical*.

Definition 4.24. *A configuration C is called 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 called essentially quantum if $\alpha \in \mathscr{Q}$ whenever $C \to_\alpha D$. Let* EQT *be the set of essentially quantum configurations.*

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

Definition 4.25. *A* CNQ *computation starting with a configuration C is a computation $\{C_i\}_{i<\varphi}$ such that $C_0 \equiv C$, $\varphi \leq \omega$ and:*

1. *for every $1 < i+1 < \varphi$, if $C_{i-1} \to_{n\mathscr{C}} C_i$ 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 reduction, while EQT is closed under quantum reduction:

Lemma 4.26. *If $C \in \mathsf{NCL}$ and $C \to_{\mathsf{new}} D$ then $D \in \mathsf{NCL}$.*

Proof. Let C be $[\mathcal{Q}, \mathcal{QV}, M]$ and D be $[\mathcal{Q}', \mathcal{QV}', M']$. The expression $\mathsf{C}[\cdot]$ will denote a term context [1]. Let $\mathtt{new}(c)$ be the reduced redex in M. Clearly, there is a context $\mathsf{C}[\cdot]$ such that $M \equiv \mathsf{C}[\mathtt{new}(c)]$ and $M' \equiv \mathsf{C}[r]$. The proof proceeds by induction on the structure of $\mathsf{C}[\cdot]$:

- If $\mathsf{C}[\cdot] \equiv [\cdot]$, then $M' \equiv r$ does not contain any redex.
- Clearly, $\mathsf{C}[\cdot] \not\equiv !\mathsf{D}[\cdot]$, because reduction cannot take place under the scope of the operator !.
- If $\mathsf{C}[\cdot] \equiv \mathtt{new}(\mathsf{D}[\cdot])$ then by IH $\mathsf{D}[r]$ cannot contain any classical redex and, hence $\mathsf{C}[r]$ cannot contain any classical redex.

[1] a term context is a term with one hole

- If $\mathsf{C}[\cdot] \equiv \mathsf{D}[\cdot]N$, then by IH $\mathsf{D}[r]$ cannot contain any classical redex. Moreover, N itself cannot contain any classical redex. So, if $M' \equiv \mathsf{D}[r]N$ contain any classical redex, the redex should be M' itself. But it is immediate to check that in any of these cases, $M \equiv \mathsf{D}[\mathtt{new}(c)]N$ is a redex too (which goes against the hypothesis). For example, if $\mathsf{D}[\cdot] \equiv \lambda x.\mathsf{E}[\cdot]$, then $M \equiv (\lambda x.\mathsf{E}[\mathtt{new}(c)])N$ contains a classical redex (M itself).
- If $\mathsf{C}[\cdot] \equiv N\mathsf{D}[\cdot]$, we can proceed exactly as in the previous case.
- If
$$\mathsf{C}[\cdot] \equiv \langle N_1, \ldots, N_{k-1}, \mathsf{D}[\cdot], N_{k+1}, \ldots, N_n\rangle$$
then, by inductive hypothesis, $\mathsf{D}[r]$ cannot contain any classical redex. Moreover, $N_1, \ldots, N_{k-1}, N_{k+1}, \ldots, N_n$ cannot contain any classical redex themselves. But this implies M' cannot contain any classical redex.
- The same argument can be applied to the cases $\mathsf{C}[\cdot] \equiv \lambda\pi.\mathsf{D}[\cdot]$ and $\mathsf{C}[\cdot] \equiv \lambda!x.\mathsf{D}[\cdot]$.

This concludes the proof.

Lemma 4.27. *If $C \in \mathsf{EQT}$ and $C \to_{\mathscr{Q}} D$ then $D \in \mathsf{EQT}$.*

Proof. Let C be $[\mathcal{Q}, \mathcal{QV}, M]$ and D be $[\mathcal{Q}', \mathcal{QV}', M']$. Let N be the reduced redex in M. Clearly, there is a context $\mathsf{C}[\cdot]$ such that $M \equiv \mathsf{C}[N]$ and $M' \equiv \mathsf{C}[N']$. Observe that N can be either in the form
$$(\lambda\langle x_1, \ldots, x_n\rangle.L)\langle r_1, \ldots, r_n\rangle$$
or in the form
$$U\langle r_1, \ldots, r_n\rangle.$$
In the first case, we say that N is a *variable passing redex*, while in the second case, we say that N is a *unitary transformation redex*. The proof proceeds by induction on the structure of $\mathsf{C}[\cdot]$:

- If $\mathsf{C}[\cdot] \equiv [\cdot]$, then:
  - If N is a unitary transformation redex, then $M' \equiv \langle r_1, \ldots, r_n\rangle$ does not contain any redex.
  - If N is a variable passing redex, then $M' \equiv L\{r_1/x_1, \ldots, r_n/x_n\}$. But the following lemma can be easily proved by induction on P: for any term P, if P only contains quantum redexes, then $P\{r_1/x_1, \ldots, r_n/x_n\}$ only contains quantum redexes, too.
- Clearly, $\mathsf{C}[\cdot] \not\equiv !\mathsf{D}[\cdot]$, because reduction cannot take place under the scope of the operator !.
- If $\mathsf{C}[\cdot] \equiv \mathtt{new}(\mathsf{D}[\cdot])$ then by IH $\mathsf{D}[N']$ only contains quantum redexes. Now, observe that $\mathsf{D}[N']$ cannot be a boolean constant. Indeed, if N is a unitary transformation redex, then N' contains, at least, the term $\langle r_1, \ldots, r_n\rangle$. If N is a variable passing redex, on the other hand, N' contains the quantum variables $r_1, \ldots, r_n$ because the variables $x_1, \ldots, x_n$ appears exactly once in L. Hence $\mathsf{C}[N']$ only contains quantum redexes.
- If $\mathsf{C}[\cdot] \equiv \mathsf{D}[\cdot]P$, then by IH $\mathsf{D}[N']$ only contains quantum redexes. Moreover, P itself only contains quantum redexes. So if $M' \equiv \mathsf{D}[N']P$ contain any non-quantum redex, the redex must be M' itself. Let us check that in any of these cases, $M \equiv \mathsf{D}[N]P$ is a non-quantum redex too:
  - If M' is a $\mathsf{l}.\beta$ redex, then $\mathsf{D}[\cdot] \equiv \lambda x.\mathsf{E}[\cdot]$, and $M \equiv (\lambda x.\mathsf{E}[N])P$ contains a classical redex (M itself).

   - If M' is a $\mathsf{c}.\beta$ redex, then $\mathsf{D}[\cdot] \equiv \lambda!x.\mathsf{E}[\cdot]$, $P \equiv !Q$ and and $M \equiv (\lambda!x.\mathsf{E}[N])!Q$ contains a classical redex.
   - If M' is a $\mathsf{l.cm}$ redex, then $P \equiv (\lambda\pi.Q)R$ and $M \equiv \mathsf{D}[N]P \equiv \mathsf{D}[N](\lambda\pi.Q)R$ is a $\mathsf{l.cm}$ redex, too.
   - If M' is a $\mathsf{r.cm}$ redex, then $\mathsf{D}[N'] \equiv (\lambda\pi.Q)R$. We have to distinguish four subcases:
     - If $\mathsf{D}[\cdot] \equiv [\cdot]$, then N must be a variable passing redex and, as a consequence, $M \equiv NP$ is a $\mathsf{r.cm}$ redex.
     - If $\mathsf{D}[\cdot] \equiv [\cdot]R$, then N must be a variable passing redex and, as a consequence, NR is a $\mathsf{r.cm}$ redex.
     - If $\mathsf{D}[\cdot] \equiv (\lambda\pi.\mathsf{E}[\cdot])R$, then M is $((\lambda\pi.\mathsf{E}[N])R)P$, which is a $\mathsf{r.cm}$ redex.
     - If $\mathsf{D}[\cdot] \equiv (\lambda\pi.Q)\mathsf{E}[\cdot]$, then M is $((\lambda\pi.Q)\mathsf{E}[N])P$, which is a $\mathsf{r.cm}$ redex.
- If $\mathsf{C}[\cdot] \equiv N\mathsf{D}[\cdot]$, we can proceed as in the previous case.
- If
$$\mathsf{C}[\cdot] \equiv \langle N_1, \dots, N_{k-1}, \mathsf{D}[\cdot], N_{k+1}, \dots, N_n\rangle$$
then, by inductive hypothesis, $\mathsf{D}[N']$ cannot contain any classical redex. Moreover, $N_1, \dots, N_{k-1}, N_{k+1}, \dots, N_n$ cannot contain any classical redex themselves. But this implies M' cannot contain any classical redex.
- The same argument can be applied to the cases $\mathsf{C}[\cdot] \equiv \lambda\pi.\mathsf{D}[\cdot]$ and $\mathsf{C}[\cdot] \equiv \lambda!x.\mathsf{D}[\cdot]$.

This concludes the proof.

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

Theorem 4.28 (Standardization). *For every computation $\{C_i\}_{i<\varphi}$ such that $\varphi \in \mathbb{N}$ there is a* CNQ *computation $\{D_i\}_{i<\xi}$ such that $C_0 \equiv D_0$ and $C_{\varphi-1} \equiv D_{\xi-1}$.*

Proof. We build a CNQ computation in three steps:
1. Let us start to reduce $D_0 \equiv C_0$ *by using* $\mathscr{C}$ reductions as much as possible. By Theorem 4.23 we must obtain a finite reduction sequence $D_0 \to_{\mathscr{C}} \dots \to_{\mathscr{C}} D_k$ s.t. $0 \le k$ and no $\mathscr{C}$ reductions are applicable to D_k
2. Reduce D_k *by using* new reductions as much as possible. By Theorem 4.23 we must obtain a finite reduction sequence $D_k \to_{\mathsf{new}} \dots \to_{\mathsf{new}} D_j$ s.t. $k \le j$ and no new reductions are applicable to D_j. Note that by Lemma 4.26 such reduction steps cannot generate classical redexes and in particular no classical redex can appear in D_j.
3. Reduce D_j *by using* $\mathscr{Q}$ reductions as much as possible. By Theorem 4.23 we must obtain a finite reduction sequence $D_j \to_{\mathscr{Q}} \dots \to_{\mathscr{Q}} D_m$ such that $j \le m$ and no $\mathscr{Q}$ reductions are applicable to D_m. Note that by Lemma 4.27 such reduction steps cannot 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 $D_0 \to_{\mathscr{C}} \dots \to_{\mathscr{C}} D_k \to_{\mathsf{new}} \dots \to_{\mathsf{new}} D_j \to_{\mathscr{Q}} \dots \to_{\mathscr{Q}} D_m$ is a CNQ computation. By Theorem 4.22 we observe that $C_{\varphi-1} \equiv D_m$, which implies the thesis.

The *intuition* behind a CNQ computation is the following: the first phase of the computation is responsible for the construction of a λ–term (abstractly) representing a quantum circuit and does not touch the underlying quantum register. The second phase builds the quantum register without introducing any superposition. The third phase corresponds to proper quantum computation (unitary operators are applied to the quantum register, possibly introducing superposition). This intuition will become a technical recipe in order to prove a side of the equivalence between Q and quantum circuit families formalism (see Section 5.2.1).

We conclude by examining the case of *non terminating computations*. From a quantum point of view, non terminating computations are not particularly interesting, because there is no final measurable quantum state, and consequently the transformations of the quantum register are inaccessible (see also Section 4.6 for a discussion on the absence of measurements in Q).

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

Lemma 4.29. *The relation $\rightarrow_{n\mathscr{C}}$ is strongly normalizing (i.e. there cannot be any infinite sequence $C_1 \rightarrow_{n\mathscr{C}} C_2 \rightarrow_{n\mathscr{C}} C_3 \rightarrow_{n\mathscr{C}} \ldots$).*

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

As a consequence of the Lemma we have that

Proposition 4.30. *Any infinite* CNQ *computation only includes classical reduction steps.*

Example 4.31. An example of non-normalizing term in Q (and then of an infinite reduction), is the following: given $M = \lambda!x.(x!x)$, clearly $M(!M) \rightarrow M(!M)$

Finally we can state the theorem:

Theorem 4.32 (Standardization for infinite computations). *For every non terminating computation $\{C_i\}_{i<\omega}$ there is a* CNQ *computation $\{D_i\}_{i<\omega}$ such that $C_0 \equiv D_0$.*

Proof. We build the CNQ computation in the following way: start to reduce $D_0 \equiv C_0$ *by using* $\mathscr{C}$ reductions as much as possible. This procedure cannot end, otherwise we would contradict Lemma 4.29 and Theorem 4.23.

4.6 On the Measurement Operator

In Q it is not possible to classically observe the content of the quantum register. More specifically, the language of terms does not include any measurement operator which, applied to a quantum variable, has the effect of observing the value of the related qubit. This in contrast with Selinger and Valiron's λ_{sv} (where such a measurement operator is indeed part of the language of terms) and with other calculi for quantum computation like the so-called measurement calculus [33], where the possibility of observing is even more central.

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

- The reduction relation would be probabilistic, since observing a qubit can have different outcomes. As a consequence, confluence would not be true anymore.
- The standardization theorem would not hold in the form it has here. In particular, the application of unitary transformations to the underlying quantum register could not necessarily be postponed until the end of a computation.

The main reason why we have restricted our attention to a calculus without any explicit measurement operator is that the (extensional) expressive power of the obtained calculus (i.e. the extensional class of quantum computable functions) would presumably be the same.

As a consequence, *we assume to perform a unique implicit measurement at the end of computation*.

Please notice that the possibility of measuring qubits internally (e.g. by a construct like $\mathtt{meas}(\cdot)$ could allow to solve certain problems more efficiently, by exploiting the inherent nondeterminism involved in measurements. Indeed, it is not known whether measurement-based quantum computation can be *efficiently* (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 on the quantum register, without any consequence on the previously stated results. Simply add to the calculus the following rule:

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

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 $|a_i|^2$.

Because the importance of measurement, in Chapter 7 we will propose an extension of Q with a measurement operator.

5

Q: expressive power

In this section we study the expressive power of Q, showing that it is equivalent to finitely generated quantum circuit families, and consequently (via the result of Ozawa and Nishimura [73]) we have the equivalence with quantum Turing machines as defined by Bernstein and Vazirani [22]. The fact that the considered class of circuit families only contains finitely generated ones is not an accident: our idea is in fact to represent an entire family by one single lambda term (which is, by definition, a finite object), and consequently we must restrict to families which are generated by a finite set of gates.

Before going into the details, an informal description of how our encoding works is in order. Data will be encoded using some variations on Scott's numerals [101]. These can be used both for classical and quantum data. In the latter case, a more strictly linear discipline (quantum bits cannot be erased, in general) is enforced through a slightly different encoding. Our analysis will concentrate on terms in Q satisfying a simple constraint: when applied to a list of classical bits, they produce a list of quantum variables. These are the *quantum relevant* terms. What is crucial from a computational point of view is the way a quantum relevant term can possibly modify the underlying quantum register.

5.1 Q and the Lambda Calculus

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

It's true that Girard's encodings of intuitionistic logic into linear logic can be somehow generalized to translations from pure, untyped, lambda calculus to untyped linear lambda terms, like the ones of Q (see, for example, [100]). Beta reduction in the lambda calculus, however, does *not* correspond to surface reduction in Q. Take, for example, the classical lambda term $M \equiv x((\lambda y.yy)(\lambda y.yy))$: it is not normalizable, but its (call-by-name) translation $\overline{M} \equiv x!((\lambda !y.y!y)!(\lambda !y.y!y))$ is clearly a normal form in Q. There are some connections between weak head reduction in the lambda calculus and surface reduction in Q: if M rewrites to N by weak head reduction, then $\overline{M}$ rewrites to $\overline{N}$ in Q. The converse is not true: $M = \lambda x.((\lambda y.yy)(\lambda y.yy))$ is a weak head normal form, but $\overline{M}$ is not normalizable in Q. Similar considerations hold for weak call-by-value reduction when the translation function $\overline{(\cdot)}$ is the one induced by the embedding $A \to B \equiv !(A \multimap B)$.

On the other hand, lambda calculus is Turing complete for any decent encoding of natural numbers into it. This holds for Scott numerals, for example. But does this correspondence scale down to more restricted notions of reduction, like weak head reduction?

Even if the above question has a positive answer, that would not settle the issue. If Q is proved to have the classical expressive power of Turing machines, this simply implies that it is possible to compute the *code* D_n of the n-th circuit C_n of any quantum circuit family from input n. But D_n is nothing but a natural number, the "Gödel's number" of C_n. Since we want to evaluate C_n inside Q, we need to prove that the correspondence $D_n \mapsto C_n$ is itself representable in Q and since the way quantum circuits are represented and evaluated in Q has nothing to do with 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 5.2.

5.2 Encoding Quantum Circuit Families

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

On the Classical Strength of the Q.

Natural numbers are encoded as Q terms as follows:

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

This way, we can compute the successor and the 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:

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

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

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

They behave as follows:

$$\forall m \leq n \quad \mathsf{case}^{\mathsf{nat}}_n \ \lceil m \rceil !M_0 \ldots !M_n !N \rightarrow^*_{\mathscr{C}} M_m$$
$$\forall m > n \quad \mathsf{case}^{\mathsf{nat}}_n \ \lceil m \rceil !M_0 \ldots !M_n !N \rightarrow^*_{\mathscr{C}} N$$

$$\begin{aligned}
\mathsf{case}^{\mathsf{nat}}_0 \ \lceil 0 \rceil !M_0 !N &\rightarrow^*_{\mathscr{C}} (\lambda !x.\lambda !y.y)!(\lambda !w.N)!M_0 \\
&\rightarrow^*_{\mathscr{C}} M_0 \\
\mathsf{case}^{\mathsf{nat}}_0 \ \lceil m+1 \rceil !M_0 !N &\rightarrow^*_{\mathscr{C}} (\lambda !x.\lambda !y.x\lceil m \rceil)!(\lambda !w.N)!M_0 \\
&\rightarrow_{\mathscr{C}} (\lambda !w.N)\lceil m \rceil \rightarrow_{\mathscr{C}} N \\
\mathsf{case}^{\mathsf{nat}}_{n+1} \ \lceil 0 \rceil !M_0 \ldots !M_{n+1} !N &\rightarrow^*_{\mathscr{C}} (\lambda !x.\lambda !y.y)!(\lambda w.\mathsf{case}^{\mathsf{nat}}_n w !M_1 \ldots !M_{n+1} !N)!M_0 \\
&\rightarrow^*_{\mathscr{C}} M_0 \\
\mathsf{case}^{\mathsf{nat}}_{n+1} \ \lceil m+1 \rceil !M_0 \ldots !M_{n+1} !N &\rightarrow^*_{\mathscr{C}} (\lambda !x.\lambda !y.x\lceil m \rceil)!(\lambda w.\mathsf{case}^{\mathsf{nat}}_n w !M_1 \ldots !M_{n+1} !N)!M_0 \\
&\rightarrow^*_{\mathscr{C}} (\lambda w.\mathsf{case}^{\mathsf{nat}}_n \ w !M_1 \ldots !M_{n+1} !N)\lceil m \rceil \\
&\rightarrow_{\mathscr{C}} \mathsf{case}^{\mathsf{nat}}_n \ \lceil m \rceil !M_1 \ldots !M_{n+1} !N
\end{aligned}$$

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

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

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 as follows on lists:

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

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

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

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:

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

Indeed:

$$\begin{aligned}\mathsf{extract}_0 M[N_1,\ldots N_m] &\rightarrow^*_{\mathscr{C}} M[N_1,\ldots,N_m]\\ \mathsf{extract}_{n+1} M[] &\rightarrow^*_{\mathscr{C}} M[]\\ \forall m\leq n\quad \mathsf{extract}_{n+1} M[N,N_1\ldots N_m] &\rightarrow^*_{\mathscr{C}} \mathsf{extract}_n M[N_1,\ldots,N_m]N\\ &\rightarrow^*_{\mathscr{C}} M[]N_m\ldots N_1 N\\ \forall m> n\quad \mathsf{extract}_{n+1} M[N,N_1\ldots N_m] &\rightarrow^*_{\mathscr{C}} \mathsf{extract}_n M[N_1,\ldots,N_m]N\\ &\rightarrow^*_{\mathscr{C}} M[N_{n+1}\ldots N_m]N_n\ldots N_1 N\end{aligned}$$

The encodings of natural numbers and lists are similar and are both in the style of the so-called Scott's numerals [101]. However, there is an essential difference between the two:

- Natural numbers are encoded *non-linearly*: any natural number is duplicable by construction, since it has the shape $!M$ for some M.
- Lists are encoded *linearly*: the occurrences of M and $[M_1,\ldots,M_n]$ which are part of $[M,M_1,\ldots,M_n]$ do not lie in the scope of any bang operator.

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

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

For each term M,

$$\begin{aligned}\mathsf{rec}!M \quad\equiv\quad (\mathsf{rec_{aux}}!\mathsf{rec_{aux}})!M &\rightarrow_{\mathscr{M}} (\lambda!y.y!((\mathsf{rec_{aux}}!\mathsf{rec_{aux}})!y))!M\\ \rightarrow_{\mathscr{M}} M!((\mathsf{rec_{aux}}!\mathsf{rec_{aux}})!M)) &\equiv M!(\mathsf{rec}!M)\end{aligned}$$

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

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

Indeed:

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

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

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

Definition 5.1. *A (partial) function $f : \mathbb{N}^n \to \mathbb{N}$ is* representable *iff there is a term M_f such that:*
- *Whenever $M_f\lceil m_1\rceil \dots \lceil m_n\rceil$ has a normal form N (with respect to $\to^*_{\mathscr{C}}$), then $N \equiv \lceil m\rceil$ for some natural number m.*
- *$M_f\lceil m_1\rceil \dots \lceil m_n\rceil \to^*_{\mathscr{C}} \lceil m\rceil$ iff $f(m_1, \dots, m_n)$ is defined and equal to m.*

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

Proposition 5.2. *The class of representable functions coincides with the class of partial recursive functions (on natural numbers).*

Proof. Kleene's partial recursive functions can be embedded into Q:
- Constant functions, the successor and projections can be easily encoded.
- The composition $f : \mathbb{N}^m \to \mathbb{N}$ of $h : \mathbb{N}^n \to \mathbb{N}$ and $g_1, \dots, g_n : \mathbb{N}^m \to \mathbb{N}$ can be represented as follows:

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

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

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

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

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

where

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

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

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

Definition 5.3. *Let $\mathscr{S}$ be any subset of $\mathscr{L}$. The expression $C \Downarrow_{\mathscr{S}} D$ means that $C \to^*_{\mathscr{S}} D$ and D is in normal form with respect to the relation $\to_{\mathscr{S}}$. $C \Downarrow D$ stands for $C \Downarrow_{\mathscr{L}} D$.*

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

Definition 5.4. *A term M is called quantum relevant (shortly,* qrel*) if it is well–formed and for each list $![!c_1, ..., !c_n]$ there are a quantum register $\mathcal{Q}$ and a natural number m such that $[1, \emptyset, M![!c_1, ..., !c_n]] \Downarrow [\mathcal{Q}, \{r_1, \ldots, r_m\}, [r_1, \ldots, r_m]]$.*

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

Circuits.

In this section, we will show that Q is at least as computationally strong as finitely generated uniform quantum circuit families (see Definition 3.17). Our task will not be too difficult, since we already know from Proposition 5.2 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 uniform family of circuits.

The n-th elementary permutation of m elements (where $1 \leq n < m$) is the function which maps n to $n+1$, $n+1$ to n and any other elements in the interval $1, \ldots, m$ to itself.

Lemma 5.5. *Any (finite) permutation can be effectively decomposed into a product of elementary permutations.*

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

Lemma 5.6. *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

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

Indeed:

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

This completes the proof.

Lemma 5.7. *There is a term M_{length} such that, for every list $[!N_1,\ldots,!N_n]$, $M_{length}[!N_1,\ldots,!N_n] \to^*_{\mathscr{C}} \lceil n\rceil$.*

Proof. M_{length} is the term

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

Indeed:

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

This completes the proof.

Lemma 5.8. *There is a term M_{choose} such that for every list $[!N_1,\ldots,!N_m]$:*

$$
\begin{aligned}
M_{choose}\lceil 0\rceil[!N_1,\ldots,!N_m] &\to^*_{\mathscr{C}} !\lceil 0\rceil\\
\forall 1\le n\le m \quad M_{choose}\lceil n\rceil[!N_1,\ldots,!N_m] &\to^*_{\mathscr{C}} !N_n\\
M_{choose}\lceil m+1\rceil[!N_1,\ldots,!N_m] &\to^*_{\mathscr{C}} !\lceil 1\rceil
\end{aligned}
$$

Proof. M_{choose} is the term
$$\lambda x.\lambda y.(\mathsf{iter}^{\mathsf{list}} y!L!P)x$$
where
$$\begin{aligned}
L &\equiv \lambda z.\lambda !w.\lambda !q.q!(\lambda s.\lambda r.(s!L_{\geq 2}!L_{=1})r)!(L_{=0})z \\
L_{=0} &\equiv \lambda t.t\lceil 0\rceil \\
L_{=1} &\equiv \lambda t.(\lambda !u.!w)(t\lceil 0\rceil) \\
L_{\geq 2} &\equiv \lambda u.\lambda t.t(\mathsf{succ}\ u) \\
P &\equiv \lambda !z.z!(\lambda !w.!\lceil 1\rceil)!\lceil 0\rceil
\end{aligned}$$
Indeed:
$$\begin{aligned}
M_{choose}\lceil 0\rceil [] &\rightarrow^*_{\mathscr{C}} (\mathsf{iter}^{\mathsf{list}}[]!L!P)\lceil 0\rceil \\
&\rightarrow^*_{\mathscr{C}} P\lceil 0\rceil \\
&\rightarrow^*_{\mathscr{C}} (\lambda !x.\lambda !y.y)!(\lambda !w.!\lceil 1\rceil)!\lceil 0\rceil \rightarrow^*_{\mathscr{C}} !\lceil 0\rceil \\
M_{choose}\lceil 1\rceil [] &\rightarrow^*_{\mathscr{C}} P\lceil 1\rceil \\
&\rightarrow^*_{\mathscr{C}} (\lambda !x.\lambda !y.x\lceil 0\rceil)!(\lambda !w.!\lceil 1\rceil)!\lceil 0\rceil \rightarrow^*_{\mathscr{C}} !\lceil 1\rceil \\
M_{choose}\lceil n\rceil [!N, !N_1, \ldots, !N_m] &\rightarrow^*_{\mathscr{C}} (\mathsf{iter}^{\mathsf{list}}[!N, !N_1, \ldots, !N_m]!L!P)\lceil n\rceil \\
&\rightarrow^*_{\mathscr{C}} L(\mathsf{iter}^{\mathsf{list}}[!N_1, \ldots, !N_m]!L!P)!N\lceil n\rceil \\
&\rightarrow^*_{\mathscr{C}} \lceil n\rceil !(\lambda !s.\lambda r.(s!L_{\geq 2}!(L_{=1}\{N/w\}))r)!(L_{=0}) \\
&\qquad (\mathsf{iter}^{\mathsf{list}}[!N_1, \ldots, !N_m]!L!P) \\
&\equiv \lceil n\rceil !Q!(L_{=0})S
\end{aligned}$$
where
$$\begin{aligned}
Q &\equiv \lambda !s.\lambda r.(s!L_{\geq 2}!(L_{=1}\{N/w\}))r \\
S &\equiv \mathsf{iter}^{\mathsf{list}}[!N_1, \ldots, !N_m]!L!P
\end{aligned}$$
Now:
$$\begin{aligned}
\lceil 0\rceil !Q!(L_{=0})S &\rightarrow^*_{\mathscr{C}} L_{=0}S \rightarrow^*_{\mathscr{C}} S\lceil 0\rceil \rightarrow^*_{\mathscr{C}} !\lceil 0\rceil \\
\lceil 1\rceil !Q!(L_{=0})S &\rightarrow^*_{\mathscr{C}} (\lambda !s.\lambda r.(s!L_{\geq 2}!L_{=1}\{N/w\})r)\lceil 0\rceil S \\
&\rightarrow^*_{\mathscr{C}} (\lambda !x.\lambda !y.y)!L_{\geq 2}!(L_{=1}\{N/w\})S \\
&\rightarrow^*_{\mathscr{C}} L_{=1}\{N/w\}S \\
&\rightarrow^*_{\mathscr{C}} (\lambda !u.!N)(S!\lceil 0\rceil) \\
&\rightarrow^*_{\mathscr{C}} (\lambda !u.!N)!\lceil 0\rceil \rightarrow^*_{\mathscr{C}} !N \\
\lceil n+2\rceil !Q!(L_{=0})S &\rightarrow^*_{\mathscr{C}} (\lambda !s.\lambda r.(s!L_{\geq 2}!L_{=1}\{N/w\})r)\lceil n+1\rceil S \\
&\rightarrow^*_{\mathscr{C}} (\lambda !x.\lambda !y.x\lceil n\rceil)!L_{\geq 2}!(L_{=1}\{N/w\})S \\
&\rightarrow^*_{\mathscr{C}} L_{\geq 2}\lceil n\rceil S \\
&\rightarrow^*_{\mathscr{C}} S\lceil n+1\rceil
\end{aligned}$$
This completes the proof.

Now, we prove that any finitely generated family of circuits can be represented in Q.

From Chapter 3, Section 3.2.2, recall that $\{\mathbf{K}_i\}_{i\in\mathbb{N}}$ is an effective enumeration of quantum circuits and we assume it is based on an elementary set of unitary operators.

It is possible to prove the following theorem:

Theorem 5.9. *For every finitely generated family of circuits (f, g, h) there is a quantum relevant term $M_{f,g,h}$ such that for each $c_1, \ldots, c_n$, the following two conditions are equivalent*

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

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

$$\mathbf{U}^i_1, r_1^{i,1}, \ldots, r_1^{i,p(i,1)}, \ldots, \mathbf{U}^i_{k(i)}, r_{k(i)}^{i,1}, \ldots, r_{k(i)}^{i,p(i,k(i))}$$

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

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

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

- A function $r : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ which, given (i, j) returns the number of elementary permutations of $\{1, \ldots, f(h(i))\}$ in which π^i_j can be decomposed (via Lemma 5.5).
- A function $q : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that $q(i, j, x) = y$ iff the x-th elementary permutation of $\{1, \ldots, f(h(i))\}$ in which π^i_j can be decomposed is the y-th elementary permutation.
- A function $s : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ which, given (i, j) returns the number of elementary permutations of $\{1, \ldots, f(h(i))\}$ in which ρ^i_j can be decomposed (via Lemma 5.5).
- A function $t : \mathbb{N} \times \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that $t(i, j, x) = y$ iff the x-th elementary permutation of $\{1, \ldots, f(h(i))\}$ in which ρ^i_j can be decomposed is the y-th elementary permutation.

Now, let us build up a term M_{init} that, given a list L of boolean constants and a natural number $\lceil n \rceil$, 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

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

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.

5.2.1 From Q to Circuits

We prove here the converse of Theorem 5.9. This way we will complete the proof of the equivalence with quantum circuit families. We will stay more informal here: the arguments are rather intuitive.

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

Proposition 5.10. *For each qrel M there exist a term N and two total computable functions $f : \mathbb{N} \to \mathbb{N}$ and $g : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ such that for every $n \in \mathbb{N}$ and for every $c_1, \ldots, c_n$, $[1, M![!c_1,...,!c_n] \Downarrow_{n\mathscr{Q}} [|r_1 \mapsto c_{g(n,1)}, \ldots, r_{f(n)} \mapsto c_{g(n,f(n))}\rangle, N]$, where we conventionally set $c_0 \equiv 0$ and $c_{n+1} \equiv 1$.*

Let us consider $[\mathcal{Q}, M] \in \mathsf{EQT}$ and let us suppose that $[\mathcal{Q}, M] \Downarrow_{\mathscr{Q}} [\mathcal{Q}', [r_1, \ldots, r_m]]$. Then $\mathcal{Q}$ and $\mathcal{Q}'$ live in the same Hilbert space

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

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

Proposition 5.11. *Let M be a term only containing quantum redexes. Then, there is a circuit $\mathbf{K}$ such that $\mathcal{Q}' = U_{\mathbf{K}}(\mathcal{Q})$ whenever $[\mathcal{Q}, M] \Downarrow_{\mathscr{Q}} [\mathcal{Q}', M']$. Moreover, $\mathbf{K}$ is generated by gates appearing in M. Furthermore $\mathbf{K}$ can be effectively computed from M.*

As a direct consequence of propositions 5.10 and 5.11 we obtain the following:

Theorem 5.12. *For each qrel M there is a quantum circuit family (f, g, h) such that for each list $c_1, ..., c_n$ the following two conditions are equivalent:*
- $[1, M![!c_1, ..., !c_n]] \Downarrow [\mathcal{Q}, [r_1, \ldots, r_m]]$
- $m = f(n)$ *and* $\mathcal{Q} = \Phi_{f,g,h}(c_1, \ldots, c_n)$.

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

6

A Poly Time Quantum Lambda Calculus

In this Chapter, following the ICC (Implicit Computational Complexity) paradigm, we give a characterization of polytime quantum complexity classes by way of a new calculus that we call SQ, based on Lafont's Soft Linear Logic [63].

Terms and configurations of SQ form subclasses of the ones of Q, the untyped lambda calculus with classical control and quantum data previously introduced.

The correspondence with quantum complexity classes is an extensional correspondence. We proved that any term in the language can be evaluated in polynomial time (where the underlying polynomial depends on the *box depth* of the considered term); and that any problem P decidable in polynomial time (in a quantum sense) can be represented in the language (i.e., there exists a term M which decides P).

6.1 On the class of unitary operators

In this paper we will show that SQ is sound and complete with respect to *polynomial time quantum Turing machines* as defined by Bernstein and Vazirani [22] (see also Chapter 3). In particular, in order to show the "perfect" equivalence of SQ with polynomial quantum Turing machines, we need to restrict our attention to the so-called computable operators (see, e.g., the paper of Nishimura and Ozawa [74] on the perfect equivalence between quantum circuit families and quantum Turing machines).

Recall that "perfect equivalence" between a subclass quantum circuit families and polytime quantum turing machine (see, e.g., the paper of Nishimura and Ozawa [74] means a correspondence between all the three polytime quantum complexity classes BQP, EQP, ZQP and their counterpart defined on quantum circuit families.

In the first proposal by Nishimura and Ozawa [73], the equivalence holds, but it not perfect: in fact, EQP and ZQP are not equivalent to their counterpart. This is due to the different choice of quantum gates set: Nishimura and Ozawa defined the so called *(polynomial size) uniform* quantum circuit families, a subclass of QCF with polytime description function, but based on a (possibly) infinite set of quantum gates.

Subsequently, the two authors developed the so called *finitely generated* QCF (a subset of uniform QCF): quantum gates of each quantum circuit are based on a finite set of elementary gates (elementary operators) and moreover the definition of the finitely generated QCF is independent w.r.t. the choice of the universal set of quantum gates.

In order to obtain a perfect correspondence between the formalisms of QTM and QCF, both must be based on the notions of computable numbers and computable operators (see Definitions 2.19, 2.20 and 2.21).

6.2 Syntax

The syntax of terms of SQ is equal to the syntax of Q, introduced in Chapters 4 and 5. The set of well-forming rules is different from the one of Q, since we want to control the duplication of resources.

6.2.1 The Language of Terms

Let $\mathcal{U}$ be the class of computable operators on $\ell^2\{0,1\}^n$. Let us associate to each computable unitary operator $\mathbf{U} \in \mathcal{U}$ on the Hilbert space $\mathcal{H}(\{0,1\}^n)$, a symbol U. The set of the *term expressions*, or *terms* for short, is defined by the grammar in Figure 4.1:

All the assumptions adopted for Q still hold.

For every term M and for every classical variable x the number of free occurrences $\mathsf{NFO}(x, M)$ of x in M is defined as follows, by induction on M:

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

6.2.2 Judgements and Well–Formed Terms

SQ is a "refinement" of Q. In particular, we have to control the use of resources, in order to manipulate the intrinsic complexity of the system. Well forming rules of SQ are different w.r.t. the Q formulation, and in particular Weakening and Contraction rules are distinct, and Contraction is more restrictive with respect to the Turing complete case.

SQ is directly inspired to the Soft Linear Logic presented in Chapter 2, Section 2.2.3 (see in particular the similar control of structural rules): the classical fragment of SQ is very similar (essentially equivalent) to the language of terms of Baillot and Mogbil's soft lambda calculus [12], where the authors show how soft lambda terms can be typed with formulas of soft linear logic [1].

Judgements are defined from various notions of environments, taking into account the way the variables are used:

[1] many interesting properties hold for soft lambda terms even in the absence of types, i.e., the structure of untyped terms is itself sufficient to enforce those properties. This includes soundness and completeness wrt polynomial time. This is the main reason why we decided to present SQ as an untyped language.

- A *classical environment* is a (possibly empty) set (denoted by Δ, possibly indexed) of classical variables. With $!\Delta$ we denote the set $!x_1, \ldots, !x_n$ whenever Δ is $x_1, \ldots, x_n$. Analogously, with $\#\Delta$, we denote the environment $\#x_1, \ldots, \#x_n$ whenever Δ is $x_1, \ldots, x_n$. If Δ is empty, then $!\Delta$ and $\#\Delta$ are empty. Notice that if Δ is a non-empty classical environment, both $\#\Delta$ and $!\Delta$ are *not* classical environments.
- A *quantum environment* is a (possibly empty) set (denoted by Θ, possibly indexed) of quantum variables.
- A *linear environment* is a (possibly empty) set (denoted by Λ, possibly indexed) Δ, Θ of classic and quantum variables.
- A *non contractible environment* is a (possibly empty) set (denoted by Ψ, possibly indexed) $\Lambda, !\Delta$ where each variable name occurs at most once.
- An *environment* (denoted by Γ, eventually indexed) is a (possibly empty) set $\Psi, \#\Delta$ where each variable name occurs at most once.
- A *judgment* is an expression $\Gamma \vdash M$, where Γ is an environment and M is a term.
- If $\Gamma_1, \ldots, \Gamma_n$ are (not necessarily pairwise distinct) environments, $\Gamma_1 \cup \ldots \cup \Gamma_n$ denotes the environment obtained by means of the standard set-union of $\Gamma_1, \ldots, \Gamma_n$.

In all the above definitions, we are implicitly assuming that the same (quantum or classical) variable name cannot appear more than once in an environment, e.g. $x, !y, \#z$ is a correct environment, while $x, !x$ is not. Given an environment Γ, $\mathsf{var}(\Gamma)$ denotes the set of variable names in Γ.

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

$$\frac{}{!\Delta, \#x \vdash x}\ \textsf{der1} \qquad \frac{}{!\Delta, !x \vdash x}\ \textsf{der2}$$

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

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

$$\frac{\Delta_1 \vdash M}{!\Delta_2, !\Delta_1 \vdash !M}\ \textsf{prom} \qquad \frac{\Gamma \vdash M}{\Gamma \vdash \mathtt{new}(M)}\ \textsf{new} \qquad \frac{\Gamma, x_1, \ldots, x_n \vdash M}{\Gamma \vdash \lambda\langle x_1, \ldots, x_n \rangle.M}\ \multimap_1$$

$$\frac{\Gamma, x \vdash M}{\Gamma \vdash \lambda x.M}\ \multimap_2 \qquad \frac{\Gamma, \#x \vdash M}{\Gamma \vdash \lambda !x.M}\ \to_{\#} \qquad \frac{\Gamma, !x \vdash M}{\Gamma \vdash \lambda !x.M}\ \to_{!}$$

Fig. 6.1. Well Forming Rules

We say that a judgement $\Gamma \vdash M$ is *well formed* (*notation:* $\triangleright \Gamma \vdash M$) if it is derivable by means of the *well forming rules* in Figure 6.1. With $d \triangleright \Gamma \vdash M$ we denote that d is a derivation of the well formed judgement $\Gamma \vdash M$. If $\Gamma \vdash M$ is *well formed* we say also that the term M *is well formed with respect to the environment* Γ, or, simply, that M *is well formed.*

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

Proposition 6.1. *If* $\triangleright \mathbf{Q}(M) \vdash M$ *then all the classical variables in* M *are bound.*

Proof. The following, stronger, statement can be proved by structural induction on d: if $d \triangleright \Gamma \vdash M$ then all the free variables of M appear in Γ.

6.3 Computations

The notion of configuration and of reduction are exactly the same of Q.

Even if the well-forming rules of SQ are different from those of Q, the well formed terms of SQ are also well formed with respect to Q.
Therefore it is natural to adopt for SQ the same reduction rules of Q, that we gave in Figure 4.3.

6.3.1 Subject Reduction

Even if SQ is not typed, we have a strong notion of well formation for terms, as for Q. As we will see, the well forming rules are strong enough to guarantee polystep termination of computations (see section 6.7).

It is necessary to introduce a suitable notion of *well formed configuration* and, moreover, to show that well formed configurations are closed under reduction.

Definition 6.2. *A configuration* $[\mathcal{Q}, \mathcal{QV}, M]$ *is said to be well–formed iff there is a context* Γ *such that* $\Gamma \vdash M$ *is well-formed.*

Theorem 6.3 (Well Formation Closure). *If* C *is a well–formed configuration and* $C \xrightarrow{*} D$ *then* D *is well formed.*

The proof of the theorem is a consequence (provable by induction) of the following stronger result that, with a little abuse of language (the calculus is untyped), we call *subject–reduction theorem*.

Theorem 6.4 (Subject Reduction). *If* $\triangleright \Lambda, !\Delta_1, \#\Delta_2 \vdash M_1$ *and* $[\mathcal{Q}_1, \mathcal{QV}_1, M_1] \to [\mathcal{Q}_2, \mathcal{QV}_2, M_2]$ *then there are environments* Δ_3, Δ_4 *such that* $\Delta_1 = \Delta_3, \Delta_4$ *and* $\triangleright \Lambda, !\Delta_3, \#\Delta_4 \cup \#\Delta_2, \mathcal{QV}_2 - \mathcal{QV}_1 \vdash M_2$*. Moreover,* $\mathcal{QV}_2 - \mathcal{QV}_1 = \mathbf{Q}(M_2) - \mathbf{Q}(M_1)$.

In order to prove the theorem we need a number of intermediate results.

Firstly, we need three technical lemmas:

Lemma 6.5 (Weakening).
If $d \triangleright \Gamma \vdash M$ *and* x *is a fresh variable, then* $\triangleright \Gamma, !x \vdash M$

Proof. By induction on the height of d and by case on the last rule r.

- d is an axiom: trivial;
- r is app

$$\dfrac{\dfrac{d_1}{\Gamma_1, \#\Delta \vdash M_1} \qquad \dfrac{d_2}{\Gamma_2, \#\Delta \vdash M_2}}{\Gamma_1, \Gamma_2, \#\Delta \vdash M_1(M_2)}\ \mathsf{app}$$

By induction hypothesis we can derive

$$\dfrac{\dfrac{H.I.(d_1)}{\Gamma_1, \#\Delta, !x \vdash M_1} \qquad \dfrac{d_2}{\Gamma_2, \#\Delta \vdash M_2}}{\Gamma_1, \Gamma_2, \#\Delta, !x \vdash M_1(M_2)}\ \mathsf{app}$$

- r is tens

$$\dfrac{\dfrac{d_1}{\Psi_1, \#\Delta_1 \vdash M_1} \quad \dots \quad \dfrac{d_k}{\Psi_k, \#\Delta_k \vdash M_k}}{\Psi_1, \dots, \Psi_k, \#\Delta_1 \cup \dots \cup \#\Delta_k \vdash \langle M_1, \dots, M_k \rangle}\ \mathsf{tens}$$

We applying the induction hypothesis on the first subderivation, obtaining

$$\dfrac{\dfrac{I.H.(d_1)}{\Psi_1, \#\Delta_1, !x \vdash M_1} \quad \dots \quad \dfrac{d_k}{\Psi_k, \#\Delta_k \vdash M_k}}{\Psi_1, \dots, \Psi_k, \#\Delta_1 \cup \dots \cup \#\Delta_k !x \vdash \langle M_1, \dots, M_k \rangle}\ \mathsf{tens}$$

- r is new:

$$\dfrac{\dfrac{d}{\Gamma \vdash M}}{\Gamma \vdash \mathsf{new}(M)}\ \mathsf{new}$$

and by I.H. we have

$$\dfrac{\dfrac{I.H.(d)}{\Gamma, !x \vdash M}}{\Gamma, !x \vdash \mathsf{new}(M)}\ \mathsf{new}$$

- r is $\multimap_1$

$$\dfrac{\begin{array}{c} d \\ \vdots \\ \Gamma, y_1, \ldots, y_n \vdash M \end{array}}{\Gamma \vdash \lambda\langle y_1, \ldots, y_n\rangle.M} \multimap_1$$

By I.H. we can conclude

$$\dfrac{\begin{array}{c} I.H.(d) \\ \vdots \\ \Gamma, !x, y_1, \ldots, y_n \vdash M \end{array}}{\Gamma, !x \vdash \lambda\langle y_1, \ldots, y_n\rangle.M} \multimap_1$$

- r is $\multimap_2$: as for the previous case;
- r is $\rightarrow_{\#}$. We have

$$\dfrac{\begin{array}{c} d \\ \vdots \\ \Gamma, \#y \vdash M \end{array}}{\Gamma \vdash \lambda!y.M} \rightarrow_{\#}$$

and by I.H.

$$\dfrac{\begin{array}{c} H.I.(d) \\ \vdots \\ \Gamma, !x, \#y \vdash M \end{array}}{\Gamma, !x \vdash \lambda!y.M} \rightarrow_{\#}$$

- r is $\rightarrow_!$. We have

$$\dfrac{\begin{array}{c} d \\ \vdots \\ \Gamma, !y \vdash M \end{array}}{\Gamma \vdash \lambda!y.M} \rightarrow_!$$

and by I.H.

$$\dfrac{\begin{array}{c} H.I.(d) \\ \vdots \\ \Gamma, !x, !y \vdash M \end{array}}{\Gamma, !x \vdash \lambda!y.M} \rightarrow_!$$

- r is prom

$$\dfrac{\begin{array}{c} d \\ \vdots \\ \Delta_1 \vdash M \end{array}}{!\Delta_2, !\Delta_1 \vdash !M} \mathsf{prom}$$

By means of promotion rule, we can conclude

$$\dfrac{\begin{array}{c} d \\ \vdots \\ \Delta_1 \vdash M \end{array}}{!\Delta_2, !\Delta_1, !x \vdash !M}\ \mathsf{prom}$$

.

Lemma 6.6 ($\# \vdash$).
If $d \triangleright \Gamma, x \vdash M$*, then* $\triangleright \Gamma, \#x \vdash N$.

Proof. By induction on the eight of derivation, and by case on the last rule r.

- r is the axiom classic-var: $\dfrac{}{\Delta, \#x \vdash x}\ \mathsf{der}_1$ holds too, by means of der_1 axiom;
- r is app: we must distinguish between two cases.
  1. We have
  $$\dfrac{\begin{array}{c} d_1 \\ \vdots \\ \Gamma_1, \#\Delta, x \vdash M_1 \end{array} \qquad \begin{array}{c} d_2 \\ \vdots \\ \Gamma_2, \#\Delta \vdash M_2 \end{array}}{\Gamma_1, \Gamma_2, \#\Delta, x \vdash M_1(M_2)}\ \mathsf{app}$$
  (where $\Gamma = \Gamma_1, \Gamma_2$) and by induction hypothesis we can conclude
  $$\dfrac{\begin{array}{c} I.H.(d_1) \\ \vdots \\ \Gamma_1, \#\Delta, \#x \vdash M_1 \end{array} \qquad \begin{array}{c} d_2 \\ \vdots \\ \Gamma_2, \#\Delta \vdash M_2 \end{array}}{\Gamma_1, \Gamma_2, \#\Delta, \#x \vdash M_1(M_2)}\ \mathsf{app}$$
  2. symmetrically to the previous case, with induction hypothesis applied on d_2;
- r is tens:
  $$\dfrac{\begin{array}{c} d_1 \\ \vdots \\ \Psi_1, \#\Delta_1 \vdash M_1 \end{array} \quad \ldots \quad \begin{array}{c} d_j \\ \vdots \\ \Psi_j, \#\Delta_j, x \vdash M_j \end{array} \quad \ldots \quad \begin{array}{c} d_k \\ \vdots \\ \Psi_k, \#\Delta_k \vdash M_k \end{array}}{\Psi_1, \ldots, \Psi_k, \#\Delta_1 \cup \ldots \cup \#\Delta_j \cup \ldots \cup \#\Delta_k, x \vdash \langle M_1, \ldots, M_k \rangle}\ \mathsf{tens}$$
  We apply the induction hypothesis on derivation d_j and we conclude:
  $$\dfrac{\begin{array}{c} d_1 \\ \vdots \\ \Psi_1, \#\Delta_1 \vdash M_1 \end{array} \quad \ldots \quad \begin{array}{c} I.H.(d_j) \\ \vdots \\ \Psi_j, \#\Delta_j, \#x \vdash M_j \end{array} \quad \ldots \quad \begin{array}{c} d_k \\ \vdots \\ \Psi_k, \#\Delta_k \vdash M_k \end{array}}{\Psi_1, \ldots, \Psi_k, \#\Delta_1 \cup \ldots \cup \#\Delta_j \cup \ldots \cup \#\Delta_k, \#x \vdash \langle M_1, \ldots, M_k \rangle}\ \mathsf{tens}$$
- r is new:
  $$\dfrac{\begin{array}{c} d \\ \vdots \\ \Gamma, x \vdash M \end{array}}{\Gamma, x \vdash \mathsf{new}(M)}\ \mathsf{new}$$

and by I.H.

$$\dfrac{\begin{array}{c} I.H.(d) \\ \vdots \\ \Gamma, \# x \vdash M \end{array}}{\Gamma, \# x \vdash \mathsf{new}(M)}\ \mathsf{new}$$

- r is $\multimap_1$:

$$\dfrac{\begin{array}{c} d \\ \vdots \\ \Gamma, x, x_1, \ldots, x_n \vdash M \end{array}}{\Gamma, x \vdash \lambda\langle x_1, \ldots, x_n\rangle M}\ \multimap_1$$

  Applying the induction hypothesis to d we obtain

$$\dfrac{\begin{array}{c} I.H.(d) \\ \vdots \\ \Gamma, \# x, x_1, \ldots, x_n \vdash M \end{array}}{\Gamma, \# x \vdash \lambda\langle x_1, \ldots, x_n\rangle M}\ \multimap_1$$

- r is $\multimap_2$: very similar to the previous case;
- r is $\rightarrow_{\#}$:

$$\dfrac{\begin{array}{c} d \\ \vdots \\ \Gamma, x, \# y \vdash M \end{array}}{\Gamma, x \vdash \lambda ! y M}\ \rightarrow_{\#}$$

  and applying the induction hypothesis to d we have

$$\dfrac{\begin{array}{c} I.H.(d) \\ \vdots \\ \Gamma, \# x, \# y \vdash M \end{array}}{\Gamma, \# x \vdash \lambda ! y M}\ \rightarrow_{\#}$$

- r is $\rightarrow_!$: very similar to the previous case.

Lemma 6.7 **($!x \vdash$).**
If $d \triangleright \Gamma, x \vdash M$ *then* $\triangleright \Gamma, !x \vdash N$.

Proof. By induction on the height of the derivation and by case on the last rule r. Note that r can not be prom

- r is

$$\dfrac{}{!\Delta, x \vdash x}$$

  We conclude taking

$$\dfrac{}{!\Delta, !x \vdash x}\ \mathsf{der}_2$$

- r is app: we distinguish two cases.

1. We have

$$\dfrac{\begin{array}{c} d_1 \\ \vdots \\ \Gamma_1, \#\Delta, x \vdash M_1 \end{array} \qquad \begin{array}{c} d_2 \\ \vdots \\ \Gamma_2, \#\Delta \vdash M_2 \end{array}}{\Gamma_1, \Gamma_2, \#\Delta, x \vdash M_1(M_2)}\ \mathsf{app}$$

(where $\Gamma = \Gamma_1, \Gamma_2$) and by induction hypothesis we can conclude

$$\dfrac{\begin{array}{c} I.H.(d_1) \\ \vdots \\ \Gamma_1, \#\Delta, !x \vdash M_1 \end{array} \qquad \begin{array}{c} d_2 \\ \vdots \\ \Gamma_2, \#\Delta \vdash M_2 \end{array}}{\Gamma_1, \Gamma_2, \#\Delta, !x \vdash M_1(M_2)}\ \mathsf{app}$$

2. symmetrically to the previous case, with induction hypothesis applied on d_2;

- r is tens:

$$\dfrac{\begin{array}{c} d_1 \\ \vdots \\ \Psi_1, \#\Delta_1 \vdash M_1 \end{array} \quad \ldots \quad \begin{array}{c} d_j \\ \vdots \\ \Psi_j, \#\Delta_j, x \vdash M_j \end{array} \quad \ldots \quad \begin{array}{c} d_k \\ \vdots \\ \Psi_k, \#\Delta_k \vdash M_k \end{array}}{\Psi_1, \ldots, \Psi_k, \#\Delta_1 \cup \ldots \cup \#\Delta_j \cup \ldots \cup \#\Delta_k, x \vdash \langle M_1, \ldots, M_k \rangle}\ \mathsf{tens}$$

We apply the induction hypothesis on derivation d_j and we conclude:

$$\dfrac{\begin{array}{c} d_1 \\ \vdots \\ \Psi_1, \#\Delta_1 \vdash M_1 \end{array} \quad \ldots \quad \begin{array}{c} I.H.(d_j) \\ \vdots \\ \Psi_j, \#\Delta_j, !x \vdash M_j \end{array} \quad \ldots \quad \begin{array}{c} d_k \\ \vdots \\ \Psi_k, \#\Delta_k \vdash M_k \end{array}}{\Psi_1, \ldots, \Psi_k, \#\Delta_1 \cup \ldots \cup \#\Delta_j \cup \ldots \cup \#\Delta_k, !x \vdash \langle M_1, \ldots, M_k \rangle}\ \mathsf{tens}$$

- r is new:

$$\dfrac{\begin{array}{c} d \\ \vdots \\ \Gamma, x \vdash M \end{array}}{\Gamma, x \vdash \mathsf{new}(M)}\ \mathsf{new}$$

and by I.H.

$$\dfrac{\begin{array}{c} I.H.(d) \\ \vdots \\ \Gamma, !x \vdash M \end{array}}{\Gamma, !x \vdash \mathsf{new}(M)}\ \mathsf{new}$$

- r is $\multimap_1$:

$$\dfrac{\begin{array}{c} d \\ \vdots \\ \Gamma, x, x_1, \ldots, x_n \vdash M \end{array}}{\Gamma, x \vdash \lambda\langle x_1, \ldots, x_n \rangle M}\ \multimap_1$$

Applying the induction hypothesis to d we obtain

$$\dfrac{\begin{array}{c} I.H.(d) \\ \vdots \\ \Gamma, !x, x_1, \dots, x_n \vdash M \end{array}}{\Gamma, !x \vdash \lambda\langle x_1, \dots, x_n\rangle M} \multimap_1$$

- r is $\multimap_2$: as for previous case;
- r is $\to_\#$:

$$\dfrac{\begin{array}{c} d \\ \vdots \\ \Gamma, x, \#y \vdash M \end{array}}{\Gamma, x \vdash \lambda!yM} \to_\#$$

  and applying the induction hypothesis to d we have

$$\dfrac{\begin{array}{c} I.H.(d) \\ \vdots \\ \Gamma, !x, \#y \vdash M \end{array}}{\Gamma, !x \vdash \lambda!yM} \to_\#$$

- r is $\to_!$: as for previous case.

As a consequence of the proved lemmata, the following rules are admissible:

$$\frac{\Gamma \vdash M}{\Gamma, !\Delta \vdash M}\,\mathsf{w} \qquad \frac{\Gamma, \Delta \vdash M}{\Gamma, !\Delta \vdash M}\,L! \qquad \frac{\Gamma, \Delta \vdash M}{\Gamma, \#\Delta \vdash M}\,L\#$$

with the proviso that in rule w, each x in Δ is a fresh variable.

As always, proving subject reduction requires some substitution lemmata too. In this case, we need four distinct substitution lemmata:

Lemma 6.8 (Substitution lemma).
Actually we have four distinct cases:

Linear Case. *If* $d_1 \rhd \Psi_1, \#\Delta_1, x \vdash P$ *and* $d_2 \rhd \Psi_2, \#\Delta_2 \vdash N$*, with* $\mathsf{var}(\Psi_1) \cap \mathsf{var}(\Psi_2) = \emptyset$*, then* $\rhd \Psi_1, \Psi_2, \#\Delta_1 \cup \#\Delta_2 \vdash P\{N/x\}$*.*

Contraction Case. *If* $d_1 \rhd \Gamma, \#x \vdash P$ *and* $d_2 \rhd \Delta \vdash N$ *and* $\mathsf{var}(\Gamma) \cap \mathsf{var}(\Delta) = \emptyset$ *then* $\rhd \Gamma, \#\Delta \vdash P\{N/x\}$*.*

Bang Case. *If* $d_1 \rhd \Gamma, !x \vdash P$ *and* $d_2 \rhd \Delta \vdash N$ *and* $\mathsf{var}(\Gamma) \cap \mathsf{var}(\Delta) = \emptyset$ *then* $\rhd \Gamma, !\Delta \vdash P\{N/x\}$*.*

Quantum Case. *If* $d_1 \rhd \Gamma, x_1, \dots, x_n \vdash P$*,* $d_2 \rhd !\Delta, r_1, \dots, r_n \vdash \langle r_1, \dots, r_n\rangle$ *and* $r_1, \dots, r_n \notin \mathsf{var}(\Gamma)$ *then* $\rhd \Gamma, !\Delta, r_1, \dots, r_n \vdash P\{r_1/x_1, \dots, r_n/x_n\}$

Proof. Each case is proved by induction on the height of the derivation and by cases on the last rule.

Linear case

- r is the axiom $\dfrac{d_1}{!\Delta_1, x \vdash x}$: we use the weakening rule and from d_2 we obtain

$$\dfrac{\begin{array}{c} d_2 \\ \vdots \\ \Psi_2, \#\Delta_2 \vdash N \end{array}}{!\Delta_1, \Psi_2, \#\Delta_2 \vdash N}\ \mathsf{w}$$

- r is app
  1. $$\dfrac{\begin{array}{c} d_1' \\ \vdots \\ \Psi_1', \#\Delta_1', x \vdash M_1 \end{array} \qquad \begin{array}{c} d_1'' \\ \vdots \\ \Psi_1'', \#\Delta_1'' \vdash M_2 \end{array}}{\Psi_1', \Psi_1'', \#\Delta_1' \cup \#\Delta_1'', x \vdash M_1(M_2)}$$

     We apply the induction hypothesis on the derivation d_1' and d_2, obtaining:

     $$\dfrac{\begin{array}{c} I.H.(d_1', d_2) \\ \vdots \\ \Psi_1', \Psi_2, \#\Delta_1' \cup \#\Delta_2, \vdash M_1\{N/x\} \end{array} \qquad \begin{array}{c} d_1'' \\ \vdots \\ \Psi_1'', \#\Delta_1'' \vdash M_2 \end{array}}{\Psi_1', \Psi_2, \Psi_1'', \#\Delta_1' \cup \#\Delta_2 \cup \#\Delta_1'', x \vdash M_1\{N/x\}(M_2)}$$

  2. x occurs in the right branch of the rule: symmetrically to the first case;

- r is tens:

  $$\dfrac{\begin{array}{c} d_1^1 \\ \vdots \\ \Phi_1^1, \#\Delta_1^1 \vdash M_1 \end{array} \quad \cdots \quad \begin{array}{c} d_1^j \\ \vdots \\ \Phi_1^j, \#\Delta_1^j, x \vdash M_j \end{array} \quad \cdots \quad \begin{array}{c} d_1^k \\ \vdots \\ \Phi_1^k, \#\Delta_1^1 \vdash M_k \end{array}}{\Psi_1^1, \ldots, \Psi_1^j, \ldots, \Psi_1^k, \#\Delta_1^1 \cup \ldots \cup \#\Delta_1^j \cup \ldots \cup \#\Delta_1^k, x \vdash \langle M_1, \ldots, M_j, \ldots, M_k \rangle}\ \mathsf{tens}$$

  We apply the induction hypothesis to d_1^j and d_2 and we conclude

  $$\dfrac{\begin{array}{c} d_1^1 \\ \vdots \\ \Phi_1^1, \#\Delta_1^1 \vdash M_1 \end{array} \quad \cdots \quad \begin{array}{c} I.H.(d_1^j, d_2) \\ \vdots \\ \Phi_1^j, \Psi_2, \#\Delta_1^j \cup \#\Delta_2 \vdash M_j\{N/x\} \end{array} \quad \cdots \quad \begin{array}{c} d_1^k \\ \vdots \\ \Phi_1^k, \#\Delta_1^1 \vdash M_k \end{array}}{\Psi_1^1, \ldots, \Psi_1^j, \ldots, \Psi_1^k, \Psi_2, \#\Delta_1^1 \cup \ldots \cup \#\Delta_1^j \cup \ldots \cup \#\Delta_1^k \cup \#\Delta_2, \vdash \langle M_1, \ldots, M_j\{N/x\}, \ldots, M_k \rangle}\ \mathsf{tens}$$

- r is new:

  $$\dfrac{\begin{array}{c} d_1' \\ \vdots \\ \Psi_1, \#\Delta_1, x \vdash M \end{array}}{\Psi_1, \#\Delta_1, x \vdash \mathsf{new}(M)}\ \mathsf{new}$$

By induction hypothesis: new:

$$\dfrac{\begin{matrix} I.H.(d_1', d_2) \\ \vdots \\ \Psi_1, \Psi_2 \# \Delta_1 \cup \# \Delta_2 \vdash M\{N/x\} \end{matrix}}{\Psi_1, \Psi_2, \# \Delta_1 \cup \# \Delta_2, \vdash \mathsf{new}(M\{N/x\})} \; \mathsf{new}$$

- r is $\multimap_1$:

$$\dfrac{\begin{matrix} d_1' \\ \vdots \\ \Psi_1, \# \Delta_1, x, y_1, \ldots, y_n \vdash M \end{matrix}}{\Psi_1, \# \Delta_1, x \vdash \lambda\langle y_1, \ldots, y_n \rangle.M} \multimap_1$$

and by induction hypothesis we have

$$\dfrac{\begin{matrix} I.H.(d_1', d_2) \\ \vdots \\ \Psi_1, \Psi_2 \# \Delta_1 \cup \# \Delta_2, y_1, \ldots, y_n \vdash M\{N/x\} \end{matrix}}{\Psi_1, \Psi_2, \# \Delta_1 \cup \# \Delta_2, \vdash \lambda\langle y_1, \ldots, y_n \rangle.M\{N/x\}} \multimap_1$$

- r is $\multimap_2$: as for the previous case.

- r is $\rightarrow_{\#}$:

$$\dfrac{\begin{matrix} d_1' \\ \vdots \\ \Psi_1, \# \Delta_1, x, \# y \vdash M \end{matrix}}{\Psi_1, \# \Delta_1, x \vdash \lambda ! y.M} \multimap_1$$

We obtain the result applying the induction hypothesis on d_1' and d_2:

$$\dfrac{\begin{matrix} H.I.(d_1', d_2) \\ \vdots \\ \Psi_1, \Psi_2, \# \Delta_1 \cup \# \Delta_2, \# y \vdash M\{N/x\} \end{matrix}}{\Psi_1, \Psi_2, \# \Delta_1 \cup \# \Delta_2 \vdash \lambda ! y.M} \multimap_1$$

Note that if $\# y$ belongs to Ψ_2 we have to use the α-renaming procedure.

- r is $\rightarrow_!$: very similar to the previous case.

Contraction Case

r is der_1: $\dfrac{d_1}{!\Delta, \# x \vdash x}$. We use auxiliary rules $L\#$ and w:

$$\dfrac{\dfrac{\begin{array}{c} d_2 \\ \vdots \\ \Delta \vdash N \end{array}}{\#\Delta \vdash N}\, L\#}{!\Delta_1, \#\Delta \vdash N}\,\mathsf{w}$$

- r is app: we have to distinguish between three different case:
  1. $\#x$ belongs to both the contexts: we have

  $$\dfrac{\begin{array}{c} d_1' \\ \vdots \\ \Psi_1, \#\Delta_{11}, \#x \vdash M_1 \end{array} \qquad \begin{array}{c} d_1'' \\ \vdots \\ \Psi_2, \#\Delta_{12}, \#x \vdash M_2 \end{array}}{\Psi_1, \Psi_2, \#\Delta_{11} \cup \#\Delta_{12}, \#x \vdash M_1(M_2)}\,\mathsf{app}$$

  and by induction hypothesis

  $$\dfrac{\begin{array}{c} I.H.(d_1', d_2) \\ \vdots \\ \Psi_1, \#\Delta_{11} \cup \#\Delta \vdash M_1\{N/x\} \end{array} \qquad \begin{array}{c} I.H.(d_1'', d_2) \\ \vdots \\ \Psi_2, \#\Delta_{12} \cup \#\Delta \vdash M_2\{N/x\} \end{array}}{\Psi_1, \Psi_2, \#\Delta_{11} \cup \#\Delta_{12} \cup \#\Delta \vdash (M_1(M_2))\{N/x\}}\,\mathsf{app}$$

  2. $\#x$ belongs to the context of the conclusions of d_1': we have

  $$\dfrac{\begin{array}{c} d_1' \\ \vdots \\ \Psi_1, \#\Delta_{11}, \#x \vdash M_1 \end{array} \qquad \begin{array}{c} d_1'' \\ \vdots \\ \Psi_2, \#\Delta_{12}, \vdash M_2 \end{array}}{\Psi_1, \Psi_2, \#\Delta_{11} \cup \#\Delta_{12}, \#x \vdash M_1(M_2)}\,\mathsf{app}$$

  and by induction hypothesis

  $$\dfrac{\begin{array}{c} I.H.(d_1', d_2) \\ \vdots \\ \Psi_1, \#\Delta_{11} \cup \#\Delta \vdash M_1\{N/x\} \end{array} \qquad \Psi_2, \#\Delta_{12} \vdash M_2}{\Psi_1, \Psi_2, \#\Delta_{11} \cup \#\Delta_{12} \cup \#\Delta \vdash (M_1(M_2))\{N/x\}}\,\mathsf{app}$$

  3. $\#x$ belongs to the context of the conclusions of d_1'': as for the previous one;
- r is tens:

$$\dfrac{\begin{array}{c} d_1' \\ \vdots \\ \Psi_1', \#\Delta_1' \vdash M_1 \end{array} \quad \cdots \quad \begin{array}{c} d_1^k \\ \vdots \\ \Psi_1^k, \#\Delta_1^k \vdash M_k \end{array}}{\Psi_1^1, \ldots . \Psi_1^k, \#\Delta_1^k \cup \ldots \cup \#\Delta_1^k \vdash \langle M_1, \ldots, M_k \rangle}\,\mathsf{tens}$$

where $x \in \#\Delta_1^k \cup \ldots \cup \#\Delta_1^k$.

We apply the induction hypothesis to d_1^j and d_2 for all j and we obtain

$$\dfrac{\cdots \quad \begin{array}{c} I.H.(d_1^j, d_2) \\ \vdots \\ \Psi_1^j, \#\tilde{\Delta}_1^j \vdash M_j\{n/x\} \end{array} \quad \cdots}{\Psi_1^1, \ldots .\Psi_1^k, \#\tilde{\Delta}_1^k \cup \ldots \cup \#\tilde{\Delta}_1^k \vdash \langle M_1, \ldots, M_j\{N/x\} \ldots, M_k\rangle}\ \mathsf{tens}$$

where for all $j = 1 \ldots k$, $\#\tilde{\Delta}_1^j = \#\Delta_1^j$ if $\#x \notin \#\Delta_1^j$ and $\#\tilde{\Delta}_1^j = (\#\Delta_1^j - \{\#x\}) \cup \#\Delta$ otherwise;

- r is prom: in this case $\#x$ does not belong to the context and so the premise is non satisfied;
- r is new and we have

$$\dfrac{\begin{array}{c} d_1' \\ \vdots \\ \Gamma, \#x \vdash M \end{array}}{\Gamma, \#x \vdash \mathsf{new}(M)}\ \mathsf{new}$$

we apply the induction hypothesis on d_1 and d_2:

$$\dfrac{\begin{array}{c} I.H.(d_1', d_2) \\ \vdots \\ \Gamma, \#\Delta \vdash M\{N/x\} \end{array}}{\Gamma, \#\Delta \vdash (\mathsf{new}(M))\{N/x\}}\ \mathsf{new}$$

- r is $\multimap_1$: we have

$$\dfrac{\begin{array}{c} d_1' \\ \vdots \\ \Gamma, \#x, x_1, \ldots, x_n \vdash M \end{array}}{\Gamma, \#x, \vdash \lambda\langle x_1, \ldots, x_n\rangle.M}\ \multimap_1$$

and by I.H.

$$\dfrac{\begin{array}{c} I.H.(d_1', d_2) \\ \vdots \\ \Gamma, \#\Delta, x_1, \ldots, x_n \vdash M\{N/x\} \end{array}}{\Gamma, \#\Delta \vdash (\lambda\langle x_1, \ldots, x_n\rangle.M)\{N/x\}}\ \multimap_1$$

- r is $\multimap_2$: as for the previous case;
- r is $\rightarrow_\#$: we have

$$\dfrac{\begin{array}{c} d_1' \\ \vdots \\ \Gamma, \#x, \#y \vdash M \end{array}}{\Gamma, \#x, \vdash \lambda !y.M}\ \rightarrow_\#$$

and by I.H.

$$\cfrac{\begin{array}{c} d'_1 \\ \vdots \\ \Gamma, \#\Delta, \#y \vdash M\{N/x\} \end{array}}{\Gamma, \#\Delta \vdash (\lambda ! y.M)\{N/x\}} \to_{\#}$$

- r is $\to_!$: as for previous case.

Bang Case

- r is der_1, there are two cases:

  1. if d_1 is
  $$\frac{d_1}{!\Gamma, !x, \bigstar z \vdash z}$$
  where $\bigstar z$ is either $!z$ or $\#z$ with $z \neq x$, we have
  $$\frac{}{!\Gamma, !\Delta, \bigstar z \vdash z}$$
  2. if d_1 is
  $$\frac{d_1}{!\Delta, !x \vdash x}$$
  we use auxiliary rules $L!$ and $\mathsf{w} \vdash$:
  $$\cfrac{\cfrac{\begin{array}{c} d_2 \\ \vdots \\ \Delta \vdash N \end{array}}{!\Delta \vdash N} L!}{!\Delta_1, !\Delta \vdash N} \mathsf{w}$$

- r is app: we have to distinguish between two different case:
  1. $\#x$ belongs to the context of the conclusions of d'_1: we have
  $$\cfrac{\begin{array}{c} d'_1 \\ \vdots \\ \Psi_1, \#\Delta_{11}, !x \vdash M_1 \end{array} \qquad \begin{array}{c} d''_1 \\ \vdots \\ \Psi_2, \#\Delta_{12}, \vdash M_2 \end{array}}{\Psi_1, \Psi_2, \#\Delta_{11} \cup \#\Delta_{12}, !x \vdash M_1(M_2)} \mathsf{app}$$
  and by induction hypothesis
  $$\cfrac{\begin{array}{c} I.H.(d'_1, d_2) \\ \vdots \\ \Psi_1, \#\Delta_{11}, !\Delta \vdash M_1\{N/x\} \end{array} \qquad \Psi_2, \#\Delta_{12} \vdash M_2}{\Psi_1, \Psi_2, \#\Delta_{11} \cup \#\Delta_{12}, !\Delta \vdash (M_1(M_2))\{N/x\}} \mathsf{app}$$

2. $\#x$ belongs to the context of the conclusions of d_1'': as for previous one;

- r is tens:

$$\dfrac{\begin{array}{c} d_1^1 \\ \vdots \\ \Psi_1^1, \#\Delta_1^1 \vdash M_1 \end{array} \quad \cdots \quad \begin{array}{c} d_1^j \\ \vdots \\ \Psi_1^j, \#\Delta_1^j, !x \vdash M_j \end{array} \quad \cdots \quad \begin{array}{c} d_1^k \\ \vdots \\ \Psi_1^k, \#\Delta_1^1 \vdash M_k \end{array}}{\Psi_1^1, \ldots, \Psi_1^k, \#\Delta_1^1 \cup \ldots \cup \#\Delta_1^k, !x \vdash \langle M_1, \ldots, M_j, \ldots, M_k \rangle} \ \mathsf{tens}$$

We apply the induction hypothesis to d_1^j and d_2 and we conclude

$$\dfrac{\begin{array}{c} d_1^1 \\ \vdots \\ \Psi_1^1, \#\Delta_1^1 \vdash M_1 \end{array} \quad \cdots \quad \begin{array}{c} I.H.(d_1^j, d_2) \\ \vdots \\ \Psi_1^j, \Psi_2, \#\Delta_1^j, !\Delta \vdash M_j\{N/x\} \end{array} \quad \cdots \quad \begin{array}{c} d_1^k \\ \vdots \\ \Psi_1^k, \#\Delta_1^1 \vdash M_k \end{array}}{\Psi_1^1, \ldots, \Psi_1^k, \Psi_2, \#\Delta_1^1 \cup \ldots \cup \#\Delta_1^k, !\Delta \vdash \langle M_1, \ldots, M_j\{N/x\}, \ldots, M_k \rangle} \ \mathsf{tens}$$

- r is prom: we have two cases
  1. x is introduced by weakening:

$$\dfrac{\begin{array}{c} d_1' \\ \vdots \\ \Delta_1' \vdash P \end{array}}{!\Delta_1'', !\Delta_1', !x \vdash !P}$$

  then x does not occurs in P and so

$$\dfrac{\begin{array}{c} d_1' \\ \vdots \\ \Delta_1' \vdash P \end{array}}{!\Delta_1', !\Delta \vdash !P}$$

  2. x belongs to the context:

$$\dfrac{\begin{array}{c} d_1' \\ \vdots \\ \Delta_1', x \vdash P \end{array}}{!\Delta_1'', !\Delta_1', !x \vdash !P}$$

  We apply the linear case of the lemma on subderivations d_1' and d_2

$$\dfrac{\Delta_1', \Delta \vdash P\{N/x\}}{!\Delta_1'', !\Delta_1', !\Delta \vdash !P\{N/x\}}$$

- r is new and we have

$$\dfrac{\begin{array}{c} d_1' \\ \vdots \\ \varGamma, !x \vdash M \end{array}}{\varGamma, !x \vdash \mathsf{new}(M)}\ \mathsf{new}$$

we apply the induction hypothesis on d_1 and d_2:

$$\dfrac{\begin{array}{c} I.H.(d_1', d_2) \\ \vdots \\ \varGamma, !\varDelta \vdash M\{N/x\} \end{array}}{\varGamma, !\varDelta \vdash \mathsf{new}((M))\{N/x\}}\ \mathsf{new}$$

- r is $\multimap_1$: we have

$$\dfrac{\begin{array}{c} d_1' \\ \vdots \\ \varGamma, !x, x_1, \ldots, x_n \vdash M \end{array}}{\varGamma, !x, \vdash \lambda\langle x_1, \ldots, x_n\rangle.M}\ \multimap_1$$

and by I.H.

$$\dfrac{\begin{array}{c} I.H.(d_1', d_2) \\ \vdots \\ \varGamma, !\varDelta, x_1, \ldots, x_n \vdash M\{N/x\} \end{array}}{\varGamma, !\varDelta \vdash (\lambda\langle x_1, \ldots, x_n\rangle.M)\{N/x\}}\ \multimap_1$$

- r is $\multimap_2$: as for the previous case;
- r is $\rightarrow_\#$: we have

$$\dfrac{\begin{array}{c} d_1' \\ \vdots \\ \varGamma, !x, \#y \vdash M \end{array}}{\varGamma, !x, \vdash \lambda !y.M}\ \rightarrow_\#$$

and by I.H.

$$\dfrac{\begin{array}{c} d_1' \\ \vdots \\ \varGamma, !\varDelta, \#y \vdash M\{N/x\} \end{array}}{\varGamma, !\varDelta \vdash (\lambda !y.M)\{N/x\}}\ \rightarrow_\#$$

- r is $\rightarrow$: very similar to the previous case.

Quantum Case We use the linear case of the lemma and weakening rule in order to prove the quantum case:

$$
\dfrac{\dfrac{\dfrac{\Gamma, x_1, \dots, x_n \vdash M \quad r_1 \vdash r_1}{\Gamma, r_1, x_2, \dots, x_n \vdash M\{r_1/x_1\}}\text{Linear Case}}{\begin{array}{c}\vdots\\ \Gamma, r_1, \dots, r_{n-1}, x_n \vdash M\{r_1/x_1, \dots, r_{n-1}/x_{n-1}\} \qquad r_n \vdash r_n\end{array}}}{\dfrac{\Gamma, r_1, \dots, r_n \vdash M\{r_1/x_1, \dots, r_n/x_n\}}{\Gamma, !\Delta, r_1, \dots, r_n \vdash M\{r_1/x_1, \dots, r_n/x_n\}}\text{weak}}\text{Linear Case}
$$

Finally, we are able to prove the Subject Reduction Theorem:

Proof of the Theorem 6.4

The proof of the Theorem it is not completely standard. The main difficulty is the presence of the patterns $!x$ and $\#x$. They both are "mapping" into $\lambda!x$ by means of the $\to_!$ and $\to_\#$ rules.
Therefore, we need the following partial function.

Let Γ be an environment. A partial function m_Γ with domain Γ is called an *m–fun* (for Γ) if:

1. if α occurring in Γ is either a classical variable, or a quantum variable, or has the shape $\#x$ then $m_\Gamma(\alpha) = \alpha$;
2. if $!x$ occurs in Γ then $m_\Gamma(!x)$ is either $!x$ or $\#x$.

It is immediate to observe that:

1. if $\Gamma = \alpha_1, \dots, \alpha_n$ is an environment and m_Γ is an *m–fun*, then $m_\Gamma[\Gamma] = m_\Gamma(\alpha_1), \dots, m_\Gamma(\alpha_n)$ is an environment;
2. if Γ_1, Γ_2 are environments and $m_{\Gamma_1}, m_{\Gamma_2}$ are *m–fun*s, then the union $m_{\Gamma_1} \cup m_{\Gamma_2}$ is an *m–fun* for $\Gamma_1 \cup \Gamma_2$.

We are now in a position to prove Theorem 6.4. We prove it in the following equivalent formulation:
if $d \triangleright \Gamma \vdash M$ and $[\mathcal{Q}_1, \mathcal{QV}_1, M_1] \to [\mathcal{Q}_2, \mathcal{QV}_2, M_2]$ then there is an *m–fun* m_Γ such that $\triangleright\, m_\Gamma[\Gamma], \mathcal{QV}_2 - \mathcal{QV}_1 \vdash M_2$.
The proof is by induction on the height of d and by cases on the last rule r of d. There are several cases.

- r is app:

$$
\dfrac{\begin{array}{c} d_1 \\ \vdots \\ \Psi_1, \#\Delta_1 \vdash P_1 \end{array} \qquad \begin{array}{c} d_2 \\ \vdots \\ \Psi_2, \#\Delta_2 \vdash P_2 \end{array}}{\Psi_1, \Psi_2, \#\Delta_1 \cup \#\Delta_2 \vdash P_1 P_2}\,\mathsf{app}\ .
$$

  and the reduction rule is

$$\frac{[\mathcal{Q}, \mathcal{QV}, P_1] \to_\alpha [\mathcal{Q}', \mathcal{QV}', P_1']}{[\mathcal{Q}, \mathcal{QV}, P_1 P_2] \to_\alpha [\mathcal{Q}', \mathcal{QV}', P_1' P_2]} \ \mathsf{l.a}$$

Applying the induction hypothesis to d_1 there are an *m–fun* m and a derivation d_3 such that:

$$\frac{\dfrac{d_3}{m[\Psi_1], \#\Delta_1, \mathcal{QV}' - \mathcal{QV} \vdash P_1} \qquad \dfrac{d_2}{\Psi_2, \#\Delta_2 \vdash P_2}}{m[\Psi_1], \Psi_2, \mathcal{QV}' - \mathcal{QV}, \#\Delta_1 \cup \#\Delta_2 \vdash P_1 P_2} \ \mathsf{app}.$$

Please note that if $!y$ occur in Ψ_1 then $\#y$ cannot occur neither in $\#\Delta_1$ nor in $\#\Delta_2$, therefore also if $m(!y) = \#y$, the rule app is applied correctly.

- r is app:

$$\frac{\dfrac{\dfrac{d_1}{\Gamma, \#x \vdash P}}{\Gamma \vdash \lambda !x.P} \to_\# \qquad \dfrac{\dfrac{d_2}{\Delta_2 \vdash N}}{!\Delta_3, !\Delta_2 \vdash !N} \ \mathsf{prom}}{\Gamma, !\Delta_3, !\Delta_2 \vdash (\lambda !x.P)(!N)} \ \mathsf{app}$$

and the reduction rule is:

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

we have the thesis by means of one of the lemmas above:

$$\frac{\dfrac{d_3}{\Gamma, \#\Delta_2 \vdash P\{N/x\}}}{\Gamma, !\Delta_3, \#\Delta_2 \vdash P\{N/x\}} \ \mathsf{w}$$

where d_3 is the derivation obtained applying the contraction case of Lemma 6.8 above to d_1 and d_2.

- r is app:

$$\frac{\dfrac{\dfrac{d_1}{\Gamma, !x \vdash P}}{\Gamma \vdash \lambda !x.P} \to_! \qquad \dfrac{\dfrac{d_2}{\Delta_2 \vdash N}}{!\Delta_3, !\Delta_2 \vdash !N} \ \mathsf{prom}}{\Gamma, !\Delta_3, !\Delta_2 \vdash (\lambda !x.P)(!N)} \ \mathsf{app}$$

and the reduction rule is:

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

Very similar to the previous case, but rather than the contraction case, we must apply the bang case of the Lemma 6.8.

- r is app:

$$\dfrac{\dfrac{\begin{array}{c} d_1 \\ \vdots \\ \Psi_1, \#\Delta_1, x \vdash P_1 \end{array}}{\Psi_1, \#\Delta_1 \vdash \lambda x.P_1} \qquad \begin{array}{c} d_2 \\ \vdots \\ \Psi_2, \#\Delta_2 \vdash P_2 \end{array}}{\Psi_1, \Psi_2, \#\Delta_1 \cup \#\Delta_2 \vdash \lambda x.P_1(P_2)}\,\mathsf{app} \quad .$$

  and and the reduction rule is:

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

  we have the thesis by means of the linear case of Lemma 6.8 (we apply the lemma to the judgements obtained from derivations d_1 and d_2):

$$\Psi_1, \Psi_2, \#\Delta_1 \cup \#\Delta_2 \vdash P\{N/x\}.$$

- r is app:

$$\dfrac{\dfrac{\begin{array}{c} d_1 \\ \vdots \\ \Gamma, x_1, \ldots, x_n \vdash P \end{array}}{\Gamma \vdash \lambda\langle x_1, \ldots, x_n\rangle.P}\,\multimap_1 \qquad \begin{array}{c} d_2 \\ \vdots \\ !\Delta, r_1, \ldots, r_n \vdash \langle r_1, \ldots, r_n\rangle \end{array}}{\Gamma, !\Delta \vdash \lambda\langle x_1, \ldots, x_n\rangle.P(\langle r_1, \ldots, r_n\rangle)}\,\mathsf{app}$$

  and the reduction rule is:

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

  We obtain the statement $\Gamma, !\Delta, r_1, \ldots, r_n \vdash P\{r_1/x_1, \ldots, r_n/x_n\}$ by means of the quantum case of Lemma 6.8 to the the judgements obtained from derivations d_1 and d_2.

- r is app:

$$\dfrac{\begin{array}{c} d_1 \\ \vdots \\ !\Delta_1 \vdash U \end{array} \qquad \begin{array}{c} d_2 \\ \vdots \\ !\Delta_2, r_1, \ldots, r_n \vdash \langle r_1, \ldots, r_n\rangle \end{array}}{!\Delta_1, !\Delta_2, r_1, \ldots, r_n \vdash U(\langle r_1, \ldots, r_n\rangle)}\,\mathsf{app}$$

  and the reduction rule is:

$$[\mathcal{Q}, \mathcal{QV}, U(\langle r_1, \ldots, r_n\rangle)] \rightarrow_{\mathsf{Uq}} [\mathbf{U}_{\langle\langle r_1, \ldots, r_n\rangle\rangle}\mathcal{Q}, \mathcal{QV}, \langle r_1, \ldots, r_n\rangle].$$

  We obtain the result from derivation d_2, by several application of weakening rule:

$$\dfrac{\begin{array}{c} d_2 \\ \vdots \\ !\Delta_2, r_1, \ldots, r_n \vdash \langle r_1, \ldots, r_n\rangle \end{array}}{!\Delta_1, !\Delta_2, r_1, \ldots, r_n \vdash \langle r_1, \ldots, r_n\rangle}\,\mathsf{w}$$

- r is app:

$$\dfrac{\begin{matrix} d_1\\ \vdots \\ \Gamma_1, \#\Delta \vdash L\end{matrix} \qquad \dfrac{\dfrac{\begin{matrix} d_2\\ \vdots\\ \Gamma_2', \#\Delta, x_1,\ldots,x_n \vdash P\end{matrix}}{\Gamma_2', \#\Delta \vdash (\lambda\langle x_1,\ldots,x_n\rangle.P)}\multimap_1 \qquad \begin{matrix} d_3\\ \vdots\\ \Gamma_2'', \#\Delta\vdash N\end{matrix}}{\Gamma_2, \#\Delta \vdash (\lambda\langle x_1,\ldots,x_n\rangle.P)(N)}\,\mathsf{app}}{\Gamma_1,\Gamma_2,\#\Delta \vdash L((\lambda\langle x_1,\ldots,x_n\rangle.P)(N))}\,\mathsf{app}$$

(where $\Gamma_2 \equiv \Gamma_2', \Gamma_2''$), and the reduction rule is:

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

.
We have:

$$\dfrac{\dfrac{\dfrac{\begin{matrix} d_1\\ \vdots \\ \Gamma_1, \#\Delta \vdash L\end{matrix} \qquad \begin{matrix} d_2\\ \vdots\\ \Gamma_2', \#\Delta, x_1,\ldots,x_n \vdash P\end{matrix}}{\Gamma_1, \Gamma_2', \#\Delta, x_1,\ldots,x_n \vdash (LP)}\,\mathsf{app}}{\Gamma_1, \Gamma_2', \#\Delta \vdash \lambda\langle x_1,\ldots,x_n\rangle.(LP)}\multimap_1 \qquad \begin{matrix} d_3\\ \vdots\\ \Gamma_2'', \#\Delta\vdash N\end{matrix}}{\Gamma_1,\Gamma_2,\#\Delta \vdash (\lambda\langle x_1,\ldots,x_n\rangle.(LP))(N)}\,\mathsf{app}$$

- r is app:

$$\dfrac{\dfrac{\dfrac{\begin{matrix} d_1\\ \vdots\\ \Gamma_1', \#\Delta, x_1,\ldots,x_n \vdash P\end{matrix}}{\Gamma_1', \#\Delta \vdash (\lambda\langle x_1,\ldots,x_n\rangle.P)}\multimap_1 \qquad \begin{matrix} d_2\\ \vdots\\ \Gamma_1'', \#\Delta\vdash N\end{matrix}}{\Gamma_1, \#\Delta \vdash (\lambda\langle x_1,\ldots,x_n\rangle.P)(N)}\,\mathsf{app} \qquad \begin{matrix} d_3\\ \vdots\\ \Gamma_2, \#\Delta\vdash L\end{matrix}}{\Gamma_1,\Gamma_2,\#\Delta \vdash ((\lambda\langle x_1,\ldots,x_n\rangle.P)(N))L}\,\mathsf{app}$$

(where $\Gamma_1 \equiv \Gamma_1', \Gamma_1''$), and and the reduction rule is:

$$[\mathcal{Q}, \mathcal{QV}, ((\lambda\langle x_1,\ldots,x_n\rangle.P)(N))L] \rightarrow_{\mathsf{r.cm}} [\mathcal{Q}, \mathcal{QV}, (\lambda\langle x_1,\ldots,x_n\rangle.(PL))(N)]$$

We have:

$$
\dfrac{
 \dfrac{
  \dfrac{\begin{array}{c} d_1 \\ \vdots \\ \Gamma_1', \#\Delta, x_1, \ldots, x_n \vdash P\end{array} \qquad \begin{array}{c} d_3 \\ \vdots \\ \Gamma_2, \#\Delta \vdash L\end{array}}{\Gamma_1', \Gamma_2, \#\Delta, x_1,\ldots,x_n \vdash (PL)}\,\mathsf{app}
 }{\Gamma_1', \Gamma_3, \#\Delta \vdash \lambda\langle x_1,\ldots,x_n\rangle.(PL)}\,\multimap_1
 \qquad \begin{array}{c} d_2\\ \vdots \\ \Gamma_1'', \#\Delta \vdash N\end{array}
}{\Gamma_1,\Gamma_2,\#\Delta \vdash (\lambda\langle x_1,\ldots,x_n\rangle.(LP))(N)}\,\mathsf{app}
\quad .
$$

- r is $\multimap_1$:

$$
\dfrac{\begin{array}{c} d \\ \vdots \\ \Gamma, x_1, \ldots, x_n \vdash P\end{array}}{\Gamma \vdash \lambda\langle x_1,\ldots,x_n\rangle.P}\,\multimap_1
$$

  and the reduction rule is:

$$
\dfrac{[\mathcal{Q}, \mathcal{QV}, P] \to_\alpha [\mathcal{Q}', \mathcal{QV}', P']}{[\mathcal{Q}, \mathcal{QV}, \lambda\langle x_1,\ldots,x_n\rangle.P] \to_\alpha [\mathcal{Q}', \mathcal{QV}', \lambda\langle x_1,\ldots,x_n\rangle.P']}\,\mathsf{in}.\lambda_2
$$

  Applying the induction hypothesis to d, there are an *m–fun* m and a derivation d' such that:

$$
\dfrac{\begin{array}{c} d' \\ \vdots \\ m[\Gamma], \mathcal{QV}' - \mathcal{QV}, x_1, \ldots, x_n \vdash P'\end{array}}{m[\Gamma], \mathcal{QV}' - \mathcal{QV} \vdash \lambda\langle x_1,\ldots,x_n\rangle.P}\,\multimap_1
$$

- r is $\multimap_2$:

$$
\dfrac{\begin{array}{c} d \\ \vdots \\ \Gamma, x \vdash P\end{array}}{\Gamma \vdash \lambda x.P}\,\multimap_2
$$

  and the reduction rule is:

$$
\dfrac{[\mathcal{Q}, \mathcal{QV}, P] \to_\alpha [\mathcal{Q}', \mathcal{QV}', P']}{[\mathcal{Q}, \mathcal{QV}, \lambda x.P] \to_\alpha [\mathcal{Q}', \mathcal{QV}', \lambda x.P']}\,\mathsf{in}.\lambda_2
$$

  Applying the induction hypothesis to d, there are an *m–fun* m and a derivation d' such that:

$$
\dfrac{\begin{array}{c} d' \\ \vdots \\ m[\Gamma], \mathcal{QV}' - \mathcal{QV}, x \vdash P'\end{array}}{m[\Gamma], \mathcal{QV}' - \mathcal{QV} \vdash \lambda x.P}\,\multimap_2
$$

- r is $\to_{\#}$:

$$\dfrac{\begin{array}{c} d \\ \vdots \\ \Gamma, \# x \vdash P \end{array}}{\Gamma \vdash \lambda ! x.P} \to_{\#}$$

and the reduction rule is:

$$\dfrac{[\mathcal{Q}, \mathcal{QV}, P] \to_{\alpha} [\mathcal{Q}', \mathcal{QV}', P']}{[\mathcal{Q}, \mathcal{QV}, \lambda ! x.P] \to_{\alpha} [\mathcal{Q}', \mathcal{QV}', \lambda ! x.P']} \; \mathsf{in}.\lambda_1$$

Applying the induction hypothesis to d, there are an *m–fun* m and a derivation d' such that:

$$\dfrac{\begin{array}{c} d' \\ \vdots \\ m[\Gamma], \mathcal{QV}' - \mathcal{QV}, \# x \vdash P' \end{array}}{m[\Gamma], \mathcal{QV}' - \mathcal{QV} \vdash \lambda ! x.P} \to_{\#} .$$

- r is $\to_{!}$:

$$\dfrac{\begin{array}{c} d \\ \vdots \\ \Gamma, ! x \vdash P \end{array}}{\Gamma \vdash \lambda ! x.P} \to_{!}$$

and the reduction rule is:

$$\dfrac{[\mathcal{Q}, \mathcal{QV}, P] \to_{\alpha} [\mathcal{Q}', \mathcal{QV}', P']}{[\mathcal{Q}, \mathcal{QV}, \lambda ! x.P] \to_{\alpha} [\mathcal{Q}', \mathcal{QV}', \lambda ! x.P']} \; \mathsf{in}.\lambda_1$$

Applying the H.I. to d there are an *m–fun* m and a derivation d'; we have two cases.

1. $$\dfrac{\begin{array}{c} d' \\ \vdots \\ m[\Gamma], \mathcal{QV}' - \mathcal{QV}, ! x \vdash P' \end{array}}{m[\Gamma], \mathcal{QV}' - \mathcal{QV} \vdash \lambda ! x.P} \to_{!}$$

2. $$\dfrac{\begin{array}{c} d' \\ \vdots \\ m[\Gamma], \mathcal{QV}' - \mathcal{QV}, \# x \vdash P' \end{array}}{m[\Gamma], \mathcal{QV}' - \mathcal{QV} \vdash \lambda ! x.P} \to_{\#}$$

- r is tens:

$$\dfrac{\begin{array}{c}d_1\\ \vdots\\ \Psi_1, \#\Delta_1 \vdash P_1\end{array} \quad \begin{array}{c}d_i\\ \vdots\\ \Psi_i, \#\Delta_i \vdash P_i\end{array} \quad \begin{array}{c}d_n\\ \vdots\\ \Psi_n, \#\Delta_n \vdash P_n\end{array}}{\Psi_1, \ldots, \Psi_n, \#\Delta_1 \cup \#\Delta_2 \cup \ldots \cup \#\Delta_k \vdash \langle P_1, \ldots, P_n \rangle}$$

and the reduction rule is

$$\dfrac{[\mathcal{Q}, \mathcal{QV}, P_i] \to_\alpha [\mathcal{Q}', \mathcal{QV}', P_i']}{[\mathcal{Q}, \mathcal{QV}, \langle P_1, \ldots, P_i, \ldots, P_n\rangle] \to_\alpha [\mathcal{Q}', \mathcal{QV}', \langle P_1, \ldots, P_i', \ldots, P_n\rangle]}\ .$$

Applying the induction hypothesis to d_i, there are an *m–fun* m and a derivation d_i' such that

$$\dfrac{\begin{array}{c}d_1\\ \vdots\\ \Psi_1, \#\Delta_1 \vdash P_1\end{array} \quad \begin{array}{c}d_i'\\ \vdots\\ m[\Psi_i], \#\Delta_i, \mathcal{QV}' - \mathcal{QV} \vdash P_i'\end{array} \quad \begin{array}{c}d_n\\ \vdots\\ \Psi_n, \#\Delta_n \vdash P_n\end{array}}{\Psi_1, \ldots, \Psi_{i-1}, \Psi_{i+1}, \ldots, \Psi_n, m[\Psi_i], \#\Delta_1 \cup \#\Delta_2 \cup \ldots \cup \#\Delta_k \vdash \langle P_1, \ldots, P_i', \ldots, P_n\rangle}\ .$$

Note that the derivation is correct. In fact if $!x \in \Psi_i$, then by means of well forming of d $\#x$ can not belongs to any $\#\Delta_j$. So, a possible modification of $!x$ into $\#x$ does not cause any new contraction.

- r is new. We have two case:

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

   and the reduction rule is:

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

   The thesis is obtained by means of q–var:

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

   .
   (Observe that $\mathcal{QV}' - \mathcal{QV} = \{p\}$).
2. $$\dfrac{!\Delta \vdash P}{!\Delta \vdash \mathtt{new}((P))}\ \mathsf{new}$$

   and the reduction rule is:

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

Applying the induction hypothesis to d, there are an *m–fun* m and a derivation d' such that:

$$\dfrac{\begin{array}{c} d' \\ \vdots \\ m[\Gamma], \mathcal{QV}' - \mathcal{QV} \vdash P' \end{array}}{m[\Gamma], \mathcal{QV}' - \mathcal{QV} \vdash \mathsf{new}(P')}\ \mathsf{new} \, .$$

In the rest of the paper we will restrict our attention to well–formed configurations, that (in order to simplify the writing) we continue to call simply configurations. We conclude this section with the notion of *computation* and *normal form*.

Definition 6.9 (Normal forms). *A configuration $C \equiv [\mathcal{Q}, \mathcal{QV}, M]$ is said to be in normal form iff there is no D such that $C \to D$. Let us denote with* NF *the set of configurations in normal form.*

We define a computation as a suitable *finite* sequence of configurations:

Definition 6.10 (Computations). *If C_1 is any configuration, a* computation *of length $m \in \mathbb{N}$ starting with C_1 is a sequence of configurations $\{C_i\}_{1\leq i\leq m}$ such that for all $1 \leq i < m$, $C_i \to C_{i+1}$ and $C_m \in$* NF.

As we will see, the limitation to finite sequences of computation is perfectly reasonable. Indeed, we will prove (as a byproduct of polytime soundness, Section 6.7) that SQ is strongly normalizing.

In the concrete realization of quantum algorithms, the initial quantum register is empty (it will be created during the computation). With this hypothesis, configurations in a computation can be proved to have a certain regular shape:

Proposition 6.11. *Let $\{[\mathcal{Q}_i, \mathcal{QV}_i, M_i]\}_{1\leq i\leq m}$ be a computation, such that $\mathbf{Q}(M_1) = \emptyset$. Then for every $i \leq m$ we have $\mathcal{QV}_i = \mathbf{Q}(M_i)$.*

Proof. The results holds for any sequence $\{[\mathcal{Q}_i, \mathcal{QV}_i, M_i]\}_{1\leq i\leq m}$ of configurations whenever $\mathbf{Q}(M_1) = \emptyset$. This stronger statement can be proved by induction on m by making use of Theorem 6.4. Indeed, if $m > 1$ (the base case is trivial):

$$\begin{aligned} \mathcal{QV}_m &= (\mathcal{QV}_m - \mathcal{QV}_{m-1}) \cup \mathcal{QV}_{m-1} = (\mathbf{Q}(M_m) - \mathbf{Q}(M_{m-1})) \cup \mathcal{QV}_{m-1} \\ &= (\mathbf{Q}(M_m) - \mathbf{Q}(M_{m-1})) \cup \mathbf{Q}(M_{m-1}) = \mathbf{Q}(M_m) \end{aligned}$$

This concludes the proof.

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

6.4 Confluence and standardization

SQ enjoys the confluence exactly as Q, and in fact the proof given for Q in Section 4.3.3 is also a proof of confluence for SQ. It is possible to obtain confluence of SQ also as a corollary of Q confluence (as for simply typed λ–calculus, where confluence is a direct consequence of confluence of pure λ–calculus).

Lemma 6.12. *If C is a configuration of* SQ *, then C is also a configuration of* Q*.*

Proof. By induction on well forming rules.

Theorem 6.13 (confluence). *Let C, D, E be configurations with $C \stackrel{*}{\longrightarrow} D$ and $C \stackrel{*}{\longrightarrow} E$. Then, there is a configuration F with $D \stackrel{*}{\longrightarrow} F$ and $E \stackrel{*}{\longrightarrow} F$.*

Proof. If $C \to^* D$ and $C \to^* E$ in SQ, then $C \to^* D$ and $C \to^* E$ in Q. By Subject Reduction, D and E are configuration of SQ and moreover, by Theorem 4.20 (confluence theorem for Q) there exists a configuration F in Q such that $D \to^* F$ and $E \to^* F$. By Subject Reduction, F is also a configuration of SQ.

Moreover, as a consequence of having adopted the so-called *surface reduction*, (i.e. it is not possible to reduce inside a subterm in the form $!M$) it is not possible to erase a diverging term (see also [91]). Therefore, exactly as for Q (see Theorem 4.23), it is possible to show that:

Theorem 6.14. *A configuration C is strongly normalizing iff C is weakly normalizing.*

The theorem is a trivial consequence of the corresponding property of Q, Lemma 6.12, and Subject Reduction. In any case such a result will be superseded by Theorem 6.32: in Section 6.7, we prove that *any* configuration is in fact strongly normalizing.

Another interesting property, that SQ inherits from Q is *quantum standardization*. The definitions of classes NCL, EQT and the notion of *standard computation* CNQ are the same of Section 4.5.

Here we will recall only the main theorem:

Theorem 6.15 (Quantum Standardization). *For every computation $\{C_i\}_{1\leq i\leq m}$ there is a* CNQ *computation $\{D_i\}_{1\leq i\leq n}$ such that $C_1 \equiv D_1$ and $C_m \equiv D_n$.*

The proof of Theorem 6.15 proceeds by first showing that NCL is closed under $\to_{\mathscr{Q}}$ and that EQT is closed under $\to_{\mathsf{new}}$, as for Q.

6.5 Encoding Classical Data Structures

It is not possible to use the encoding given in Section 5.2 for Q. Classically, SQ has the expressive power of soft linear logic and we need to control the duplication of resources. In this section we will illustrate some encodings of usual data structures such as natural numbers, binary strings and lists. Notice that some of the encodings we are going to present are non-standard: they are not the usual Church-style encodings, which are not necessarily available in a restricted setting like the one we are considering here. The results in this Section will be essential in Section 6.8.

6.5.1 Natural Numbers

We need to stay as abstract as possible here: there will be many distinct terms representing the same natural number. Given a natural number $n \in \mathbb{N}$ and a term M, the class of *n-banged forms of M* is defined by induction on n:

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

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

Please notice that $!^n M$ is an n-banged form of M for every $n \in \mathbb{N}$ and for every term M.

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

$$ML \to_{\mathscr{N}} \lambda z.\underbrace{N(N(N(\ldots(N}_{m \text{ times}}z)\ldots))).$$

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

$$ML \to_{\mathscr{N}}^{*} P$$

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

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

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

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

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

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

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

Lemma 6.16. *Let $id : \mathbb{N} \to \mathbb{N}$ be the identity function. For every natural number n, there is a term M_{id}^n which $(n,1)$-represents id. Moreover, for every $m \in \mathbb{N}$ and for every term N, $M_{id}^n !^{n+m} N \to_{\mathscr{N}}^{*} !^m N$*

Proof. By induction on n:

- $M_{id}^0 = \lambda x.x$. Indeed, for every N 1-representing $m \in \mathbb{N}$ and for every 0-banged form L of N:
$$M_{id}^0 L = M_{id}^0 N = (\lambda x.x)N \to_{\mathscr{N}} N.$$
- $M_{id}^{n+1} = \lambda !x.M_{id}^n x$. Indeed, for every N 1-representing $m \in \mathbb{N}$ and for every $n+1$-banged form L of N:
$$M_{id}^{n+1} L = M_{id}^{n+1} !P = (\lambda !x.M_{id}^n x)!P \to_{\mathscr{N}} M_{id}^n P \to_{\mathscr{N}}^{*} M_{id}^n Q \to_{\mathscr{N}}^{*} L$$
where Q is an n-banged form of N and L 1-represents m.

This concludes the proof.

SQ can compute any polynomial, in a strong sense:

Proposition 6.17. *For any polynomial with natural coefficients $p : \mathbb{N} \to \mathbb{N}$ of degree n, there is a term M_p that $(2n+1, 2n+1)$-represents p.*

Proof. Any polynomial can be written as an *Horner polynomial*, which is either

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

So, proving that the thesis holds for Horner's polynomials suffices. We go by induction, following the above recursion schema:

- Any constant polynomial $p : \mathbb{N} \to \mathbb{N}$ in the form $x \mapsto k$ is $(1, 1)$-representable. Just take $M_p = \lambda!x.\lceil k \rceil$. Indeed:
$$M_p!N \to \lceil k \rceil.$$
- Suppose $r : \mathbb{N} \to \mathbb{N}$ is a polynomial of degree n which can be $(2n + 1, 2n + 1)$-represented by M_r. Suppose $k \in \mathbb{N}$ and let $p : \mathbb{N} \to \mathbb{N}$ be the polynomial $x \mapsto k + x \cdot r(x)$. Consider the term
$$M_p = \lambda!x.\lambda!y.\lambda z.(\lceil k \rceil^{2n+2} y)((M_{id}^{2n+2} x)((\lambda!w.\lambda!u.!(M_r wu))xy)z)$$
Let now N be a term 1-representing a natural number i, L be any term, $!P$ be any $(2n + 3)$-banged form of N and $!Q$ be any $(2n + 3)$-banged form of L. Then
$$\begin{aligned}
M_p!P!Q &\to^*_{\mathscr{N}} \lambda z.(\lceil k \rceil^{2n+2} Q)((M_{id}^{2n+2} P)((\lambda!w.\lambda!u.!(M_r wu))PQ)z)\\
&\to^*_{\mathscr{N}} \lambda z.(\lceil k \rceil^{2n+2} Q)((M_{id}^{2n+2} P)((\lambda!w.\lambda!u.!(M_r wu))!R!S)z)\\
&\to^*_{\mathscr{N}} \lambda z.(\lceil k \rceil^{2n+2} !S)((M_{id}^{2n+2} !R)!(M_r RS)z)\\
&\to^*_{\mathscr{N}} \lambda z.(\lambda z.\underbrace{L(L(L(\ldots(L}_{k\text{ times}} z)\ldots)))\\
&\qquad (V!(M_r RS)z)\\
&\to^*_{\mathscr{N}} \lambda z.(\lambda z.\underbrace{L(L(L(\ldots(L}_{k\text{ times}} z)\ldots)))\\
&\qquad (V!(M_r TU)z)\\
&\to^*_{\mathscr{N}} \lambda z.(\lambda z.\underbrace{L(L(L(\ldots(L}_{k\text{ times}} z)\ldots)))\\
&\qquad (\lambda z.\underbrace{(M_r TU)(\ldots((M_r TU)}_{i\text{ times}} z)\ldots)))\\
&\to^*_{\mathscr{N}} \lambda z.(\lambda z.\ \underbrace{L(L(L(\ldots(L}_{k+ir(i)\text{ times}}\ z)\ldots)))z\\
&\to_{\mathscr{N}} (\lambda z.\ \underbrace{L(L(L(\ldots(L}_{k+ir(i)\text{ times}}\ z)\ldots)))
\end{aligned}$$

where V 1-represents i, $!R$ is a $(2n+2)$-banged form of N, $!S$ is a $(2n+2)$-banged form of L, T is a $(2n + 1)$-banged form of N and U is a $(2n + 1)$-banged form of L. This concludes the proof.

6.5.2 Strings

Other than natural numbers, we are interested in representing strings in an arbitrary (finite) alphabet. Given any string $s = b_1 \ldots b_n \in \Sigma^*$ (where Σ is a finite alphabet), the term $\lceil s \rceil^{\Sigma}$ is the following:

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

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

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

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

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

6.5.3 Lists

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

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

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 as follows on lists:

$$\begin{aligned}
\mathsf{cons}M[M_1, \ldots, M_n] &\rightarrow^2_{\mathscr{N}} [M, M_1, \ldots, M_n] \\
\mathsf{sel}[]!N!L &\rightarrow^3_{\mathscr{N}} L \\
\mathsf{sel}[M, M_1, \ldots, M_n]!N!L &\rightarrow^3_{\mathscr{N}} NM[M_1, \ldots, M_n]
\end{aligned}$$

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

$$\begin{aligned}
\mathsf{append}_n[N_1, \ldots, N_m]M_1, \ldots, M_n &\rightarrow^*_{\mathscr{N}} [M_1, \ldots, M_n, N_1, \ldots, N_m] \\
\forall m \leq n.\mathsf{extract}_n M[N_1, \ldots, N_m] &\rightarrow^*_{\mathscr{N}} M[]N_m N_{m-1} \ldots N_1 \\
\forall m \geq n.\mathsf{extract}_n M[N_1, \ldots N_m] &\rightarrow^*_{\mathscr{N}} M\ [N_{n+1} \ldots N_m]N_n N_{n-1} \ldots N_1
\end{aligned}$$

Terms append_n can be built by induction on n:

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

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

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

Indeed:

$$\begin{aligned}\forall m.\mathsf{extract}_0 M[N_1,\ldots N_m] &\to^2_{\mathscr{N}} M[N_1,\ldots,N_m]\\ \forall n.\mathsf{extract}_{n+1} M[] &\to^5_{\mathscr{N}} M[]\\ \forall m < n.\mathsf{extract}_{n+1} M[N, N_1 \ldots N_m] &\to^7_{\mathscr{N}} \mathsf{extract}_n M[N_1,\ldots,N_m]N\\ &\to^*_{\mathscr{N}} M[]N_m \ldots N_1 N\\ \forall m \geq n.\mathsf{extract}_{n+1} M[N, N_1 \ldots N_m] &\to^7_{\mathscr{N}} \mathsf{extract}_n M[N_1,\ldots,N_m]N\\ &\to^*_{\mathscr{N}} M[N_{n+1}\ldots N_m]N_n \ldots N_1 N\end{aligned}$$

6.6 Representing Decision Problems

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

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

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

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

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

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

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

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

Definition 6.18. *Three classes of languages in the alphabet* $\{0,1\}$ *are defined below:*

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

The purpose of the following two sections is precisely proving that ESQ, BSQ and ZSQ coincide with the quantum complexity classes EQP, BQP ad ZQP, respectively.

6.7 Polytime Soundness

Following the approach proposed by Girard in [50] and subsequently developed in [10, 12, 63] we show that SQ is intrinsically a poly time calculus. This allows to show that decision problems which can be represented in SQ lie in certain polytime (quantum) complexity classes.

In order to simplify the treatment we will consider reduction between terms rather than between configurations. If $[\mathcal{Q}, \mathcal{QV}, M] \xrightarrow{\mathscr{L}}_{\mathscr{K}} [\mathcal{Q}', \mathcal{QV}', M']$, then we will simply write $M \xrightarrow{\mathscr{L}}_{\mathscr{K}} M'$. This is a good definition, since M' only depends on M (and does not depend on $\mathcal{Q}$ nor on $\mathcal{QV}$).

In this section we assume that *all the involved terms are well formed*.

We start with some definitions. The *size* of a term is defined in a standard way as:

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

Lemma 6.19. *For every term M and for every variable x, $\mathsf{NFO}(x, M) \leq |M|$*

Proof. By induction on M.

We remember to the reader the definition of two subsets of $\mathscr{L}$, namely $\mathscr{K} = \{\mathsf{r.cm}, \mathsf{l.cm}\}$ and $\mathscr{N} = \mathscr{L} - \mathscr{K}$ (defined in Section 4.3.3).

Lemma 6.20. *If $M \xrightarrow{n}_{\mathscr{K}} N$, then (i) $|M| = |N|$; (ii) $n \leq |M|^2$.*

Proof. (i) By induction on the derivation of $M \to_{\mathscr{K}} N$.

Observe that $|L((\lambda\pi.M_1)M_2)| = |(\lambda\pi.LM_1)M_2|$ and $|((\lambda\pi.M_1)M_2)L| = |(\lambda\pi.M_1L)M_2|$; these are base cases in which $L((\lambda\pi.M_1)M_2 \to_{\mathsf{l.cm}} (\lambda\pi.LM_1)M_2$ or $((\lambda\pi.M_1)M_2)L \to_{\mathsf{r.cm}} (\lambda\pi.M_1L)M_2$. We have context closures as inductive steps. For example, let M be LP and let be

$$\frac{P \to_{\mathscr{K}} Q}{LP \to_{\mathscr{K}} LQ}\,\mathsf{l.a}$$

the last rule in the derivation. By induction hypothesis we have $|P| = |Q|$, and $|M| = |LP| = |L| + |P| + 1 = |L| + |Q| + 1 = |LQ|$. The other cases are very similar to the previous one.

(ii) Define the abstraction size $|M|_\lambda$ of M as the sum over all subterms of M in the form $\lambda\pi.L$, of $|L|$. Clearly $|M|_\lambda \le |M|^2$. Moreover, $n \le |M|_\lambda$ because

$$|L((\lambda\pi.M_1)M_2)|_\lambda < |(\lambda\pi.LM_1)M_2|_\lambda$$
$$|((\lambda\pi.M_1)M_2)L|_\lambda < |(\lambda\pi.M_1L)M_2|_\lambda$$

In other words, $|M|_\lambda$ always increases along commuting reduction.
This concludes the proof.

In order to prove polytime soundness of the calculus we need to assign to each term M the following degrees:

Definition 6.21 (Box-depth, Duplicability-Factor, Weights).

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

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

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

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

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

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

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

We need some lemmas in order to relate duplicability factor, size and weights.

Lemma 6.22. *For every term* M, $\mathsf{D}(M) \leq |M|$.

Proof. By induction on M:
- M is a variable or a constant or a quantum variable; then $\mathsf{D}(M) = 1 = |M|$.
- M is of the form $\lambda x.N$. Then, by Lemma 6.19:

$$\begin{aligned}\mathsf{D}(\lambda x.N) &= \max\{\mathsf{D}(N), \mathsf{NFO}(x,N)\}\\ &\stackrel{IH}{\leq} \max\{|N|, \mathsf{NFO}(x,N)\}\\ &\leq \max\{|N|, |N|\}\\ &= |N| \leq |N| + 1 = |M|.\end{aligned}$$

- M is of the form $\lambda !x.N$ or $\lambda\pi.N$: very similar to the previous case.
- M is PQ. Then:

$$\begin{aligned}\mathsf{D}(PQ) &= \max\{\mathsf{D}(P), \mathsf{D}(Q)\} \stackrel{IH}{\leq} \max\{|P|, |Q|\}\\ &\leq \max\{|P| + |Q| + 1, |P| + |Q| + 1\}\\ &= |P| + |Q| + 1 = |PQ|.\end{aligned}$$

- M is $\mathsf{new}(N)$:

$$\mathsf{D}(\mathsf{new}(N)) = \mathsf{D}(N) \stackrel{IH}{\leq} |N| < |N| + 1 = |M|.$$

- M is $!N$, then

$$\mathsf{D}(!N) = \mathsf{D}(N) \stackrel{IH}{\leq} |N| < |N| + 1 = |M|.$$

- M is $\langle N_1, \ldots, N_k\rangle$ and for all N_i, $\quad i = 1 \ldots k$, we have $\mathsf{D}(N_i) \leq |N_i|$ by induction hypothesis; then

$$\begin{aligned}\mathsf{D}(M) &= \max\{\mathsf{D}(N_1), \ldots, \mathsf{D}(N_k)\} \stackrel{IH}{\leq} \max\{|N_1|, \ldots, |N_k|\}\\ &< |N_1| + \ldots + |N_k| + 1 = |M|.\end{aligned}$$

This concludes the proof.

The number of free occurrences of a variable cannot increase too much during reduction:

Lemma 6.23. *If* $P \to_{\mathscr{L}} Q$ *then* $\max\{\mathsf{NFO}(x,P), \mathrm{d}P\} \geq \mathsf{NFO}(x,Q)$.

Proof. The proof proceeds by proving the following facts:
1. if $\triangleright \Gamma, x \vdash P$ and $P \to_{\mathscr{L}} Q$ then $\mathsf{NFO}(x,P) \geq \mathsf{NFO}(x,Q)$;
2. if $\triangleright \Gamma, \#x \vdash P$ and $P \to_{\mathscr{L}} Q$ then $\mathsf{NFO}(x,P) \geq \mathsf{NFO}(x,Q)$;
3. if $\triangleright \Gamma, !x \vdash P$ and $P \to_{\mathscr{L}} Q$ then $\max\{\mathsf{NFO}(x,P), \mathrm{d}P\} \geq \mathsf{NFO}(x,Q)$.

The Lemma is therefore a trivial consequence of the above facts. The proofs of 1., 2. and 3. are simple inductions on the derivation of $P \to_{\mathscr{L}} Q$. We will show here only some interesting cases.

1. We distinguish two cases:
   - If the last rule is a base rule, we have several sub-cases. If the reduction rule is $(\lambda!y.L)!M \rightarrow_{\mathsf{c}.\beta} L\{M/y\}$, please observe that $\mathsf{NFO}(x, !M) = 0$ and conclude. If the reduction rule is $(\lambda y.L)M \rightarrow_{\mathsf{l}.\beta} L\{M/y\}$, we have only two possibilities: either $\mathsf{NFO}(x, M) = 0$ and $\mathsf{NFO}(x, L) = 1$ or $\mathsf{NFO}(x, M) = 1$ and $\mathsf{NFO}(x, L) = 0$; in both cases the conclusion is immediate. The other sub-cases are easier.
   - If the last reduction rule is a context closure rules, the result follows easily by applying the induction hypothesis. For example, if the closure rule is

     $$\frac{M \rightarrow N}{ML \rightarrow NL}$$

     we have two sub-cases: either $\mathsf{NFO}(x, P_1) = 0$ and $\mathsf{NFO}(x, P_2) = 1$ or $\mathsf{NFO}(x, P_1) = 1$ and $\mathsf{NFO}(x, P_2) = 0$. In the first sub-case the thesis follow immediately. In the second sub-case the result follows by applying the induction hypothesis $\mathsf{NFO}(x, M) \geq \mathsf{NFO}(x, L)$.
2. We distinguish two cases:
   - If the last rule is a base rule, we have several sub-cases If the reduction rule is $(\lambda!y.L)!M \rightarrow_{\mathsf{c}.\beta} L\{M/y\}$, please observe that $\mathsf{NFO}(x, !M) = 0$ and conclude. If the reduction rule is $(\lambda y.L)M \rightarrow_{\mathsf{l}.\beta} L\{M/y\}$, simply observe that y must occur exactly once in L and therefore $\mathsf{NFO}(x, (\lambda y.L)M) = \mathsf{NFO}(x, M) + \mathsf{NFO}(x, L) = \mathsf{NFO}(x, L\{M/y\})$. All the other base cases can be easily proved.
   - If the last reduction rule is a context closure rule, the result follows easily by applying the induction hypothesis. For example if the reduction rule is

     $$\frac{M \rightarrow N}{ML \rightarrow NL}$$

     we have two sub-cases: (i) $\mathsf{NFO}(x, M) = 0$; in this case the thesis follow immediately; (ii) $\mathsf{NFO}(x, M) \neq 0$; the result follows by applying the induction hypothesis $\mathsf{NFO}(x, M) \geq \mathsf{NFO}(x, L)$.
3. The proof remains simple but it slightly more delicate, because we must consider the phenomenon of duplication.
   - If the last rule is a base rule: we have several cases. If the reduction rule is $(\lambda!y.L)!M \rightarrow_{\mathsf{c}.\beta} L\{M/y\}$, please observe that differently from the previous facts, we have two possibilities: if $\mathsf{NFO}(x, !M) = 0$ we conclude; otherwise, if $\mathsf{NFO}(x, !M) \neq 0$ we must have that $\mathsf{NFO}(x, !M) = 1$ and $\mathsf{NFO}(x, L) = 0$. Consequently

     $$\begin{aligned}
     \mathsf{max}\{\mathsf{NFO}(x, P), \mathrm{d}P\} &= \mathrm{d}P = \mathsf{max}\{\mathrm{d}\lambda!y.L), \mathrm{d}!M\} \\
     &\geq \mathrm{d}\lambda!y.L \geq \mathsf{max}\{D(L), \mathsf{NFO}(y, L)\} \\
     &\geq \mathsf{NFO}(y, L) \\
     &= \mathsf{NFO}(x, L) + \mathsf{NFO}(y, L) \cdot \mathsf{NFO}(x, M) \\
     &= \mathsf{NFO}(x, L\{M/y\}).
     \end{aligned}$$

     If the reduction rule is $(\lambda y.L)M \rightarrow_{\mathsf{l}.\beta} L\{M/y\}$, simply observe that y must occur exactly once in L and therefore

     $$\mathsf{NFO}(x, (\lambda y.L)M) = \mathsf{NFO}(x, M) + \mathsf{NFO}(x, L) = \mathsf{NFO}(x, L\{M/y\}).$$

All the other base case are easily proved.

- if the last reduction rule is a context closure rules, the result follows easily by applying the induction hypothesis. For example if the reduction rule is

$$\frac{M \to N}{ML \to NL}$$

we have two cases: (i) $\mathsf{NFO}(x, M) = 0$; in this case the thesis follows immediately; (ii) $\mathsf{NFO}(x, M) = 1$ and $\mathsf{NFO}(x, L) = 0$ the result follows by applying the induction hypothesis $\mathsf{max}\{\mathsf{NFO}(x, M), \mathrm{d}M\} \geq \mathsf{NFO}(x, N)$:

$$\begin{aligned}\mathsf{max}\{\mathsf{NFO}(x, ML), \mathrm{d}ML\} &= \mathsf{max}\{\mathsf{NFO}(x, M) + \mathsf{NFO}(x, L), \mathrm{d}M, \mathrm{d}L\}\\ &= \mathsf{max}\{\mathsf{NFO}(x, M), \mathrm{d}M, \mathrm{d}L\}\\ &\geq \mathsf{max}\{\mathsf{NFO}(x, M), \mathrm{d}M\}\\ &\geq \mathsf{NFO}(x, N) = \mathsf{NFO}(x, NL).\end{aligned}$$

This concludes the proof.

Lemma 6.24. *For all terms P and Q,* $\mathsf{D}(P\{Q/x\}) \leq max\{\mathsf{D}(P), \mathsf{D}(Q)\}$.

Proof. By induction on the term P.

Thanks to the well forming rules it is possible to show that $\mathsf{D}(\cdot)$ is non-increasing wrt reduction:

Lemma 6.25. *(i) If $M \to_{\mathscr{K}} N$ then* $\mathsf{D}(M) = \mathsf{D}(N)$*;*
(ii) If $M \to_{\mathscr{N}} N$ then $\mathsf{D}(M) \geq \mathsf{D}(N)$.

Proof. (i) By induction on the derivation of $\to_{\mathscr{K}}$. For the base cases, if $M \to_{\mathsf{l.cm}} N$, M is of the form $L((\lambda\pi.M_1)M_2)$ and N is $(\lambda\pi.LM_1)M_2$. Observe that $\mathsf{l.cm}$ has a side condition on variables, and so $\mathsf{NFO}(x_i, LM_1) = \mathsf{NFO}(x_i, M_1)$ for every i. We have:

$$\begin{aligned}\mathsf{D}(M) &= \max\{\mathsf{D}(L), \mathsf{D}((\lambda\pi.M_1)M_2)\}\\ &= \max\{\mathsf{D}(L), \mathsf{D}(\lambda\pi.M_1), \mathsf{D}(N)\}\\ &= \max\{\mathsf{D}(L), \mathsf{D}(M_1), \mathsf{NFO}(x_1, M_1), \ldots, \mathsf{NFO}(x_n, M_1), \mathsf{D}(M_2)\}\\ &= \max\{\mathsf{D}(L), \mathsf{D}(M_1), \mathsf{NFO}(x_i, LM_1), \ldots, \mathsf{NFO}(x_n, LM_1), \mathsf{D}(M_2)\}\\ &= \max\{\mathsf{D}(LM_1), \mathsf{NFO}(x_i, LM_1), \ldots, \mathsf{NFO}(x_n, LM_1), \mathsf{D}(M_2)\}\\ &= \max\{\mathsf{D}(\lambda\pi.LM_1), \mathsf{D}(M_2)\}\\ &= \mathsf{D}((\lambda\pi.LM_1)M_2).\end{aligned}$$

We have context closures as inductive steps. For example, let M be M_1M_2 and let

$$\frac{M_1 \to_{\mathscr{K}} M_1'}{M_1M_2 \to_{\mathscr{K}} M_1'M_2}\ \mathsf{l.a}$$

be the last rule instance in the derivation. Then:

$$\begin{aligned}\mathsf{D}(M) &= \max\{\mathsf{D}(M_1), \mathsf{D}(M_2)\} \stackrel{IH}{=} \max\{\mathsf{D}(M_1'), \mathsf{D}(M_2)\}\\ &= \mathsf{D}(M_1'M_2)\end{aligned}$$

The other cases are very similar to the previous one.

(ii) By induction on the derivation of $\to_{\mathscr{N}}$. We prove some cases depending on the last rule in the derivation.

- the reduction rule is
$$\frac{P_1 \to P_1'}{P_1P_2 \to P_1'P_2}$$
Then:
$$\begin{aligned}\mathsf{D}(M) &= \max\{\mathsf{D}(P_1), \mathsf{D}(P_2)\}\\ &\overset{IH}{\geq} \max\{\mathsf{D}(P_1'), \mathsf{D}(P_2)\} = \max\{\mathsf{D}(P_1'P_2)\}\end{aligned}$$
- the reduction rule is
$$\frac{P_2 \to P_2'}{P_1P_2 \to P_1P_2'}$$
The argument is symmetric to the previous one.
- the reduction rule is $(\lambda!x.P)!Q \to_{\mathsf{c}.\beta} P\{Q/x\}$. Then:
$$\begin{aligned}\mathsf{D}((\lambda!x.P)!Q) &= \max\{\mathsf{D}(P), \mathsf{NFO}(x,P), \mathsf{D}(Q)\}\\ &\geq \max\{\mathsf{D}(P), \mathsf{D}(Q)\} \geq \mathsf{D}(P\{Q/x\})\end{aligned}$$
where the last step is justified by Lemma 6.24.
- the reduction rule is $(\lambda x.P)Q \to_{\mathsf{l}.\beta} P\{Q/x\}$. Similar to the previous case.
- the reduction rule is $(\lambda\langle x_1,\dots,x_k\rangle.P)\langle r_1,\dots,r_k\rangle \to_{\mathsf{q}.\beta} P\{r_1/x_1,\dots,r_k/x_k\}$. Again similar to the previous case.
- the reduction rule is $U\langle r_1,\dots,r_k\rangle \to_{\mathsf{Uq}} \langle r_1,\dots,r_k\rangle$; the result follows by definitions.
- the reduction rule is
$$\frac{M_i \to_\alpha M_i'}{\langle M_1,\dots,M_i,\dots,M_k\rangle \to_\alpha \langle M_1,\dots,M_i',\dots,M_k\rangle}\,.$$
By induction hypothesis we have
$$\begin{aligned}\mathsf{D}(\langle M_1,\dots,M_i,\dots,M_k\rangle) &= \max\{\mathsf{D}(M_1),\dots,\mathsf{D}(M_i),\dots,\mathsf{D}(M_k)\}\\ &\overset{IH}{\geq} \max\{\mathsf{D}(M_1),\dots,\mathsf{D}(M_i'),\dots,\mathsf{D}(M_k)\}\\ &= \mathsf{D}(\langle M_1,\dots,M_i',\dots,M_k\rangle)\end{aligned}$$
- the reduction rule is
$$\frac{P \to_\alpha Q}{\mathsf{new}(P) \to_\alpha \mathsf{new}(Q)}\ \mathsf{in.new}$$
Then:
$$\mathsf{D}(\mathsf{new}(P)) = \mathsf{D}(P) \overset{IH}{\geq} \mathsf{D}((Q)) = \mathsf{D}(\mathsf{new}(Q)).$$
- the reduction rule is
$$\frac{P \to_\alpha Q}{\lambda!x.P \to_\alpha \lambda!x.Q}\ \mathsf{in}.\lambda_1$$
Then, by Lemma 6.23

$$\mathsf{D}(\lambda !x.P) = \max\{\mathsf{D}(P), \mathsf{NFO}(x,P)\} \geq \mathsf{NFO}(x,Q)$$

moreover by induction hypothesis

$$\mathsf{D}(\lambda !x.P) = \max\{\mathsf{D}(P), \mathsf{NFO}(x,P)\} \geq \mathsf{D}(P) \geq \mathsf{D}(Q)$$

and therefore

$$\mathsf{D}(\lambda !x.P) = \max\{\mathsf{D}(P), \mathsf{NFO}(x,P)\} \geq \max\{\mathsf{D}(Q), \mathsf{NFO}(x,Q)\} = \mathsf{D}(\lambda !x, Q)$$

- the reduction rule is
$$\frac{P \to_\alpha Q}{\lambda\pi.P \to_\alpha \lambda\pi.Q}\ \mathsf{in}.\lambda_2$$
Observe that $\mathsf{D}(\lambda\pi.P) = \mathsf{D}(P)$. The result follows by induction hypothesis.

This concludes the proof.

It is important to remark that such a property does not hold for non well formed terms. For example let us take $M = \lambda !x.((\lambda !z.zz)!(xxx))$, we have $M \to N$ where $N = \lambda !x.((xxx)(xxx))$, but $\mathsf{D}(M) = 3$ and $\mathsf{D}(N) = 6$. By the way, M is well-formed in Q.

Lemma 6.26. *For every term M, $|M| \leq \mathsf{W}(M)$.*

Proof. By induction on the term M. In some cases, we will use the following fact: for all terms M, for all $n, m \in \mathbb{N}$, if $1 \leq m \leq n$, then $\mathsf{W}_m(M) \leq \mathsf{W}_n(M)$.

- M is a variable, a constant or a quantum variable. Then, $|M| = 1 = \mathsf{W}_0(M) = \mathsf{W}_{\mathsf{D}(M)}(M) = \mathsf{W}(M)$.
- M is $!N$. We can proceed as follows:

$$\begin{aligned}|M| &= |N| + 1 \leq \mathsf{W}(N) + 1\\ &= \mathsf{W}_{\mathsf{D}(N)}(N) + 1 = \mathsf{W}_{\mathsf{D}(N)}(!N)\\ &= \mathsf{W}_{\mathsf{D}(!N)}(!N) = \mathsf{W}_{\mathsf{D}(M)}(M) = \mathsf{W}(M)\end{aligned}$$

- M is $\mathsf{new}(N)$; then

$$\begin{aligned}|\mathsf{new}(N)| &= |N| + 1 \overset{IH}{\leq} \mathsf{W}_{\mathsf{D}(N)}(N) + 1\\ &= \mathsf{W}_{\mathsf{D}(N)}(\mathsf{new}(N)) = \mathsf{W}_{\mathsf{D}(\mathsf{new}N)}(\mathsf{new}N) = \mathsf{W}(\mathsf{new}(N)).\end{aligned}$$

- M is PQ; then

$$\begin{aligned}|M| &= |P| + |Q| + 1 \leq \mathsf{W}(P) + \mathsf{W}(Q) + 1\\ &= \mathsf{W}_{\mathsf{D}(P)}(P) + \mathsf{W}_{\mathsf{D}(Q)}(Q) + 1\\ &\leq \mathsf{W}_{\max\{\mathsf{D}(P),\mathsf{D}(Q)\}}(P) + \mathsf{W}_{\max\{\mathsf{D}(P),\mathsf{D}(Q)\}}(Q) + 1\\ &= \mathsf{W}_{\mathsf{D}(PQ)}(P) + \mathsf{W}_{\mathsf{D}(PQ)}(Q) + 1\\ &= \mathsf{W}_{\mathsf{D}(PQ)}(PQ) = \mathsf{W}(PQ) = \mathsf{W}(M).\end{aligned}$$

- M is $\langle N_1, \ldots, N_k \rangle$; then

$$\begin{aligned} |\langle N_1, \ldots, N_k \rangle| &= |N_1| + \ldots + |N_k| + 1 \\ &\stackrel{IH}{\leq} \mathsf{W}_{\mathsf{D}(N_1)}(N_1) + \ldots + \mathsf{W}_{\mathsf{D}(N_k)}(N_k) + 1 \\ &\leq \mathsf{W}_{\mathsf{D}(M)}(N_1) + \ldots + \mathsf{W}_{\mathsf{D}(M)}(N_k) + 1 \\ &= \mathsf{W}_{\mathsf{D}(M)}(M) = \mathsf{W}(M). \end{aligned}$$

- M is $\lambda x.N$; then

$$|M| = |N| + 1 \stackrel{IH}{\leq} \mathsf{W}_{\mathsf{D}(N)}(N) + 1 \leq \mathsf{W}_{\mathsf{D}(M)}(N) + 1 = \mathsf{W}(M)$$

  where the last inequality holds observing that $\mathsf{D}(M) = max\{\mathsf{D}(N), \mathsf{NFO}(x, N)\}$, so $\mathsf{D}(N) \leq \mathsf{D}(M)$.
- M is $\lambda\pi.N$ or M is $\lambda!x.N$: as in the previous case.

This concludes the proof.

We now need to revisit the substitution lemma:

Lemma 6.27 (Substitution Lemma, Revisited).
- *Linear case. If* $\triangleright \Psi_1, \#\Delta_1, x \vdash M$ *and* $\triangleright \Psi_2, \#\Delta_2 \vdash N$, *with* $\mathsf{var}(\Psi_1) \cap \mathsf{var}(\Psi_2) = \emptyset$, *then for all* $m, n \in \mathbb{N}$, $n \geq m \geq 1$, $\mathsf{W}_m(M\{N/x\}) \leq \mathsf{W}_n(M) + \mathsf{W}_n(N)$;
- *Contraction case. If* $\triangleright \Gamma, \#x \vdash M$ *and* $\triangleright \Delta \vdash N$, $\mathsf{var}(\Gamma) \cap \mathsf{var}(\Delta) = \emptyset$, *then for all* $m, n \in \mathbb{N}$, $n \geq m \geq 1$, $\mathsf{W}_m(M\{N/x\}) \leq \mathsf{W}_n(M) + \mathsf{NFO}(x, M) \cdot \mathsf{W}_n(N)$;
- *Bang case. If* $\triangleright \Gamma, !x \vdash M$ *and* $\triangleright \Delta \vdash N$, $\mathsf{var}(\Gamma) \cap \mathsf{var}(\Delta) = \emptyset$, *then for all* $m, n \in \mathbb{N}$, $n \geq m \geq 1$, $\mathsf{W}_m(M\{N/x\}) \leq \mathsf{W}_n(M) + n \cdot \mathsf{W}_n(N)$;
- *Quantum case. If* $\triangleright \Gamma, x_1, \ldots, x_k \vdash M$ *and* $\triangleright !\Delta, r_1, \ldots, r_k \vdash \langle r_1, \ldots, r_k \rangle$, $\mathsf{var}(\Gamma) \cap \mathsf{var}(!\Delta) = \emptyset$, *then for all* $m, n \in \mathbb{N}$, $n \geq m \geq 1$, $\mathsf{W}_m(M\{r_1/x_1, \ldots, r_k/x_k\}) \leq \mathsf{W}_n(M)$;

Proof. The four statements can be proved by induction on the structure of the derivation for M. We give here only some cases as examples.

- Linear case. For example, if M is x, with $\overline{!\Delta, x \vdash x}$. We have $\mathsf{W}_m(x\{N/x\}) = \mathsf{W}_m(N) \leq \mathsf{W}_n(x) + \mathsf{W}_n(N)$
- Contraction case. For example, suppose M is PQ. The last rule in the derivation for $\triangleright \Gamma, \#x \vdash M$ must have the following shape:

$$\frac{\Psi_1, \#\Delta_1 \vdash P \quad \Psi_2, \#\Delta_2 \vdash Q}{\Psi_1, \Psi_2, \#\Delta_1 \cup \#\Delta_2 \vdash PQ}\ \mathsf{app}$$

  Suppose that $\#x$ is both in $\#\Delta_1$ and in $\#\Delta_2$. By induction hypothesis we have $\mathsf{W}_m(P\{N/x\}) \leq \mathsf{W}_n(P) + \mathsf{NFO}(x, P) \cdot \mathsf{W}_n(N)$, and $\mathsf{W}_m(Q\{N/x\}) \leq \mathsf{W}_n(Q) + \mathsf{NFO}(x, Q) \cdot \mathsf{W}_n(N)$. Now:

$$
\begin{aligned}
\mathsf{W}_m(P(Q)\{N/x\}) &= \mathsf{W}_m(P\{N/x\}Q\{N/x\}) \\
&= \mathsf{W}_m(P\{N/x\}) + \mathsf{W}_m(Q\{N/x\}) + 1 \\
&\overset{IH}{\leq} \mathsf{W}_n(P) + \mathsf{NFO}(x,P)\cdot \mathsf{W}_n(N) \\
&\quad + \mathsf{W}_n(Q) + \mathsf{NFO}(x,Q)\cdot \mathsf{W}_n(N) + 1 \\
&= \mathsf{W}_n(P) + \mathsf{W}_n(Q) + 1 \\
&\quad + (\mathsf{NFO}(x,P) + \mathsf{NFO}(x,Q))\cdot \mathsf{W}_n(N) \\
&= \mathsf{W}_n(P(Q)) + \mathsf{NFO}(x,P(Q))\cdot \mathsf{W}_n(N).
\end{aligned}
$$

- Bang case. For Example: M is $!P$. P may have two possible derivations, by means of prom rule: either
$$\frac{\Delta \vdash P}{!\Delta, !\Delta_1, !x \vdash !P}\ \mathsf{prom}$$
or
$$\frac{\Delta, x \vdash P}{!\Delta, !\Delta_1, !x \vdash !P}\ \mathsf{prom}.$$
The only interesting case is the second. Using the linear case, we obtain
$$
\begin{aligned}
\mathsf{W}_m((!P)\{N/x\}) &= \mathsf{W}_m(!(P\{N/x\})) \\
&= m\cdot \mathsf{W}_m(P\{N/x\}) + 1 \\
&\leq m\cdot(\mathsf{W}_n(P) + \mathsf{W}_n(N)) + 1 \leq n\cdot \mathsf{W}_n(P) + 1 + n\cdot \mathsf{W}_n(N) \\
&= \mathsf{W}_n(!P) + n\cdot \mathsf{W}_n(N).
\end{aligned}
$$
- Quantum case. For example: M is PQ and for simplicity, suppose that $x_1,\dots,x_k$ occur in P. Then
$$
\begin{aligned}
\mathsf{W}_m(M\{r_1/x_1,\dots,r_k/x_k\}) &= \mathsf{W}_m(P(Q)\{r_1/x_1,\dots,r_k/x_k\}) \\
&= \mathsf{W}_m(P\{r_1/x_1,\dots,r_k/x_k\}(Q)) \\
&= \mathsf{W}_m(P\{r_1/x_1,\dots,r_k/x_k\}) + \mathsf{W}_m(Q) + 1 \\
&\overset{IH}{\leq} \mathsf{W}_n(P) + \mathsf{W}_m(Q) + 1 \leq \mathsf{W}_n(P) + \mathsf{W}_n(Q) + 1 \\
&= \mathsf{W}_n(PQ)
\end{aligned}
$$

This concludes the proof.

The following lemma tell us that weight $\mathsf{W}(\cdot)$, as for $\mathsf{D}(\cdot)$, is monotone:

Lemma 6.28. *(i) If $M \to_{\mathscr{K}} N$, then $\mathsf{W}(M) \geq \mathsf{W}(N)$;*
(ii) if $M \to_{\mathscr{N}} N$, then $\mathsf{W}(M) > \mathsf{W}(N)$.

Proof. By means of the previous substitution lemmas it is possible to prove that for all terms M, N and for all $n, m \in \mathbb{N}$, $n \geq m \geq 1$ and $n \geq \mathsf{D}(M)$, (i) if $M \to_{\mathscr{K}} N$ then $\mathsf{W}_n(M) \geq \mathsf{W}_m(N)$, and (ii) if $M \to_{\mathscr{N}} N$ then $\mathsf{W}_n(M) > \mathsf{W}_m(N)$. The proof is by induction on the derivation of $\to_{\mathscr{L}}$. We cite only the most interesting cases.

(i) Notice that, by previous lemma, if $M \to_{\mathscr{K}} N$, then $\mathsf{D}(M) = \mathsf{D}(N)$. The result follows by definitions. Inductive steps are performed by means of context closures.

(ii) Let r be the last rule of the derivation.
- M is $(\lambda !x.P)!Q$ and the reduction rule is $(\lambda !x.P)!Q \to_{\mathsf{c}.\beta} P\{Q/x\}$. We have to distinguish two sub-cases
  - if the derivation for M is

$$\dfrac{\dfrac{\Psi, \#\Delta_1, \#x \vdash P}{\Psi, \#\Delta_1 \vdash \lambda !x.P} \qquad \dfrac{\Delta_2 \vdash Q}{!\Delta_2, !\Delta_3 \vdash !Q}\,\mathsf{prom}}{\Psi, \#\Delta_1, !\Delta_2, !\Delta_3 \vdash (\lambda !x.P)!Q}\,\mathsf{app}$$

    then we can exploit the contraction case of Lemma 6.27 as follows:

$$\begin{aligned}\mathsf{W}_n((\lambda !x.P)!Q) &= \mathsf{W}_n(\lambda !x.P) + \mathsf{W}_n(!Q) + 1\\ &= \mathsf{W}_n(\lambda !x.P) + n \cdot \mathsf{W}_n(Q) + 2\\ &= \mathsf{W}_n(P) + n \cdot \mathsf{W}_n(Q) + 3\\ &> \mathsf{W}_n(P) + n \cdot \mathsf{W}_n(Q)\\ &\geq \mathsf{W}_n(P) + \mathsf{NFO}(x, P) \cdot \mathsf{W}_n(Q)\\ &\geq \mathsf{W}_m(P\{Q/x\}).\end{aligned}$$

  - if the derivation for M is

$$\dfrac{\dfrac{\Psi, \#\Delta_1, !x \vdash P}{\Psi, \#\Delta_1 \vdash \lambda !x.P} \qquad \dfrac{\Delta_2 \vdash Q}{!\Delta_2, !\Delta_3 \vdash !Q}\,\mathsf{prom}}{\Psi, \#\Delta_1, !\Delta_2, !\Delta_3 \vdash (\lambda !x.P)!Q}\,\mathsf{app}$$

    then we can exploit the bang case of Lemma 6.27 as follows:

$$\begin{aligned}\mathsf{W}_n((\lambda !x.P)!Q) &= \mathsf{W}_n(\lambda !x.P) + \mathsf{W}_n(!Q) + 1\\ &= \mathsf{W}_n(\lambda !x.P) + n \cdot \mathsf{W}_n(Q) + 2\\ &= \mathsf{W}_n(P) + n \cdot \mathsf{W}_n(Q) + 3\\ &> \mathsf{W}_n(P) + n \cdot \mathsf{W}_n(Q)\\ &\geq \mathsf{W}_m(P\{Q/x\}).\end{aligned}$$

    and we obtain the result by Lemma 6.25.

This concludes the proof.

The weight $\mathsf{W}(\cdot)$ and the box-depth $\mathsf{B}(\cdot)$ are related by the following properties:

Lemma 6.29. *For every term M, for all positive $n \in \mathbb{N}$, $\mathsf{W}_n(M) \leq |M| \cdot n^{\mathsf{B}(M)}$*

Proof. By induction on M:
- M is a variable, a constant or a quantum variable. We have

$$\mathsf{W}_n(M) = 1 \leq 1 \cdot n^0 = |M| \cdot n^{\mathsf{B}(M)}.$$

- M is $\mathsf{new}(N)$:

$$\begin{aligned}\mathsf{W}_n(\mathsf{new}(N)) &= \mathsf{W}_n(N) + 1 \overset{IH}{\leq} |N| \cdot n^{\mathsf{B}(N)} + 1\\ &\leq |N| \cdot n^{\mathsf{B}(N)} + n^{\mathsf{B}(N)} = (|N| + 1) \cdot n^{\mathsf{B}(N)}\\ &= |M| \cdot n^{\mathsf{B}(N)} = |M| \cdot n^{\mathsf{B}(M)}.\end{aligned}$$

- M is $!N$:

$$\begin{aligned}\mathsf{W}_n(!N) &= n \cdot \mathsf{W}_n(N) + 1 \overset{IH}{\leq} n \cdot |N| \cdot n^{\mathsf{B}(N)} + 1 \\ &= |N| \cdot n^{\mathsf{B}(N)+1} + 1 \leq |N| \cdot n^{\mathsf{B}(N)+1} + n^{\mathsf{B}(N)+1} \\ &= (|N| + 1) \cdot n^{\mathsf{B}(N)+1} = |M| \cdot n^{\mathsf{B}(M)}.\end{aligned}$$

- M is PQ:

$$\begin{aligned}\mathsf{W}_n(P(Q)) &= \mathsf{W}_n(P) + \mathsf{W}_n(Q) + 1 \overset{IH}{\leq} |P| \cdot n^{\mathsf{B}(P)} + |Q| \cdot n^{\mathsf{B}(Q)} + 1 \\ &\leq |P| \cdot n^{\mathsf{B}(P(Q))} + |Q| \cdot n^{\mathsf{B}(P(Q))} + 1 \\ &\leq |P| \cdot n^{\mathsf{B}(P(Q))} + |Q| \cdot n^{\mathsf{B}(P(Q))} + n^{\mathsf{B}(P(Q))} \\ &= (|P| + |Q| + 1) \cdot n^{\mathsf{B}(P(Q))} = |M| \cdot n^{\mathsf{B}(M)}.\end{aligned}$$

- M is $\langle N_1, \ldots, N_k \rangle$:

$$\begin{aligned}\mathsf{W}_n(\langle N_1, \ldots, N_k \rangle) \; &= \; \mathsf{W}_n(N_1) + \ldots + \mathsf{W}_n(N_k) + 1 \\ &\overset{IH}{\leq} \; |N_1| \cdot n^{\mathsf{B}(N_1)} + \ldots + |N_k| \cdot n^{\mathsf{B}(N_k)} + 1 \\ &\leq \; |N_1| \cdot n^{\mathsf{B}(M)} + \ldots + |N_k| \cdot n^{\mathsf{B}(M)} + 1 \\ &\leq \; |N_1| \cdot n^{\mathsf{B}(M)} + \ldots + |N_k| \cdot n^{\mathsf{B}(M)} + n^{\mathsf{B}(M)} = |M| \cdot n^{\mathsf{B}(M)}.\end{aligned}$$

- M is $\lambda x.N$ or $\lambda !x.N$ or $\lambda \langle x_1, \ldots, x_k \rangle.N$:

$$\begin{aligned}\mathsf{W}_n(M) &= \mathsf{W}_n(N) + 1 \overset{IH}{\leq} |N| \cdot n^{\mathsf{B}(N)} + 1 \\ &\leq |N| \cdot n^{\mathsf{B}(N)} + n^{\mathsf{B}(N)} = (|N| + 1) \cdot n^{\mathsf{B}(N)} = |M| \cdot n^{\mathsf{B}(M)}.\end{aligned}$$

This concludes the proof.

Lemma 6.30. *For every term M, $\mathsf{W}(M) \leq |M|^{\mathsf{B}(M)+1}$.*

Proof. By means of Lemma 6.29 and Lemma 6.22: $\mathsf{W}(M) = \mathsf{W}_{\mathsf{D}(M)}(M) \leq |M| \cdot \mathsf{D}(M)^{\mathsf{B}(M)} \leq |M| \cdot |M|^{\mathsf{B}(M)} = |M|^{\mathsf{B}(M)+1}$.

We have all the technical tools to prove another crucial lemma:

Lemma 6.31. *If $M \xrightarrow{*} N$, then $|N| \leq |M|^{\mathsf{B}(M)+1}$*

Proof. By means of Lemma 6.26, Lemma 6.28 and Lemma 6.30: $|N| \leq \mathsf{W}(N) \leq \mathsf{W}(M) \leq |M|^{\mathsf{B}(M)+1}$.

With all the intermediate lemmas we have just presented, proving that SQ is polystep is relatively easy:

Theorem 6.32 (Bounds). *There is a family of unary polynomials $\{p_n\}_{n \in \mathbb{N}}$ such that for any term M, for any $m \in \mathbb{N}$, if $M \xrightarrow{m} N$ (M reduces to N in m steps) then $m \leq p_{\mathsf{B}(M)}(|M|)$ and $|N| \leq p_{\mathsf{B}(M)}(|M|)$.*

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

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

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

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

Finally:

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

Moreover,

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

This concludes the proof.

Here is the main result of this section:

Theorem 6.33 (Polytime Soundness). *The following inclusions hold:* $\mathsf{ESQ} \subseteq \mathsf{EQP}$*,* $\mathsf{BSQ} \subseteq \mathsf{BQP}$ *and* $\mathsf{ZSQ} \subseteq \mathsf{ZQP}$*.*

Proof. Let us consider the first inclusion. Suppose a language $\mathcal{L}$ is in ESQ. This implies that $\mathcal{L}$ can be $(n,s,r,1)$-decided by a term M. By the Standardization Theorem, for every $t \in \{0,1\}^*$, there is a CNQ computation $\{C_i^t\}_{1\leq i\leq n_t}$ starting at $[1, \emptyset, M!^n\lceil t\rceil^{\{0,1\}}]$. By Theorem 6.32, n_t is bounded by a polynomial on the length $|t|$ of t. Moreover, the size of any C_i^t (that is to say, the sum of the term in C_i^t and the number of quantum variables in the second component of C_i^t) is itself bounded by a polynomial on $|t|$. Since $\{C_i^t\}_{1\leq i\leq n_t}$ is CNQ, any classical reduction step comes before any new-reduction step, which itself comes before any quantum reduction step. As a consequence, there is a polynomial time deterministic Turing machine which, for every t, computes one configuration in $\{C_i^t\}_{i\leq n_t}$ which only contains non-classical redexes (if any). But notice that a configuration only containing non-classical redexes is nothing but a concise abstract representation of a quantum circuit, fed with boolean inputs. Moreover, all the quantum circuits produced in this way are finitely generated, i.e., they can only contain the quantum gates (i.e. unitary operators) which appears in M, since $!^n\lceil t\rceil^{\{0,1\}}$ does not contain any unitary operator and reduction does not introduce new unitary operators in the underlying term. Summing up, the first component $\mathcal{Q}$ of $C^t_{n_t}$ is simply an element of an Hilbert Space $\mathcal{H}(\{q_1,\dots,q_m\})$ (where $[q_1,\dots,q_m]$ is the third component of $C^t_{n_t}$) obtained by evaluating a finitely generated quantum circuit whose size is polynomially bounded on $|t|$ and whose code can be effectively computed from t in polynomial time. By the results in [74], $L \in \mathsf{EQP}$. The other two inclusions can be handled in the same way.

6.8 Polytime Completeness

In Section 3.2.4 we recalled Yao's encoding of Quantum Turing machines into quantum circuit families [104]. In this Section we use the resut in order to prove SQ polytime completeness.

6.8.1 Encoding Polytime Quantum Turing Machines

We now need to show that SQ is able to simulate Yao's construction. Moreover, the simulation must be uniform, i.e. there must be *a single* term M generating all the possible L_m where m varies over the natural numbers.

Proposition 6.34. *For every n, there is a term M^n_G which uniformly generates G_m, i.e. such that whenever L n-encodes the natural number m, $M^n_G L \to_c R^m_G$ where R^m_G encodes G_m.*

Proof. Consider the following terms:

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

For the purpose of proving the correctness of the encoding, let us define $P^{n,m}_G$ for every $n,m\in\mathbb{N}$ by induction on m as follows:

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

First of all, observe that if L n-encodes the natural number m, then $N^n_G L \to_c P^m_G$. Indeed, if L n-encodes m, then

$$\begin{aligned}
N^n_G L &\to_c L!^n(\lambda y.\lambda z.\mathsf{extract}_{\lambda+2}((L_G y)z))(\lambda y.y) \\
&\to_c \underbrace{P(P(P(\dots(P}_{m\text{ times}}(\lambda x.x))\dots))) = P^m_G.
\end{aligned}$$

where $P = (\lambda y.\lambda z.(\mathsf{extract}_{\lambda+2}((L_G y)z)))$. Now, we can prove that for every $m\in\mathbb{N}$:

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

where

$$\mathcal{R} = \mathbf{cnot}_{\langle\langle q_{\lambda+1}, q_{\lambda+2}\rangle\rangle}(\mathbf{cnot}_{\langle\langle q_{2\lambda+3}, q_{2\lambda+4}\rangle\rangle}(\ldots(\mathbf{cnot}_{\langle\langle q_{m(\lambda+2)-1}, q_{m(\lambda+2)}\rangle\rangle}(\mathcal{Q}))\ldots))$$

By induction on m:

- If $m = 0$, then

$$[\mathcal{Q}, \mathcal{QV}, P_G^0[q_1, \ldots, q_h]] \xrightarrow{*} [\mathcal{Q}, \mathcal{QV}, [q_1, \ldots, q_h]]$$

- Now, suppose the thesis holds for m. Then:

$$\begin{aligned}
&[\mathcal{Q}, \mathcal{QV}, P_G^{m+1}[q_1, \ldots, q_{(m+1)(\lambda+2)}, ldots, q_h]]\\
\xrightarrow{*}\ &[\mathcal{Q}, \mathcal{QV}, \mathsf{extract}_{\lambda+2}(L_G^n P_G^m)[q_1, \ldots, q_h]]\\
\xrightarrow{*}\ &[\mathcal{Q}, \mathcal{QV}, (L_G P_G^m)[q_{\lambda+3}, \ldots, q_h]q_1 \ldots q_{\lambda+2}]\\
\xrightarrow{*}\ &[\mathcal{Q}, \mathcal{QV}, \mathsf{append}_{\lambda+2}(P_G^m[q_{\lambda+3}, \ldots, q_h])q_1 \ldots q_{\lambda+2}]\\
\xrightarrow{*}\ &[\mathcal{R}, \mathcal{QV}, \mathsf{append}_{\lambda+2}[q_{\lambda+3}, \ldots, q_h]q_1 \ldots q_{\lambda+2}]\\
\xrightarrow{*}\ &[\mathcal{S}, \mathcal{QV}, [q_1, \ldots, q_h]]
\end{aligned}$$

where

$$\begin{aligned}
\mathcal{R} &= \mathbf{cnot}_{\langle\langle q_{2\lambda+3}, q_{\lambda+4}\rangle\rangle}(\mathbf{cnot}_{\langle\langle q_{3\lambda+5}, q_{3\lambda+6}\rangle\rangle}(\ldots(\mathbf{cnot}_{\langle\langle q_{m(\lambda+2)-1}, q_{m(\lambda+2)}\rangle\rangle}(\mathcal{Q}))\ldots))\\
\mathcal{S} &= \mathbf{cnot}_{\langle\langle q_{\lambda+1}, q_{\lambda+2}\rangle\rangle}(\mathbf{cnot}_{\langle\langle q_{2\lambda+3}, q_{2\lambda+4}\rangle\rangle}(\ldots(\mathbf{cnot}_{\langle\langle q_{m(\lambda+2)-1}, q_{m(\lambda+2)}\rangle\rangle}(\mathcal{Q}))\ldots))
\end{aligned}$$

Now, if L n-encodes the natural number m, then

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

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

Proposition 6.35. *For every n, there is a term M_J^n which uniformly generates J_m, i.e. such that $M_J^n L \to_c R_J^m$ where R_J^m encodes J_m whenever L n-encodes the natural number m.*

Proof. Consider the following terms:

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

For the purpose of proving the correctness of the encoding, let us define $R_J^{n,m}$ for every $n, m \in \mathbb{N}$ by induction on m as follows:

$$\begin{aligned}
R_J^0 &= \lambda x.x\\
R_J^{m+1} &= \lambda z.(\mathsf{extract}_{\eta+\lambda+2}(L_J R_J^m))z
\end{aligned}$$

First of all, observe that if L n-encodes the natural number m, then $M^n_J L \to_c R^m_G$. Indeed, if L n-encodes m, then

$$\begin{aligned} M^n_J L &\to_c L!^n(N_J)(\lambda y.y) \\ &\to_c \underbrace{N_J(N_J(N_J(\ldots(N_J}_{m \text{ times}}(\lambda x.x))\ldots))) = R^m_J. \end{aligned}$$

Now, we can prove that for every $m \in \mathbb{N}$:

$$[\mathcal{Q}, \mathcal{QV}, R^m_J[q_1, \ldots, q_{\eta+(2m+1)(\lambda+2)}]] \xrightarrow{*} [\mathcal{R}, \mathcal{QV}, [q_1, \ldots, q_{\eta+(2m+1)(\lambda+2)}]]$$

where

$$\mathcal{R} = J_m(\mathcal{Q})$$

by induction on m:

- If $m = 0$, then

$$[\mathcal{Q}, \mathcal{QV}, R^0_J[q_1, \ldots, q_h]] \xrightarrow{*} [\mathcal{Q}, \mathcal{QV}, [q_1, \ldots, q_h]]$$

- Now, suppose the thesis holds for m. Then:

$$\begin{aligned} & [\mathcal{Q}, \mathcal{QV}, R^{m+1}_J[q_1, \ldots, q_{(2m+3)(\lambda+2)}]] \\ \xrightarrow{*}\ & [\mathcal{Q}, \mathcal{QV}, \mathsf{extract}_{\eta+\lambda+2}(L_J R^m_J)[q_1, \ldots, q_{(2m+3)(\lambda+2)}]] \\ \xrightarrow{*}\ & [\mathcal{Q}, \mathcal{QV}, \mathsf{extract}_{\eta+2(\lambda+2)}(P_J q_{\eta+1} \ldots q_{\eta+\lambda+2}) \\ & (R^m_J(\mathsf{append}_\eta[q_{\eta+(\lambda+2)+1}, \ldots, q_{(2m+3)(\lambda+2)}]q_1 \ldots q_\eta))] \\ \xrightarrow{*}\ & [\mathcal{R}, \mathcal{QV}, \mathsf{extract}_{\eta+2(\lambda+2)}(P^n_J q_{\eta+1} \ldots q_{\eta+\lambda+2}) \\ & ([q_1, \ldots, q_\eta, q_{\eta+(\lambda+2)+1}, \ldots, q_{(2m+3)(\lambda+2)}]] \\ \xrightarrow{*}\ & [\mathcal{R}, \mathcal{QV}, P_J q_{\eta+1} \ldots q_{\eta+\lambda+2}[q_{\eta+3(\lambda+2)+1}, \ldots, q_{(2m+3)(\lambda+2)}]q_1 \ldots q_\eta q_{\eta+\lambda+3} \ldots q_{\eta+3(\lambda+2)}] \\ \xrightarrow{*}\ & [\mathcal{R}, \mathcal{QV}, (\lambda\langle q_1. \ldots .\lambda q_{\eta+3(\lambda+2)}\rangle. \\ & (\mathsf{append}_{\eta+3(\lambda+2)}[q_{\eta+3(\lambda+1)+1}, \ldots, q_{(2m+3)(\lambda+2)}]q_1 \ldots q_{\eta+3(\lambda+2)})(H\langle q_1, \ldots, q_{\eta+3(\lambda+2)}\rangle))] \\ \xrightarrow{*}\ & [\mathcal{S}, \mathcal{QV}, (\mathsf{append}_{\eta+3(\lambda+2)}[q_{\eta+3(\lambda+1)+1}, \ldots, q_{(2m+3)(\lambda+2)}]q_1 \ldots q_{\eta+3(\lambda+2)}] \\ \xrightarrow{*}\ & [\mathcal{S}, \mathcal{QV}, [q_1, \ldots, q_{(2m+3)(\lambda+2)}]] \end{aligned}$$

  where

$$\begin{aligned} \mathcal{R} &= (I_{\langle q_{\eta+1}, \ldots, q_{\eta+\lambda+2}\rangle} \otimes (J_m)_{\langle q_1, \ldots, q_\eta, q_\eta+\lambda+3, \ldots, q_{(2m+3)(\lambda+2)}\rangle})(\mathcal{Q}) \\ \mathcal{S} &= (I_{\langle q_{\eta+3(\lambda+2)+1}, \ldots, q_{(2m+3)(\lambda+2)}\rangle} \otimes H_{\langle q_1, \ldots, q_{\eta+3(\lambda+2)}\rangle})(\mathcal{R}) \end{aligned}$$

  which implies

$$\mathcal{S} = ((J_{m+1})_{\langle q_1, \ldots, q_{(2m+3)(\lambda+2)}\rangle})(\mathcal{Q}).$$

This concludes the proof.

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

$$q_1, \ldots, q_n = s,$$

which means that the value of the variables $q_1, \ldots, q_n$ is $s \in \{0,1\}^n$.

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

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

where for every $q \in Q$

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

Theorem 6.36. *For every polynomial time quantum Turing machine $\mathcal{M} = (Q, \Sigma, \delta)$ there is a term $M_{\mathcal{M}}$ such that $M_{\mathcal{M}}$ simulates the machine $\mathcal{M}$.*

Proof. The Theorem follows from Proposition 6.34, Proposition 6.35 and Proposition 6.17. More precisely, the term $M_{\mathcal{M}}$ has the form $\lambda!x.(M^{circ}_{\mathcal{M}} x)(M^{init}_{\mathcal{M}} x)$ where
- $M^{circ}_{\mathcal{M}}$ builds the Yao's circuit, given a string representing the input;
- $M^{init}_{\mathcal{M}}$ builds a list of quantum variables to be fed to the Yao's circuit, given a string representing the input.

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

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

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

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

where

$$\begin{aligned} N^{circ}_{\mathcal{M}} &= \lambda x.M_{2p+1}(\{\mathsf{strtonat}_{\Sigma}\}_{2k+1}(M^{2k+2}_{id} x)) \\ L^{circ}_{\mathcal{M}} &= \lambda x.(\{P^{circ}_{\mathcal{M}}\}_{2k+1} x) \\ P^{circ}_{\mathcal{M}} &= \lambda!z.\lambda y.(M^{2k+1}_{J}(M_{2p+1}(\{\mathsf{strtonat}_{\Sigma}\}_{2k+1} z)))(M^{2k+1}_{G}(M_{2p+1}(\{\mathsf{strtonat}_{\Sigma}\}_{2k+1} z)))y \end{aligned}$$

M^{2k+1}_{G} comes from Proposition 6.34, M^{2k+1}_{J} comes from Proposition 6.35 and M_{2p+1} comes from Proposition 6.17. Now, consider any string $s = b_1 \ldots b_n \in \Sigma^*$. First of all:

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

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

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

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

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

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

Corollary 6.37 (Polytime Completeness). *The following inclusions hold:* $\mathsf{EQP} \subseteq \mathsf{ESQ}$, $\mathsf{BQP} \subseteq \mathsf{BSQ}$ *and* $\mathsf{ZQP} \subseteq \mathsf{ZSQ}$.

From Theorem 6.33 and Corollary 6.37, $\mathsf{EQP} = \mathsf{ESQ}$, $\mathsf{BQP} = \mathsf{BSQ}$ and $\mathsf{ZQP} = \mathsf{ZSQ}$. In other words, there is a perfect correspondence between (polynomial time) quantum complexity classes and classes of languages decidable by SQ terms.

7

Adding A Measurement Operator to Q

In Chapter 4 we have introduced the measurement–free, untyped quantum λ–calculus, Q. Now, we study an extension of Q obtained by endowing the language of terms with a suitable measurement operator and coherently extending the reduction relation. We investigate the resulting calculus, called Q*, focusing on confluence.

An explicit measurement operator in the syntax allows an observation at an intermediate step of the computation: this feature is needed if we want, for example, to write algorithms such as Shor's factorization. In quantum calculi the intended meaning of a measurement is to observe the status of a possibly superimposed quantum bit, giving as output a classical bit; the two possible outcomes (i.e., the two possible values of the obtained classical bit) can be observed with two probabilities summing to 1. Since measurement forces a probabilistic evolution in the computation, it is not surprising that we need probabilistic instruments in order to investigate the main features of the language.

But, is it possible to preserve confluence in the probabilistic setting induced by measurements? Apparently, the questions above cannot receive a positive answer: as we will see in Section 7.3, it is possible to exhibit a configuration C such that there are two different reductions starting at C and ending in two essentially different normal forms configurations $[1, \emptyset, 0]$ and $[1, \emptyset, 1]$. In other words, confluence fails in its usual form. But the question now becomes: are the usual notions of computations and confluence adequate in this setting?

In Q* there are two sources of divergence, which should not be confused:

- on the one hand, a redex involving the measurement operator can be reduced in two different ways, i.e., divergence can come from *a single redex*;
- on the other hand, a term can contain more than one redex and the calculus is not endowed with a reduction strategy. As a consequence, some configurations can be reduced in two distinct ways due to the presence of *different redexes*.

We cannot hope to be confluent with respect to the first source of divergence, but we can anyway ask ourselves whether all reduction strategies are somehow equivalent. More precisely, we say that Q* is *confluent* if *for every configuration C and for every configuration in normal form D, there is a fixed real number p such that the probability of observing D when reducing C is always p, independently of the reduction strategy*.

This notion of confluence can be captured by analyzing rewriting on *mixed states* rather than rewriting on configurations. A mixed state is a probabilistic distribution on configurations whose support is finite. Rewriting on configurations naturally extend to rewriting on mixed states. Rewriting on mixed states is *not* a probabilistic relation, and the notion of confluence is the standard one from rewriting theory.

We prove that Q* is indeed confluent in this sense, by means of non standard techniques. The key point is that we need a new definition of computation. The usual notion of computation as a sequence of reductions is not adequate here. A notion of *probabilistic computation* replaces it, essentially as something more general than a sequence of reduction but less general than the reduction tree: a probabilistic computation is a (possibly) infinite tree, in which the binary choice (a node can have at most two children) corresponds to the two different outcomes of a measurement. The set of leaves of a probabilistic computation is consequently a probabilistic distribution of configurations.The notion of reduction is then extended to mixed states defining the so called *mixed computations*.

Another important property of any quantum lambda calculus with measurement is the importance of infinite computations. As we will see in Section 7.3, it is possible and necessary to deal with infinite computations in order to properly deal with finite final outcomes (finite probability distribution of finite normal form configurations). *This phenomenon forced us to extend the study of confluence also to the case of infinite probabilistic computations*.

7.1 The Q* calculus: Syntax and Computations

In Q* there are three kinds of operations on quantum registers: *(i)* the *new* operation, responsible for the creation of qubits; *(ii) unitary operators*: each unitary operator $\mathbf{U}_{\langle\langle q_1,\ldots,q_n\rangle\rangle}$ corresponds to a pure quantum operation acting on qubits with names $q_1,\ldots,q_n$ *(iii) one qubit measurement* operations $\mathcal{M}_{r,0},\mathcal{M}_{r,1}$ responsible of the probabilistic reduction of the quantum state plus the destruction of the measured qubit referenced by r: given a quantum register $\mathcal{Q}\in\mathcal{H}(\mathcal{QV})$, and a quantum variable name $r\in\mathcal{QV}$, we allow the measurement of the qubit with name r.

7.1.1 Terms, Judgements and Well-Formed-Terms

Let $\mathcal{U}$ be an elementary set (see Chapter 3, Section 3.2.2) of unitary operators. Let us associate to each elementary operator $\mathbf{U}\in\mathcal{U}$ a symbol U. The set of *term expressions*, or *terms* for short, is defined by the following grammar:

$$
\begin{array}{lll}
x & ::= x_0, x_1, \ldots & \textit{classical variables}\\
r & ::= r_0, r_1, \ldots & \textit{quantum variables}\\
\pi & ::= x \mid \langle x_1,\ldots,x_n\rangle & \textit{linear patterns}\\
\psi & ::= \pi \mid !x & \textit{patterns}\\
B & ::= 0 \mid 1 & \textit{boolean constants}\\
U & ::= U_0, U_1, \ldots & \textit{unitary operators}\\
C & ::= B \mid U & \textit{constants}\\
M & ::= x \mid r \mid !M \mid C \mid \mathtt{new}(M) \mid M_1M_2 \mid & \\
 & \quad \mathsf{meas}(M) \mid \mathsf{if}\ N\ \mathsf{then}\ M_1\ \mathsf{else}\ M_2 \mid & \\
 & \quad \langle M_1,\ldots,M_n\rangle \mid \lambda\psi.M & \textit{terms (where } n\geq 2)
\end{array}
$$

The syntax is clearly the extension of the syntax of Q ; we adopt the same assumptions on variables and on α–conversion.
Notice that the term constructor $\mathsf{meas}(\cdot)$ perform a single qubit measurement when applied to a quantum variable.

For each qvs $\mathcal{QV}$ and for each quantum variable $r \in \mathcal{QV}$, we assume to have two, measurement based, linear transformation of quantum registers: $\mathcal{M}_{r,0}, \mathcal{M}_{r,1} : \mathcal{H}(\mathcal{QV}) \to \mathcal{H}(\mathcal{QV} - \{r\})$ (see Section 7.2 for more details).

Enviroments and judgements are defined exactly as for Q (see Section 4.2.3).

We say that a judgement $\Gamma \vdash M$ is *well–formed* (*notation:* $\triangleright \Gamma \vdash M$) if it is derivable from the *well–forming rules* in Figure 7.1. The well-forming rules of Q^* extend the well–forming rules of Q (Figure 4.2) with the two new rules meas and if.

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

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

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

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

Fig. 7.1. Well–Forming Rules

Remark 7.1. Q^* comes equipped with two constants 0 and 1 (as for Q), and an $\mathsf{if}\ (\cdot)\ \mathsf{then}\ (\cdot)\ \mathsf{else}\ (\cdot)$ constructor. However, these constructors can be thought of as syntactic sugar. Indeed, 0 and 1 can be encoded as pure terms: $0 = \lambda !x.\lambda !y.y$ and $1 = \lambda !x.\lambda !y.x$. In doing so, $\mathsf{if}\ M\ \mathsf{then}\ N\ \mathsf{else}\ L$ becomes $M!N!L$. The well–forming rule if (see Figure 7.1) of Q^* fully agrees with the above encodings.

7.2 Quantum Registers and Measurements

Before giving the definition of destructive measurement used in this thesis we must clarify something about quantum spaces.

The smallest quantum space is $\mathcal{H}(\emptyset)$, which is (isomorphic to) the field $\mathbb{C}$. The so called *empty quantum register* is nothing more than a unitary element of $\mathbb{C}$ (i.e., a complex number c such that $|c| = 1$). We have chosen the scalar number 1 as the canonical empty quantum register. In particular the number 1 represents also the computational basis of $\mathcal{H}(\emptyset)$.

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

$$\mathcal{H}(\mathcal{QV}) \otimes \mathcal{H}(\mathcal{RV}) \overset{i_s}{\simeq} \mathcal{H}(\mathcal{QV} \cup \mathcal{RV}).$$

In the rest of this thesis we will assume to work up-to such an isomorphism[1]. Note that the previous isomorphism holds even if either $\mathcal{QV}$ or $\mathcal{RV}$ is empty.

Since a quantum space $\mathcal{H}(\mathcal{QV})$ is an Hilbert space, $\mathcal{H}(\mathcal{QV})$ has a zero element $0_{\mathcal{QV}}$ (we will omit the subscript, when this does not cause ambiguity). In particular, if $\mathcal{QV} \cap \mathcal{RV} = \emptyset$, $\mathcal{Q} \in \mathcal{H}(\mathcal{QV})$ and $\mathcal{R} \in \mathcal{H}(\mathcal{RV})$, then $\mathcal{Q} \otimes 0_{\mathcal{RV}} = 0_{\mathcal{QV}} \otimes \mathcal{R} = 0_{\mathcal{QV} \cup \mathcal{RV}} \in \mathcal{H}(\mathcal{QV} \cup \mathcal{RV})$.

Definition 7.2 (Quantum registers). *Given a quantum space $\mathcal{H}(\mathcal{QV})$, a* quantum register *is any $\mathcal{Q} \in \mathcal{H}(\mathcal{QV})$ such that either $\mathcal{Q} = 0_{\mathcal{QV}}$ or $\mathcal{Q}$ is a normalised vector.*

Let $\mathcal{QV}$ be a qvs with cardinality $n \geq 1$. Moreover, let $\mathcal{Q} \in \mathcal{H}(\mathcal{QV})$ and let $r \in \mathcal{QV}$.+ Each state $\mathcal{Q}$ may be represented as follows:

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

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

Definition 7.3 (Destructive measurements). *Let $\mathcal{QV}$ be a qvs with cardinality $n = |\mathcal{QV}| \geq 1$, $r \in \mathcal{QV}$, $\{b_i\}_{i \in [1,2^{n-1}]}$ be the computational basis of $\mathcal{H}(\mathcal{QV} - \{r\})$ and $\mathcal{Q}$ be $\sum_{i=1}^{2^{n-1}} \alpha_i |r \mapsto 0\rangle \otimes b_i + \sum_{i=1}^{2^{n-1}} \beta_i |r \mapsto 1\rangle \otimes b_i \in \mathcal{H}(\mathcal{QV})$. The two linear functions*

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

such that

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

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

The just defined measurement operators are *general measurements* [58, 72]:

Proposition 7.4 (Completeness Condition). *Let $r \in \mathcal{QV}$ and $\mathcal{Q} \in \mathcal{H}(\mathcal{QV})$. Then $\mathsf{m}_{r,0}^{\dagger}\mathsf{m}_{r,0} + \mathsf{m}_{r,1}^{\dagger}\mathsf{m}_{r,1} = Id_{\mathcal{H}(\mathcal{QV})}$.*

Proof. In order to prove the proposition we will use the following general property of inner product spaces: let $\mathcal{H}$ be an inner product space and let $A : \mathcal{H} \to \mathcal{H}$ be a linear map. If for each $x, y \in \mathcal{H}$, $\langle Ax, y \rangle = \langle x, y \rangle$ then A is the identity map[3]. Let $\mathcal{Q}, \mathcal{R} \in \mathcal{H}(\mathcal{QV})$. If $\{b_i\}_{i \in [1,2^n]}$ is the computational basis of $\mathcal{H}(\mathcal{QV} - \{r\})$, then:

[1] in particular, if $\mathcal{Q} \in \mathcal{H}(\mathcal{QV})$, $r \notin \mathcal{QV}$ and $|r \mapsto c\rangle \in \mathcal{H}(\{r\})$ then $\mathcal{Q} \otimes |r \mapsto c\rangle$ will denote the element $i_s(\mathcal{Q} \otimes |r \mapsto c\rangle) \in \mathcal{H}(\mathcal{QV} \cup \{r\})$

[2] in the mathematical literature, computational basis are usually called standard basis; see [30], for the definition of computational/standard basis of $\mathcal{H}(\mathcal{QV})$.

[3] such a property is an immediate consequence of the *Riesz representation theorem*, see e.g. [81]

$$\mathcal{Q} = \sum_{i=1}^{2^n} \alpha_i |r \mapsto 0\rangle \otimes b_i + \sum_{i=1}^{2^n} \beta_i |r \mapsto 1\rangle \otimes b_i$$
$$\mathcal{R} = \sum_{i=1}^{2^n} \gamma_i |r \mapsto 0\rangle \otimes b_i + \sum_{i=1}^{2^n} \delta_i |r \mapsto 1\rangle \otimes b_i.$$

We have:

$$\begin{aligned}
\langle (\mathsf{m}^\dagger_{r,0}\mathsf{m}_{r,0} + \mathsf{m}^\dagger_{r,1}\mathsf{m}_{r,1})(\mathcal{Q}), \mathcal{R}\rangle &= \langle \mathsf{m}^\dagger_{r,0}\mathsf{m}_{r,0}(\mathcal{Q}), \mathcal{R}\rangle + \langle \mathsf{m}^\dagger_{r,1}\mathsf{m}_{r,1}(\mathcal{Q}), \mathcal{R}\rangle \\
&= \langle \mathsf{m}_{r,0}(\mathcal{Q}), \mathsf{m}_{r,0}(\mathcal{R})\rangle + \langle \mathsf{m}_{r,1}(\mathcal{Q}), \mathsf{m}_{r,1}(\mathcal{R})\rangle \\
&= \langle \sum_{i=1}^{2^n} \alpha_i b_i, \sum_{i=1}^{2^n} \gamma_i b_i\rangle + \langle \sum_{i=1}^{2^n} \beta_i b_i, \sum_{i=1}^{2^n} \delta_i b_i\rangle \\
&= \sum_{i=0}^{2^n} \alpha_i \gamma_i + \sum_{i=0}^{2^n} \beta_i \delta_i \\
&= \langle \mathcal{Q}, \mathcal{R}\rangle.
\end{aligned}$$

This concludes the proof.

For $c \in \{0, 1\}$, the measurement operators $\mathsf{m}_{r,c}$ enjoys the following properties:

Proposition 7.5. *Let $\mathcal{Q} \in \mathcal{H}(\mathcal{QV})$. Then:*

1. $\mathsf{m}_{r,c}(\mathcal{Q} \otimes |q \mapsto d\rangle) = (\mathsf{m}_{r,c}(\mathcal{Q})) \otimes |q \mapsto d\rangle$ *if $r \in \mathcal{QV}$ and $q \notin \mathcal{QV}$;*
2. $\langle \mathcal{Q} \otimes |s \mapsto d\rangle | \mathsf{m}^\dagger_{r,c}\mathsf{m}_{r,c} | \mathcal{Q} \otimes |s \mapsto d\rangle\rangle = \langle \mathcal{Q}, \mathsf{m}^\dagger_{r,c}\mathsf{m}_{r,c}|\mathcal{Q}\rangle$*; if $r \in \mathcal{QV}$ and $r \neq s$;*
3. $\mathsf{m}_{q,e}(\mathsf{m}_{r,d}(\mathcal{Q})) = \mathsf{m}_{r,d}(\mathsf{m}_{q,e}(\mathcal{Q}))$*; if $r, q \in \mathcal{QV}$.*

Proof. 1. Given the computational basis $\{b_i\}_{i \in [1,2^n]}$ of $\mathcal{H}(\mathcal{QV} - \{r\})$, we have that:

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

and therefore

$$\begin{aligned}
\mathsf{m}_{r,0}(\mathcal{Q} \otimes |q \mapsto d\rangle) &= \sum_{i=1}^{2^n} \alpha_i (b_i \otimes |r \mapsto d\rangle) \\
&= \left(\sum_{i=1}^{2^n} \alpha_i b_i\right) \otimes |q \mapsto d\rangle \\
&= (\mathsf{m}_{r,c}(\mathcal{Q})) \otimes |q \mapsto d\rangle.
\end{aligned}$$

In the same way we prove the equality for $\mathsf{m}_{r,1}$.

2. Just observe that:

$$\begin{aligned}
\langle \mathcal{Q} \otimes |s \mapsto d\rangle | \mathsf{m}^\dagger_{r,c}\mathsf{m}_{r,c} | \mathcal{Q} \otimes |s \mapsto d\rangle\rangle &= \langle \mathcal{Q} \otimes |s \mapsto d\rangle, \mathsf{m}^\dagger_{r,c}(\mathsf{m}_{r,c}(\mathcal{Q} \otimes |s \mapsto d\rangle))\rangle \\
&= \langle \mathsf{m}_{r,c}(\mathcal{Q} \otimes |s \mapsto d\rangle), \mathsf{m}_{r,c}(\mathcal{Q} \otimes |s \mapsto d\rangle)\rangle \\
&= \langle \mathsf{m}_{r,c}(\mathcal{Q}), \mathsf{m}_{r,c}(\mathcal{Q})\rangle \\
&= \langle \mathcal{Q}, \mathsf{m}^\dagger_{r,c}\mathsf{m}_{r,c}\mathcal{Q}\rangle = \langle \mathcal{Q} | \mathsf{m}^\dagger_{r,c}\mathsf{m}_{r,c} | \mathcal{Q}\rangle.
\end{aligned}$$

3. Given the computational basis $\{b_i\}_{i\in[1,2^n]}$ of $\mathcal{H}(\mathcal{QV}-\{r,q\})$, we have that:

$$\mathcal{Q} = \sum_{i=1}^{2^n} \alpha_i |r \mapsto 0\rangle \otimes |q \mapsto 0\rangle \otimes b_i + \sum_{i=1}^{2^n} \beta_i |r \mapsto 0\rangle \otimes |q \mapsto 1\rangle \otimes b_i + \sum_{i=1}^{2^n} \gamma_i |r \mapsto 1\rangle \otimes |q \mapsto 0\rangle \otimes b_i + \sum_{i=1}^{2^n} \delta_i |r \mapsto 1\rangle \otimes |q \mapsto 1\rangle \otimes b_i.$$

Let us show that $\mathsf{m}_{q,0}(\mathsf{m}_{r,0}(\mathcal{Q})) = \mathsf{m}_{r,0}(\mathsf{m}_{q,0}(\mathcal{Q}))$, the proof of other cases follow the same pattern.

$$\begin{aligned}\mathsf{m}_{r,0}(\mathsf{m}_{q,0}(\mathcal{Q})) &= \mathsf{m}_{r,0}\left(\sum_{i=1}^{2^n} \alpha_i |r \mapsto 0\rangle \otimes b_i + \sum_{i=1}^{2^n} \gamma_i |r \mapsto 1\rangle \otimes b_i\right)\\ &= \sum_{i=1}^{2^n} \alpha_i b_i = \mathsf{m}_{q,0}\left(\sum_{i=1}^{2^n} \alpha_i |q \mapsto 0\rangle \otimes b_i + \sum_{i=1}^{2^n} \beta_i |q \mapsto 1\rangle \otimes b_i\right)\\ &= \mathsf{m}_{q,0}(\mathsf{m}_{r,0}(\mathcal{Q})).\end{aligned}$$

This concludes the proof.

Given a qvs $\mathcal{QV}$ and a variable $r \in \mathcal{QV}$, we can define two linear maps:

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

which are "normalized" versions of $\mathsf{m}_{r,0}$ and $\mathsf{m}_{r,1}$ as follows:
1. if $\langle \mathcal{Q}|\mathsf{m}^{\dagger}_{r,c}\mathsf{m}_{r,c}|\mathcal{Q}\rangle = 0$ then $\mathcal{M}_{r,c}(\mathcal{Q}) = \mathsf{m}_{r,c}(\mathcal{Q})$;
2. if $\langle \mathcal{Q}|\mathsf{m}^{\dagger}_{r,c}\mathsf{m}_{r,c}|\mathcal{Q}\rangle \neq 0$ then $\mathcal{M}_{r,c}(\mathcal{Q}) = \frac{\mathsf{m}_{r,c}(\mathcal{Q})}{\sqrt{\langle \mathcal{Q}|\mathsf{m}^{\dagger}_{r,c}\mathsf{m}_{r,c}|\mathcal{Q}\rangle}}$.

Proposition 7.6. *Let $\mathcal{Q} \in \mathcal{H}(\mathcal{QV})$ be a quantum register. Then:*
1. *$\mathcal{M}_{r,c}(\mathcal{Q})$ is a quantum register;*
2. *$\mathcal{M}_{q,e}(\mathcal{Q} \otimes |r \mapsto d\rangle) = (\mathcal{M}_{q,e}(\mathcal{Q})) \otimes |r \mapsto d\rangle$, with $q \in \mathcal{QV}$ and $q \neq r$;*
3. *$\mathcal{M}_{q,e}(\mathcal{M}_{r,d}(\mathcal{Q})) = \mathcal{M}_{r,d}(\mathcal{M}_{q,e}(\mathcal{Q}))$, with $q, r \in \mathcal{QV}$;*
4. *if $q, r \in \mathcal{QV}$, $p_{r,c} = \langle \mathcal{Q}|\mathsf{m}^{\dagger}_{r,c}\mathsf{m}_{r,c}|\mathcal{Q}\rangle$, $p_{q,d} = \langle \mathcal{Q}|\mathsf{m}^{\dagger}_{q,d}\mathsf{m}_{q,d}|\mathcal{Q}\rangle$, $\mathcal{Q}_{r,c} = \mathcal{M}_{r,c}(\mathcal{Q})$, $\mathcal{Q}_{q,d} = \mathcal{M}_{q,d}(\mathcal{Q})$, $s_{r,c} = \langle \mathcal{Q}_{q,d}|\mathsf{m}^{\dagger}_{r,c}\mathsf{m}_{r,c}|\mathcal{Q}_{q,d}\rangle$, $s_{q,d} = \langle \mathcal{Q}_{r,c}|\mathsf{m}^{\dagger}_{q,d}\mathsf{m}_{q,d}|\mathcal{Q}_{r,c}\rangle$ then $p_{r,c} \cdot s_{q,d} = p_{q,d} \cdot s_{r,c}$;*
5. *$(\mathbf{U}_{\langle q_1,\ldots,q_k\rangle} \otimes \mathbf{I}_{\mathcal{QV}-\{q_1,\ldots,q_k\}})(\mathcal{M}_{r,c}(\mathcal{Q})) = \mathcal{M}_{r,c}((\mathbf{U}_{\langle q_1,\ldots,q_k\rangle} \otimes \mathbf{I}_{\mathcal{QV}-\{q_1,\ldots,q_k\}})(\mathcal{Q}))$ with $\{q_1,\ldots,q_k\} \subseteq \mathcal{QV}$ and $r \neq q_j$ for all $j = 1,\ldots,k$.*

Proof. The proofs of 1, 2 and 5 are immediate consequences of Proposition 7.5 and of general basic properties of Hilbert spaces. About 3 and 4: if $\mathcal{Q} = 0_{\mathcal{QV}}$ then the proof is trivial; if either $p_{r,c} = 0$ or $p_{q,d} = 0$ (possibly both), observe that $s_{r,c} = s_{q,d} = 0$ and $\mathcal{M}_{q,e}(\mathcal{M}_{r,d}(\mathcal{Q})) = \mathcal{M}_{r,d}(\mathcal{M}_{q,e}(\mathcal{Q})) = 0_{\mathcal{QV}-\{q,r\}}$ and conclude. Suppose now that $\mathcal{Q} \neq 0_{\mathcal{QV}}$, $p_{r,c} \neq 0$ and $p_{q,d} \neq 0$. Given the computational basis $\{b_i\}_{i\in[1,2^n]}$ of $\mathcal{H}(\mathcal{QV}-\{r,q\})$, we have that:

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

Let us examine the case $c = 0$ and $d = 0$ (the other cases can be handled in the same way).

$$p_{r,0} = \sum_{i=1}^{2^n} |\alpha_i|^2 + \sum_{i=1}^{2^n} |\beta_i|^2; \qquad p_{q,0} = \sum_{i=1}^{2^n} |\alpha_i|^2 + \sum_{i=1}^{2^n} |\gamma_i|^2;$$

$$\mathcal{Q}_{r,0} = \mathcal{M}_{r,0}(\mathcal{Q}) = \frac{\sum_{i=1}^{2^n} \alpha_i |q \mapsto 0\rangle \otimes b_i + \sum_{i=1}^{2^n} \beta_i |q \mapsto 1\rangle \otimes b_i}{\sqrt{p_{r,0}}}$$

$$\mathcal{Q}_{q,0} = \mathcal{M}_{q,0}(\mathcal{Q}) = \frac{\sum_{i=1}^{2^n} \alpha_i |r \mapsto 0\rangle \otimes b_i + \sum_{i=1}^{2^n} \gamma_i |r \mapsto 1\rangle \otimes b_i}{\sqrt{p_{q,0}}}$$

Now let us consider the two states:

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

By definition:

$$s_{q,0} = \frac{\sum_{i=1}^{2^n} |\alpha_i|^2}{p_{r,0}} \qquad s_{r,0} = \frac{\sum_{i=1}^{2^n} |\alpha_i|^2}{p_{q,0}}$$

and therefore $p_{r,0} \cdot s_{q,d} = p_{q,0} \cdot s_{r,0}$. Moreover, if $\mathcal{QV} = \emptyset$ then $\mathcal{M}_{q,0}(\mathcal{Q}_{r,0}) = \mathcal{M}_{r,0}(\mathcal{Q}_{q,0}) = 1$, otherwise:

$$\mathcal{M}_{q,0}(\mathcal{Q}_{r,0}) = \frac{\mathcal{Q}^{q,0}_{r,0}}{\sqrt{p_{q,0}}} = \frac{\sum_{i=1}^{2^n} \alpha_i b_i}{\sqrt{p_{r,0}} \cdot \sqrt{p_{q,0}}} = \frac{\sum_{i=1}^{2^n} \alpha_i b_i}{\sqrt{p_{r,0}} \cdot \sqrt{\frac{\sum_{i=1}^{2^n} |\alpha_i|^2}{p_{r,0}}}} = \frac{\sum_{i=1}^{2^n} \alpha_i b_i}{\sqrt{\sum_{i=1}^{2^n} |\alpha_i|^2}}$$

$$\mathcal{M}_{r,0}(\mathcal{Q}_{s,0}) = \frac{\mathcal{Q}^{r,0}_{q,0}}{\sqrt{p_{r,0}}} = \frac{\sum_{i=1}^{2^n} \alpha_i b_i}{\sqrt{p_{q,0}} \cdot \sqrt{p_{r,0}}} = \frac{\sum_{i=1}^{2^n} \alpha_i b_i}{\sqrt{p_{q,0}} \cdot \sqrt{\frac{\sum_{i=1}^{2^n} |\alpha_i|^2}{p_{q,0}}}} = \frac{\sum_{i=1}^{2^n} \alpha_i b_i}{\sqrt{\sum_{i=1}^{2^n} |\alpha_i|^2}}$$

and therefore $\mathcal{M}_{q,0}(\mathcal{Q}_{r,0}) = \mathcal{M}_{r,0}(\mathcal{Q}_{s,0})$.

7.2.1 Computations

The notion of configurations is exactly the same of Q and SQ.

Let $\mathscr{L} = \{\mathsf{Uq}, \mathsf{new}, \mathsf{l}.\beta, \mathsf{q}.\beta, \mathsf{c}.\beta, \mathsf{l.cm}, \mathsf{r.cm}, \mathsf{if}_1, \mathsf{if}_2, \mathsf{meas}_r\}$. For every $\alpha \in \mathscr{L}$ and for every $p \in \mathbb{R}_{[0,1]}$, we define a relation $\rightarrow^p_\alpha \subseteq \mathcal{C} \times \mathcal{C}$ by the set of *contractions* in Figure 7.2. The notation $C \rightarrow_\alpha D$ stands for $C \rightarrow^1_\alpha D$.

We adopt surface reduction [30,91] as for Q and SQ, and furthermore, we also forbid reduction in N and P in the term $\mathsf{if}\ M\ \mathsf{then}\ N\ \mathsf{else}\ P$.

We distinguish three particular subsets of $\mathscr{L}$, namely $\mathscr{K} = \{\mathsf{l.cm}, \mathsf{r.cm}\}$, $\mathscr{N} = \mathscr{L} - (\mathscr{K} \cup \{\mathsf{meas}_r\})$ and $n\mathscr{M} = \mathscr{L} - \{\mathsf{meas}_r\}$. In the following, we write $M \rightarrow_\alpha N$ meaning that there are $\mathcal{Q}$, $\mathcal{QV}$, $\mathcal{R}$ and $\mathcal{RV}$ such that $[\mathcal{Q}, \mathcal{QV}, M] \rightarrow_\alpha [\mathcal{R}, \mathcal{RV}, N]$. Similarly for the notation $M \rightarrow_{\mathscr{S}} N$ where $\mathscr{S}$ is a subset of $\mathscr{L}$.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Fig. 7.2. Contractions.

7.3 The Confluence Problem, an informal introduction

It is well known that the measurement-free evolution of a quantum system is deterministic. As a consequence it is to be expected that a good measurement-free quantum lambda calculus enjoys confluence. This is the case of Qand of the lambda calculus recently introduced by Arrighi and Dowek [9] (an algebraic lambda calculus inspired to quantum computing). The situation becomes more complicated if we introduce a measurement operator. In fact measurements break the deterministic evolution of a quantum system: in presence of measurements the behaviour becomes irremediably probabilistic. The confluence problem is central for any quantum λ-calculus with measurements, as stressed in the introduction.

Let us consider the following configuration:

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

If we focus on reduction sequences, it is easy to check that there are two different reduction sequences starting with C, the first ending in the normal form $[1, \emptyset, 0]$ (with probability $1/2$) and the second in the normal form $[1, \emptyset, 1]$ (with probability $1/2$). But if we reason with mixed states, the situation changes: the mixed state $\{1 : C\}$ (i.e., the mixed state assigning probability 1 to C and 0 to any other configuration) rewrites *deterministically* to $\{1/2 : [1, \emptyset, 0], 1/2 : [1, \emptyset, 1]\}$ (where both $[1, \emptyset, 0]$ and $[1, \emptyset, 1]$ have probability $1/2$). So, confluence seems to hold.

Confluence in Other Quantum Calculi.

Contrarily to the measurement-free case, the above notion of confluence is *not* an expected result for a quantum lambda calculus. Indeed, it does not hold in the quantum lambda calculus λ_{sv} proposed by Selinger and Valiron [87][4]. In λ_{sv}, it is possible to exhibit a configuration C that gives as outcome the distribution $\{1 : [1, \emptyset, 0]\}$ when reduced call-by-value and the distribution $\{1/2 : [1, \emptyset, 0], 1/2 : [1, \emptyset, 1]\}$ if reduced call-by-name. This is a *real* failure of confluence, which is there even if one uses probability distributions in place of configurations. The same phenomenon cannot happen in Q* (as we will show in Section 7.5): this fundamental difference can be traced back to another one: the linear lambda calculus with surface reduction (on which Q* is based) enjoys (a slight variation on) the so-called diamond property [91], while in usual, pure, lambda calculus (on which λ_{sv} is based) confluence only holds in a weaker sense.

Finite or infinite rewriting?

In Q*, an infinite computation can tend to a configuration which is essentially different from the configurations in the computation itself. For example, a configuration $C = [1, \emptyset, M]$ can be built[5] such that:

- after a finite number of reduction steps C rewrites to a distribution in the form $\{\sum_{1<i\leq n} \frac{1}{2^i} : [1, \emptyset, 0], 1 - \sum_{1<i\leq n} \frac{1}{2^i} : D\}$
- only after infinitely many reduction steps the distribution $\{1 : [1, \emptyset, 0]\}$ is reached.

Therefore finite probability distributions of finite configurations could be obtained by means of infinite rewriting. We believe that the study of confluence for infinite computations is important.

[4] As written in Chapter 3, the interest of Selinger and Valiron is for a quantum programming language. They are not interest in the confluence problem, but rather in the definition of the right reduction strategy

[5] $M \equiv (\mathsf{Y}!(\lambda !f.\lambda !x \mathsf{if}\ x\ \mathsf{then}\ 0\ \mathsf{else}\ f(\mathtt{meas}(H(\mathtt{new}(0))))))(\mathtt{meas}(H(\mathtt{new}(0))))$, where Y is a fix point operator.

Related Work.

In the literature, probabilistic rewriting systems have been already analyzed. For example, Bournez and Kirchner [24] have introduced the notion of a probabilistic abstract rewriting system as a structure $A = (|A|, [\cdot \rightsquigarrow \cdot])$ where $|A|$ is a set and $[\cdot \rightsquigarrow \cdot]$ is a function from $|A|$ to $\mathbb{R}$ such that for every $a \in |A|$, $\sum_{b \in |A|}[a \rightsquigarrow b]$ is either 0 or 1. Then, they define a notion of *probabilistic confluence* for a PARS: such a structure is probabilistically locally confluent iff the probability to be locally confluent, in a classical sense, is different from 0. Unfortunately, Bournez and Kirchner's analysis does not apply to Q*, since Q* is *not* a PARS. Indeed, the quantity $\sum_{b \in |A|}[a \rightsquigarrow b]$ can in general be any natural number. Similar considerations hold for the probabilistic lambda calculus introduced by Di Pierro, Hankin and Wiklicky in [37].

7.4 A Probabilistic Notion of Computation

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

Definition 7.7. *A set of (possibly) infinite trees $\mathscr{S}$ is said to be a* set of probabilistic computations *if $P \in \mathscr{S}$ iff (exactly) one of the following three conditions holds:*
1. $P = [C]$ *and* $C \in \mathscr{C}(,.)$
2. $P = [C, R]$, *where* $C \in \mathscr{C}(,,)$ $R \in \mathscr{S}$ *has root* D *and* $C \rightarrow_{n\mathscr{M}} D$
3. $P = [(p, q, C), R, Q]$, *where* $C \in \mathscr{C}(,,)$ $R, Q \in \mathscr{S}$ *have roots* D *and* E, $C \rightarrow^{p}_{meas_r} D$, $C \rightarrow^{q}_{meas_r} E$ *and* $p, q \in \mathbb{R}_{[0,1]}$;

The set of all (respectively, the set of finite) probabilistic computations is the largest set $\mathscr{P}$ (respectively, the smallest set $\mathscr{F}$) of probabilistic computations with respect to set inclusion. $\mathscr{P}$ and $\mathscr{F}$ exist because of the Knapster-Tarski Theorem.

We will often say that the root of $P = [(p, q, C), R, Q]$ is simply C, slightly diverging from the above definition without any danger of ambiguity.

Definition 7.8. *A probabilistic computation P is maximal if for every leaf C in P, $C \in$ NF. More formally, (sets of) maximal probabilistic computations can be defined as in Definition 7.7, where clause 1 must be restricted to $C \in$ NF.*

We can give definitions and proofs over *finite* probabilistic computations (i.e., over $\mathscr{F}$) by ordinary induction. An example is the following definition. Notice that the same is not true for arbitrary probabilistic definitions, since $\mathscr{P}$ is not a well-founded set.

Definition 7.9. *Let $P \in \mathscr{P}$ be a probabilistic computation. A finite probabilistic computation $R \in \mathscr{F}$ is a* sub-computation *of P, written $R \sqsubseteq P$ iff one of the following conditions is satisfied:*
- $R = [C]$ *and the root of* P *is* C.
- $R = [C, Q]$, $P = [C, S]$, *and* $Q \sqsubseteq S$.
- $R = [(p, q, C), Q, S]$, $P = [(p, q, C), U, V]$, $Q \sqsubseteq U$ *and* $S \sqsubseteq V$.

Let $\delta : \mathcal{C} \rightarrow \{0, 1\}$ be a function defined as follows: $\delta(C) = 0$ if the quantum register of C is 0, otherwise, $\delta(C) = 1$.

Quantitative Properties of Computations.

The outcomes of a probabilistic computation P are given by the configurations which appear as leaves of P. Starting from this observation, the following definitions formalize some quantitative properties of probabilistic computations. For every *finite* probabilistic computation P and every $C \in \mathsf{NF}$ we define $\mathcal{P}(P,C) \in \mathbb{R}_{[0,1]}$ by induction on the structure of P:

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

Similarly for $\mathcal{N}(P,C) \leq \aleph_0$:

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

Informally, $\mathcal{P}(P,C)$ is the probability of observing C as a leaf in P, and $\mathcal{N}(P,C)$ is the number of times C appears as a leaf in P.

The definitions above can be easily modified to get the probability of observing *any* configuration (in normal form) as a leaf in P, $\mathcal{P}(P)$, or the number of times *any* configuration appears as a leaf in P, $\mathcal{N}(P)$. Since $\mathbb{R}_{[0,1]}$ and $\mathbb{N} \cup \{\aleph_0\}$ are complete lattices (with respect to standard orderings), we extend the above notions to the case of *arbitrary* probabilistic computations, by taking the least upper bound over all finite sub-computations. If $P \in \mathscr{P}$ and $C \in \mathsf{NF}$, then:

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

For every *finite* probabilistic computation P and every $C \in \mathsf{NF}$ we define $\mathcal{P}(P,C) \in \mathbb{R}_{[0,1]}$ and $\mathcal{N}(P,C) \leq \aleph_0$ by induction on the structure of P:

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

More informally, $\mathcal{P}(P,C)$ is the probability of observing C as a leaf in P. On the other hand, $\mathcal{N}(P,C)$ is the number of times C appears as a leaf in P. The definitions above can be easily modified to get the probability of observing *any* configuration as a leaf in P, $\mathcal{P}(P)$, or the number of times *any* configuration appears as a leaf in P, $\mathcal{N}(P)$.

In turn, the functions $\mathcal{P}$ and $\mathcal{N}$ on finite probabilistic computations above can be generalized to functions on arbitrary probabilistic computations by taking the least upper bound over all finite sub-computations. For example, if $P \in \mathscr{P}$ and $C \in \mathsf{NF}$, then

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

Analogously,

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

Both quantities above exists because $\mathbb{R}_{[0,1]}$ and $\mathbb{N} \cup \{\aleph_0\}$ are complete lattices. The following lemmas involve finite computations and can be prove by induction.

Lemma 7.10. *If $P \sqsubseteq R$, then $\mathcal{P}(P) \leq \mathcal{P}(R)$ and $\mathcal{N}(P) \leq \mathcal{N}(R)$. Moreover, $\mathcal{P}(P, C) \leq \mathcal{P}(R, C)$ and $\mathcal{N}(P, C) \leq \mathcal{N}(R, C)$ for every $C \in \mathsf{NF}$.*

Proof. A trivial induction on P.

Lemma 7.11. *If $P \sqsubseteq R$ and P is maximal, then R is maximal and $P = R$.*

Proof. A trivial induction on P.

7.5 A Strong Confluence Result

In this Section, we will prove a strong confluence result in the following form: *any two maximal probabilistic computations P and R with the same root have exactly the same quantitative and qualitative behaviour*, that is to say, the following equations hold for every $C \in \mathsf{NF}$:

$$\begin{aligned} \mathcal{P}(P, C) &= \mathcal{P}(R, C); \\ \mathcal{N}(P, C) &= \mathcal{N}(R, C); \\ \mathcal{P}(P) &= \mathcal{P}(R); \\ \mathcal{N}(P) &= \mathcal{N}(R). \end{aligned}$$

Remark 7.12. Please notice that equalities like the ones above do *not* even hold for the ordinary lambda calculus. For example, the lambda term $(\lambda x.\lambda y.y)\Omega$ is the root of two (linear) computations, the first having one leaf $\lambda y.y$ and the second having no leaves. This is the reason why the confluence result we prove here is dubbed as strong.

Before embarking in the proof of the equalities above, let us spend a few words to explain their consequences. The fact $\mathcal{P}(P, C) = \mathcal{P}(R, C)$ whenever P and R have the same root can be read as a confluence result: the probability of observing C is independent from the adopted strategy. On the other hand, $\mathcal{P}(P) = \mathcal{P}(R)$ means that the probability of converging is not affected by the underlying strategy. The corresponding results on $\mathcal{N}(\cdot, \cdot)$ and $\mathcal{N}(\cdot)$ can be read as saying that the number of (not necessarily distinct) leaves in any probabilistic computation with root C does not depend on the strategy.

Lemma 7.13 (Uniformity). *For every M, N such that $M \to_\alpha N$, exactly one of the following conditions holds:*

1. *$\alpha \neq \mathsf{new}$ and $\alpha \neq \mathsf{meas}_r$ and there is a unitary transformation $U_{M,N} : \mathcal{H}(\mathbf{Q}(M)) \to \mathcal{H}(\mathbf{Q}(M))$ such that $[\mathcal{Q}, \mathcal{QV}, M] \to_\alpha [\mathcal{R}, \mathcal{RV}, N]$ iff $[\mathcal{Q}, \mathcal{QV}, M] \in \mathcal{C}$, $\mathcal{RV} = \mathcal{QV}$ and $\mathcal{R} = (U_{M,N} \otimes I_{\mathcal{QV} - \mathbf{Q}(M)})\mathcal{Q}$.*
2. *$\alpha = \mathsf{new}$ and there are a constant c and a quantum variable r such that $[\mathcal{Q}, \mathcal{QV}, M] \to_{\mathsf{new}} [\mathcal{R}, \mathcal{RV}, N]$ iff $[\mathcal{Q}, \mathcal{QV}, M] \in \mathcal{C}$, $\mathcal{RV} = \mathcal{QV} \cup \{r\}$ and $\mathcal{R} = \mathcal{Q} \otimes |r \mapsto c\rangle$.*
3. *$\alpha = \mathsf{meas}_r$ and there are a constant c and a probability $p_c \in \mathbb{R}_{[0,1]}$ such that $[\mathcal{Q}, \mathcal{QV}, M] \to^{p_c}_{\mathsf{meas}_r} [\mathcal{R}, \mathcal{RV}, N]$ iff $[\mathcal{Q}, \mathcal{QV}, M] \in \mathcal{C}$, $\mathcal{R} = \mathcal{M}_{r,c}(\mathcal{Q})$ and $\mathcal{RV} = \mathcal{QV} - \{r\}$.*

Proof. We go by induction on M. M cannot be a variable nor a constant nor a unitary operator nor a term $!L$. If M is an abstraction $\lambda\psi.L$, then $N \equiv \lambda\psi.P$, $L \rightarrow_\alpha P$ and the thesis follows from the inductive hypothesis. If M is $\mathtt{meas}(L)$ and N is $\mathtt{meas}(P)$ then $L \rightarrow_\alpha P$ and the thesis follows from the inductive hypothesis. Similarly if M is $\mathtt{new}(L)$ and N is $\mathtt{new}(P)$. And again if M is $\langle M_1, \ldots, L, \ldots, M_n\rangle$ and N is $\langle M_1, \ldots, P, \ldots, M_n\rangle$. If $M \equiv LQ$, then we distinguish a number of cases:

- $N \equiv PQ$ and $L \rightarrow_\alpha P$. The thesis follows from the inductive hypothesis.
- $N \equiv LS$ and $Q \rightarrow_\alpha S$. The thesis follows from the inductive hypothesis.
- $L \equiv U$, $Q \equiv \langle r_1, ..., r_n\rangle$ and $N \equiv \langle r_1, ..., r_n\rangle$. Then case 1 holds. In particular, $\mathbf{Q}(M) = \{r_1, ..., r_n\}$ and $U_{M,N} = U_{\langle\langle r_1,...,r_n\rangle\rangle}$.
- $L \equiv \lambda x.R$ and $N = R\{Q/x\}$. Then case 1 holds. In particular $U_{M,N} = I_{\mathbf{Q}(M)}$.
- $L \equiv \lambda\langle x_1, \ldots, x_n\rangle.R$, $Q = \langle r_1, \ldots, r_n\rangle$ and $N \equiv R\{r_1/x_1, \ldots, r_n/x_n\}$. Then case 1 holds and $U_{M,N} = I_{\mathbf{Q}(M)}$.
- $L \equiv \lambda !x.R$, $Q = !T$ and $N \equiv R\{T/x\}$. Then case 1 holds and $U_{M,N} = I_{\mathbf{Q}(M)}$.
- $Q \equiv (\lambda\pi.R)T$ and $N \equiv (\lambda\pi.LR)T$. Then case 1 holds and $U_{M,N} = I_{\mathbf{Q}(M)}$.
- $L \equiv (\lambda\pi.R)T$ and $N \equiv (\lambda\pi.RQ)T$. Then case 1 holds and $U_{M,N} = I_{\mathbf{Q}(M)}$.

If $M \equiv \mathtt{new}(c)$ then N is a quantum variable r and case 2 holds. If $M \equiv \mathtt{meas}(r)$ then there are a constant c and a probability p_c such that N is a term $!c$ and case 3 holds. This concludes the proof.

Notice that $U_{M,N}$ is always the identity function when performing classical reduction. The following technical lemma will be useful when proving confluence:

Lemma 7.14. *Suppose* $[\mathcal{Q}, \mathcal{QV}, M] \rightarrow_\alpha [\mathcal{R}, \mathcal{RV}, N]$.

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

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

We prove now that Q^* enjoys a slight variation of the so-called diamond property, whose proof is fully standard (it is a slight extension of the analogous proof given in [30] for Q). As for Q, Q^* does not enjoy the diamond property in a strict sense, due to the presence of commutative reduction rules (see, e.g., case 2 of the following Proposition). But thanks to Lemma 7.17 below, this does not have harmful consequences.

Proposition 7.15 (Quasi-One-step Confluence). *Let* C, D, E *be configurations with* $C \rightarrow^p_\alpha D$, $C \rightarrow^s_\beta E$. *Then:*

1. *If* $\alpha \in \mathscr{K}$ *and* $\beta \in \mathscr{K}$*, then either* $D = E$ *or there is* F *with* $D \rightarrow_{\mathscr{K}} F$ *and* $E \rightarrow_{\mathscr{K}} F$.

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

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

$$[\mathcal{Q}, \mathcal{QV}, N] \to_{\alpha} [\mathcal{R}, \mathcal{RV}, S]$$
$$[\mathcal{Q}, \mathcal{QV}, N] \to_{\beta} [\mathcal{S}, \mathcal{SV}, T]$$

The IH easily leads to the thesis. Similarly when $M \equiv \lambda!x.N$, and when $M \equiv \mathtt{meas}(N)$ or $M \equiv \mathsf{if}\ N\ \mathsf{then}\ P\ \mathsf{else}\ Q$ with $N \neq 0, 1$. If $M \equiv NL$, we can distinguish a number of cases depending on the last rule used to prove $C \to^{p}_{\alpha} D$, $C \to_{\beta} sE$:

- $D \equiv [\mathcal{R}, \mathcal{RV}, SL]$ and $E \equiv [\mathcal{S}, \mathcal{SV}, NR]$ where $[\mathcal{Q}, \mathcal{QV}, N] \to^{p}_{\alpha} [\mathcal{R}, \mathcal{RV}, S]$ and $[\mathcal{Q}, \mathcal{QV}, L] \to^{s}_{\beta} [\mathcal{S}, \mathcal{SV}, R]$. We need to distinguish several sub-cases:
  - If $\alpha, \beta = \mathsf{new}$, then, by Lemma 7.13, there exist two quantum variables $s, t \notin \mathcal{QV}$ and two constants d, e such that $\mathcal{RV} = \mathcal{QV} \cup \{s\}$, $\mathcal{SV} = \mathcal{QV} \cup \{t\}$, $\mathcal{R} = \mathcal{Q} \otimes |s \mapsto d\rangle$ and $\mathcal{S} = \mathcal{Q} \otimes |t \mapsto e\rangle$. Applying 7.13 again, we obtain

    $$D \to_{\mathsf{new}} [\mathcal{Q} \otimes |s \mapsto d\rangle \otimes |u \mapsto e\rangle, \mathcal{QV} \cup \{s, u\}, SR\{u/t\}] \equiv F;$$
    $$E \to_{\mathsf{new}} [\mathcal{Q} \otimes |t \mapsto e\rangle \otimes |v \mapsto d\rangle, \mathcal{QV} \cup \{t, v\}, S\{u/s\}R] \equiv G.$$

    As can be easily checked, $F \equiv G$.
  - If $\alpha = \mathsf{new}$ and $\beta \neq \mathsf{new}, \mathsf{meas}_r$, then, by Lemma 7.13 there exist a quantum variable r and a constant c such that $\mathcal{RV} = \mathcal{QV} \cup \{r\}$, $\mathcal{R} = \mathcal{Q} \otimes |r \mapsto c\rangle$, $\mathcal{SV} = \mathcal{QV}$ and $\mathcal{S} = (U_{L,R} \otimes I_{\mathcal{QV}-\mathbf{Q}(L)})\mathcal{Q}$. As a consequence, applying Lemma 7.13 again, we obtain

    $$D \to_{\beta} [(U_{L,R} \otimes I_{\mathcal{QV}\cup\{r\}-\mathbf{Q}(L)})(\mathcal{Q} \otimes |r \mapsto c\rangle), \mathcal{QV} \cup \{r\}, SR] \equiv F;$$
    $$E \to_{\mathsf{new}} [((U_{L,R} \otimes I_{\mathcal{QV}-\mathbf{Q}(L)})\mathcal{Q}) \otimes |r \mapsto c\rangle, \mathcal{QV} \cup \{r\}, SR] \equiv G.$$

    As can be easily checked, $F \equiv G$.
  - If $\alpha \neq \mathsf{new}, \mathsf{meas}_r$ and $\beta = \mathsf{new}$, then we can proceed as in the previous case.
  - If $\alpha, \beta \neq \mathsf{new}, \alpha \neq \mathsf{meas}_r, \beta \neq \mathsf{meas}_q$ (r, q not necessarily distinct) , then by Lemma 7.13, there exist $\mathcal{SV} = \mathcal{RV} = \mathcal{QV}$, $\mathcal{R} = (U_{N,S} \otimes I_{\mathcal{QV}-\mathbf{Q}(N)})\mathcal{Q}$ and $\mathcal{S} = (U_{L,R} \otimes I_{\mathcal{QV}-\mathbf{Q}(L)})\mathcal{Q}$. Applying 7.13 again, we obtain

    $$D \to_{\beta} [(U_{L,R} \otimes I_{\mathcal{QV}-\mathbf{Q}(L)})((U_{N,S} \otimes I_{\mathcal{QV}-\mathbf{Q}(N)})\mathcal{Q}), \mathcal{QV}, SR] \equiv F;$$
    $$E \to_{\alpha} [(U_{N,S} \otimes I_{\mathcal{QV}-\mathbf{Q}(L)})((U_{L,R} \otimes I_{\mathcal{QV}-\mathbf{Q}(L)})\mathcal{Q}), \mathcal{QV}, SR] \equiv G.$$

    As can be easily checked, $F \equiv G$.

- If $\alpha = \mathsf{meas}_r, \beta = \mathsf{meas}_q$ $(r \neq q)$ then, by Lemma 7.13, there exist two constants d, e and two probabilities t, u such that $\mathcal{RV} = \mathcal{QV} - \{r\}$, $\mathcal{SV} = \mathcal{QV} - \{q\}$, $\mathcal{R} = \mathcal{M}_{r,d}(\mathcal{Q})$ and $\mathcal{S} = \mathcal{M}_{q,e}(\mathcal{Q})$. Remember that the quantum variable q occurs in the subterm N and the quantum variable r occurs in the subterm L. Starting from $D \equiv [\mathcal{M}_{r,d}(\mathcal{Q}), \mathcal{QV} - \{r\}, SL]$ and $E \equiv [\mathcal{M}_{q,e}(\mathcal{Q}), \mathcal{QV} - \{q\}, NR]$, applying 7.13 again, we obtain

$$\begin{aligned} D \rightarrow^{\bar{s}}_{\mathsf{meas}_q} & [\mathcal{M}_{q,e}(\mathcal{M}_{r,d}(\mathcal{Q})), \mathcal{QV} - \{r\} - \{q\}, SR] \\ \equiv \quad & [\mathcal{M}_{q,e}(\mathcal{R}), \mathcal{RV} - \{q\}, SR] \equiv F; \\ E \rightarrow^{\bar{p}}_{\mathsf{meas}_r} & [\mathcal{M}_{r,d}(\mathcal{M}_{q,e}(\mathcal{Q})), \mathcal{QV} - \{q\} - \{r\}, SR] \\ \equiv \quad & [\mathcal{M}_{r,d}(\mathcal{S}), \mathcal{SV} - \{r\}, SR] \equiv G. \end{aligned}$$

  Clearly, $\mathcal{QV} - \{r\} - \{q\} \equiv \mathcal{QV} - \{q\} - \{r\}$ and by Proposition 7.6, case 4, $\mathcal{M}_{q,e}(\mathcal{M}_{r,d}(\mathcal{Q})) \equiv \mathcal{M}_{r,d}(\mathcal{M}_{q,e}(\mathcal{Q}))$. Then $F \equiv G$. Moreover by Proposition 7.6, case 3, $pt = su$.
- If $\alpha = \mathsf{new}, \beta = \mathsf{meas}_r$, then, by Lemma 7.13 there exists a quantum variable q $(q \neq r)$ two constants d and e and a probability p_e such that $\mathcal{RV} = \mathcal{QV} \cup \{q\}$, $\mathcal{R} = \mathcal{Q} \otimes |q \mapsto d\rangle, \mathcal{SV} = \mathcal{QV} - \{r\}$ and $\mathcal{S} = \mathcal{M}_{r,e}(\mathcal{Q})$. As a consequence, starting from $D \equiv [\mathcal{QV} \cup \{q\}, \mathcal{Q} \otimes |q \mapsto d\rangle, SL]$ and $E \equiv [\mathcal{M}_{r,e}(\mathcal{Q}), \mathcal{QV} - \{r\}, NR]$ applying Lemma 7.13 again, we obtain

$$\begin{aligned} D \rightarrow^{p_e}_{\mathsf{meas}_r} & [\mathcal{M}_{r,e}(\mathcal{Q} \otimes |q \mapsto d\rangle), \mathcal{QV} \cup \{q\} - \{r\}, SR] \\ \equiv \quad & [\mathcal{M}_{r,e}(\mathcal{R}), \mathcal{QV} \cup \{q\} - \{r\}, SR] \equiv F; \\ E \ \rightarrow_{\mathsf{new}} & [(\mathcal{M}_{r,e}(\mathcal{Q})) \otimes |q \mapsto d\rangle, \mathcal{QV} - \{r\} \cup \{q\}, SR] \\ \equiv \quad & [(\mathcal{S}) \otimes |q \mapsto d\rangle, \mathcal{SV} \cup \{q\}, SR] \equiv G. \end{aligned}$$

  Clearly, $\mathcal{QV} \cup \{q\} - \{r\} \equiv \mathcal{QV} - \{r\} \cup \{q\}$. By Proposition 7.6, case 2, it is possible to commute the measurement of the quantum variable r with the creation of the quantum variable q, in fact they are distinct quantum variable. Then $\mathcal{M}_{r,e}(\mathcal{Q} \otimes |q \mapsto d\rangle)$ and $(\mathcal{M}_{r,e}(\mathcal{Q})) \otimes |q \mapsto d\rangle$ give the same quantum register. We can conclude that $F \equiv G$.
- If $\alpha = \mathsf{meas}_r, \beta = \mathsf{new}$, the case is symmetric to the previous one.
- If $\alpha = \mathsf{meas}_r, \beta \neq \mathsf{new}, \mathsf{meas}_q$, then by Lemma 7.13 there exist a constant c and a probability p_c such that $\mathcal{R} = \mathcal{M}_{r,c}(\mathcal{Q})$, $\mathcal{RV} = \mathcal{QV} - \{r\}$, $\mathcal{SV} = \mathcal{QV}$ and $\mathcal{S} = (U_{L,R} \otimes I_{\mathcal{QV} - \mathbf{Q}(L)})\mathcal{Q}$. As a consequence, starting from $D \equiv [\mathcal{M}_{r,c}(\mathcal{Q}), \mathcal{QV} - \{r\}, SL]$ and $E \equiv [(U_{L,R} \otimes I_{\mathcal{QV} - \mathbf{Q}(L)})\mathcal{Q}, \mathcal{QV}, NR]$, applying Lemma 7.13 again, we obtain

$$\begin{aligned} D \quad \rightarrow_{\beta} \quad & [(U_{L,R} \otimes I_{\mathcal{QV} - \{r\} - \mathbf{Q}(L)})(\mathcal{M}_{r,c}(\mathcal{Q})), \mathcal{QV} - \{r\}, SR] \\ \equiv \quad & [(U_{L,R} \otimes I_{\mathcal{QV} - \{r\} - \mathbf{Q}(L)})(\mathcal{R}), \mathcal{RV}, SR] \equiv F \\ E \rightarrow^{p_c}_{\mathsf{meas}_r} & [\mathcal{M}_{r,c}((U_{L,R} \otimes I_{\mathcal{QV} - \mathbf{Q}(L)})\mathcal{Q}), \mathcal{QV} - \{r\}, SR] \\ \equiv \quad & [\mathcal{M}_{r,c}(\mathcal{S}), \mathcal{QV} - \{r\}, SR] \equiv G \end{aligned}$$

  Note that the operators $(U_{L,R} \otimes I_{\mathcal{QV} - \{r\} - \mathbf{Q}(L)}) \circ \mathcal{M}_{r,c}$ and $\mathcal{M}_{r,c} \circ (U_{L,R} \otimes I_{\mathcal{QV} - \mathbf{Q}(L)})$ act on $\mathcal{Q}$ in the same way, by means of Proposition 7.6, case 5. We can conclude that $F \equiv G$.
- If $\alpha \neq \mathsf{new}, \mathsf{meas}_q, \beta = \mathsf{meas}_r$, the case is symmetric to the previous one.

- $D \equiv [\mathcal{R}, \mathcal{RV}, SL]$ and $E \equiv [\mathcal{S}, \mathcal{SV}, TL]$, where $[\mathcal{Q}, QV, N] \to [\mathcal{R}, \mathcal{RV}, S]$ and $[\mathcal{Q}, \mathcal{QV}, N] \to [\mathcal{S}, \mathcal{SV}, T]$. Here we can apply the inductive hypothesis.
- $D \equiv [\mathcal{R}, \mathcal{RV}, NR]$ and $E \equiv [\mathcal{S}, \mathcal{SV}, NU]$, where $[\mathcal{Q}, QV, L] \to [\mathcal{R}, \mathcal{RV}, R]$ and $[\mathcal{Q}, \mathcal{QV}, L] \to [\mathcal{S}, \mathcal{SV}, U]$. Here we can apply the inductive hypothesis as well.
- $N \equiv (\lambda x.P)$, $D \equiv [\mathcal{Q}, \mathcal{QV}, P\{L/x\}]$, $E \equiv [\mathcal{R}, \mathcal{RV}, NR]$, where $[\mathcal{Q}, \mathcal{QV}, L] \to_\beta [\mathcal{R}, \mathcal{RV}, R]$. Clearly $[\mathcal{Q}, \mathcal{QV}, P\{L/x\}] \in \mathcal{C}$ and, by Lemma 7.14, $[\mathcal{Q}, \mathcal{QV}, P\{L/x\}] \to [\mathcal{R}, \mathcal{RV}, P\{R/x\}]$. Moreover, $[\mathcal{R}, \mathcal{RV}, NR] \equiv [\mathcal{R}, \mathcal{RV}, (\lambda x.P)R] \to [\mathcal{R}, \mathcal{RV}, P\{R/x\}]$.
- $N \equiv (\lambda x.P)$, $D \equiv [\mathcal{Q}, \mathcal{QV}, P\{L/x\}]$, $E \equiv [\mathcal{R}, \mathcal{RV}, (\lambda x.V)L]$, where $[\mathcal{Q}, \mathcal{QV}, P] \to_\beta [\mathcal{R}, \mathcal{RV}, V]$. Clearly $[\mathcal{Q}, \mathcal{QV}, P\{L/x\}] \in \mathcal{C}$ and, by Lemma 7.14, $[\mathcal{Q}, \mathcal{QV}, P\{L/x\}] \to_\beta [\mathcal{R}, \mathcal{RV}, V\{L/x\}]$. Moreover, $[\mathcal{R}, \mathcal{RV}, (\lambda x.V)L] \to_\beta [\mathcal{R}, \mathcal{RV}, V\{L/x\}]$.
- $N \equiv (\lambda !x.P)$, $L \equiv !Q$, $D \equiv [\mathcal{Q}, \mathcal{QV}, P\{Q/x\}]$, $E \equiv [\mathcal{R}, \mathcal{RV}, (\lambda !x.V)L]$, where $[\mathcal{Q}, \mathcal{QV}, P] \to_\beta [\mathcal{R}, \mathcal{RV}, V]$. Clearly $[\mathcal{Q}, \mathcal{QV}, P\{Q/x\}] \in \mathcal{C}$ and, by Lemma 7.14, $[\mathcal{Q}, \mathcal{QV}, P\{Q/x\}] \to_\beta [\mathcal{R}, \mathcal{RV}, V\{Q/x\}]$. Moreover, $[\mathcal{R}, \mathcal{RV}, (\lambda x.V)!Q] \to_\beta [\mathcal{R}, \mathcal{RV}, V\{Q/x\}]$.
- $N \equiv (\lambda\langle x_1, \ldots, x_n\rangle.P)$, $L \equiv \langle r_1, \ldots, r_n\rangle$, $D \equiv [\mathcal{Q}, \mathcal{QV}, P\{r_1/x_1, \ldots, r_n/x_n\}]$, $E \equiv [\mathcal{R}, \mathcal{RV}, (\lambda\langle x_1, \ldots, x_n\rangle.V)L]$, where $[\mathcal{Q}, \mathcal{QV}, P] \to_\beta [\mathcal{R}, \mathcal{RV}, V]$. Clearly $[\mathcal{Q}, \mathcal{QV}, P\{r_1/x_1, \ldots, r_n/x_n\}] \in \mathcal{C}$ and, by Lemma 7.14, $[\mathcal{Q}, \mathcal{QV}, P\{r_1/x_1, \ldots, r_n/x_n\}] \to_\beta [\mathcal{R}, \mathcal{RV}, V\{r_1/x_1, \ldots, r_n/x_n\}]$. Moreover, $[\mathcal{R}, \mathcal{RV}, (\lambda\langle x_1, \ldots, x_n\rangle.V)L] \to_\beta [\mathcal{R}, \mathcal{RV}, V\{r_1/x_1, \ldots, r_n/x_n\}]$.
- $N \equiv (\lambda x.P)Q$, $D \equiv [\mathcal{Q}, \mathcal{QV}, (\lambda x.PL)Q]$, $E \equiv [\mathcal{Q}, \mathcal{QV}, (P\{Q/x\})L]$, $\alpha = \mathsf{r.cm}$, $\beta = \mathsf{l}.\beta$. Clearly, $[\mathcal{Q}, \mathcal{QV}, (\lambda x.PL)Q] \to_{\mathsf{l}.\beta} [\mathcal{Q}, \mathcal{QV}, (P\{Q/x\})L]$.
- $N \equiv (\lambda\pi.P)Q$, $D \equiv [\mathcal{Q}, \mathcal{QV}, (\lambda\pi.PL)Q]$, $E \equiv [\mathcal{R}, \mathcal{RV}, ((\lambda\pi.V)Q)L]$, $\alpha = \mathsf{r.cm}$, where $[\mathcal{Q}, \mathcal{QV}, P] \to_\beta [\mathcal{R}, \mathcal{RV}, V]$. Clearly, $[\mathcal{Q}, \mathcal{QV}, (\lambda x.PL)Q] \to_{\mathsf{r.cm}} [\mathcal{R}, \mathcal{RV}, (\lambda x.VL)Q]$ and $[\mathcal{R}, \mathcal{RV}, ((\lambda\pi.V)Q)L] \to_\beta [\mathcal{R}, \mathcal{RV}, (\lambda\pi.VL)Q]$.
- $N \equiv (\lambda\pi.P)Q$, $D \equiv [\mathcal{Q}, \mathcal{QV}, (\lambda x.PL)Q]$, $E \equiv [\mathcal{R}, \mathcal{RV}, ((\lambda\pi.P)W)L]$, $\alpha = \mathsf{r.cm}$, where $[\mathcal{Q}, \mathcal{QV}, Q] \to_\beta [\mathcal{R}, \mathcal{RV}, W]$. Clearly, $[\mathcal{Q}, \mathcal{QV}, (\lambda x.PL)Q] \to_{\mathsf{r.cm}} [\mathcal{R}, \mathcal{RV}, (\lambda x.PL)W]$ and $[\mathcal{R}, \mathcal{RV}, ((\lambda\pi.P)W)L] \to_\beta [\mathcal{R}, \mathcal{RV}, (\lambda\pi.PL)W]$.
- $N \equiv (\lambda\pi.P)Q$, $D \equiv [\mathcal{Q}, \mathcal{QV}, (\lambda x.PL)Q]$, $E \equiv [\mathcal{R}, \mathcal{RV}, ((\lambda\pi.P)Q)R]$, $\alpha = \mathsf{r.cm}$, where $[\mathcal{Q}, \mathcal{QV}, L] \to_\beta [\mathcal{R}, \mathcal{RV}, R]$. Clearly, $[\mathcal{Q}, \mathcal{QV}, (\lambda x.PL)Q] \to_{\mathsf{r.cm}} [\mathcal{R}, \mathcal{RV}, (\lambda x.PR)Q]$ and $[\mathcal{R}, \mathcal{RV}, ((\lambda\pi.P)Q)R] \to_\beta [\mathcal{R}, \mathcal{RV}, (\lambda\pi.PR)Q]$.
- $N \equiv (\lambda\pi.P)$, $L \equiv (\lambda x.Q)R$, $D \equiv [\mathcal{Q}, \mathcal{QV}, (\lambda x.NQ)R]$, $E \equiv [\mathcal{Q}, \mathcal{QV}, N(Q\{R/x\})]$, $\alpha = \mathsf{l.cm}$, $\beta = \mathsf{l}.\beta$. Clearly, $[\mathcal{Q}, \mathcal{QV}, (\lambda x.NQ)R] \to \mathsf{l}.\beta[\mathcal{Q}, \mathcal{QV}, N(Q\{R/x\})]$.

If M is in the form $\mathtt{new}(c)$, then $D \equiv E$.

Remark 7.16. Unfortunately, Proposition 7.15 does not translate into an equivalent result on mixed states, because of commutative reduction rules. As a consequence, it is more convenient to first study confluence at the level of probabilistic computations.

Note that, even if the calculus is untyped, we cannot build an infinite sequence of commuting reductions:

Lemma 7.17. *The relation $\to_{\mathscr{K}}$ is strongly normalizing. In other words, there cannot be any infinite sequence $C_1 \to_{\mathscr{K}} C_2 \to_{\mathscr{K}} C_3 \to_{\mathscr{K}} \ldots$.*

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

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

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

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

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

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

Now we propose a probabilistic variation on the classical *strip lemma* of the λ-calculus. It will have a crucial rôle in the proof of strong confluence (Theorem 7.20).

Lemma 7.18 (Probabilistic Strip Lemma). *Let P be a finite probabilistic computation with root C and positive weight $\mathsf{W}(P)$.*

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

Proof. By induction on the structure of P:

- P cannot simply be $[C]$, because $\mathsf{W}(P) \geq 1$.
- If $P = [C,S]$, where S has root F and $C \to_{\mathscr{N}} F$, then:
  - Suppose $C \to_{\mathscr{N}} D$. If $D = F$, then the required R is simply S. Otherwise, by Proposition 7.15, there is G such that $D \to_{\mathscr{N}} G$ and $F \to_{\mathscr{N}} G$. Now, if S is simply $[F]$, then the required probabilistic computation is simply $[D]$, because neither F nor D are in normal form and, moreover, $\mathsf{W}([D]) = 0 < 1 = \mathsf{W}(P)$. If, on the other hand, S has positive weight we can apply the IH to it, obtaining a probabilistic computation T with root G such that $\mathsf{W}(T) < \mathsf{W}(S)$, $\mathsf{B}(T) \leq \mathsf{B}(S)$, $\mathcal{P}(T,H) \geq \mathcal{P}(S,H)$ and $\mathcal{N}(T,H) \geq \mathcal{N}(S,H)$ for every $H \in \mathsf{NF}$. Then, the required probabilistic computation is $[D,T]$, since

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

- Suppose $C \rightarrow_{\mathscr{K}} D$. By Proposition 7.15 one of the following two cases applies:
  - There is G such that $D \rightarrow_{\mathscr{N}} G$ and $F \rightarrow_{\mathscr{K}} G$ Now, if S is simply $[F]$, then the required probabilistic computation is simply $[D,[G]]$, because $\mathsf{W}([D,[G]]) = 1 = \mathsf{W}(P)$. If, on the other hand, S has positive weight we can apply the IH to it, obtaining a probabilistic computation T with root G such that $\mathsf{W}(T) \leq \mathsf{W}(S)$, $\mathsf{B}(T) \leq \mathsf{B}(S)$ and $\mathcal{P}(T,H) \geq \mathcal{P}(T,H)$ for every $H \in \mathsf{NF}$. Then, the required probabilistic computation is $[D,T]$, since

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

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

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

- Suppose $C \rightarrow^{q}_{\mathsf{meas}_r} D$ and $C \rightarrow^{p}_{\mathsf{meas}_r} E$. By Proposition 7.15, there are G and H such that $D \rightarrow_{\mathscr{N}} G$, $E \rightarrow_{\mathscr{N}} H$, $F \rightarrow^{q}_{\mathsf{meas}_r} G$, $F \rightarrow^{p}_{\mathsf{meas}_r} H$. Now, if S is simply F, then the required probabilistic computations are simply $[D]$ and $[E]$, because neither F nor D nor E are in normal form and, moreover, $\mathsf{W}([D]) = \mathsf{W}([E]) = 0 < 1 = \mathsf{W}(P)$. If, on the other hand, S has positive weight we can apply the IH to it, obtaining probabilistic computations T and U with roots G and H such that $\mathsf{W}(T) < \mathsf{W}(S)$, $\mathsf{W}(U) < \mathsf{W}(S)$, $\mathsf{B}(T) \leq \mathsf{B}(S)$, $\mathsf{B}(U) \leq \mathsf{B}(S)$, $q\mathcal{P}(T,H) + (p)\mathcal{P}(U,H) \geq \mathcal{P}(S,H)$ and $\mathcal{N}(T,H) + \mathcal{N}(U,H) \geq \mathcal{N}(S,H)$ for every $H \in \mathsf{NF}$. Then, the required probabilistic computations are $[D,T]$ and $[E,U]$, since

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

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

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

- The other cases are similar.

The following Proposition follows from the probabilistic strip lemma. It can be read as a simulation result: if P and R are maximal and have the same root, then P can simulate R (and viceversa).

Proposition 7.19. *For every maximal probabilistic computations P and for every finite probabilistic computation R such that P and R have the same root, there is a finite sub-computation Q of P such that for every $C \in \mathsf{NF}$, $\mathcal{P}(Q,C) \geq \mathcal{P}(R,C)$ and $\mathcal{N}(Q,C) \geq \mathcal{N}(R,C)$. Moreover, $\mathcal{P}(Q) \geq \mathcal{P}(R)$ and $\mathcal{N}(Q) \geq \mathcal{N}(R)$.*

Proof. Given any probabilistic computation S, its $\mathscr{K}$-degree n_S is the number of consecutive commutative rules you find descending S, starting at the root. By Lemma 7.17, this is a good definition. The proof goes by induction on $(\mathsf{W}(R), n_R)$, ordered lexicographically:

- If $\mathsf{W}(R) = 0$, then R is just $[D]$ for some configuration D. Then, $Q = R$ and all the required conditions hold.
- If $\mathsf{W}(R) > 0$, then we distinguish three cases, depending on the shape of P:
  - If $P = [D,S]$, E is the root of S and $D \rightarrow_{\mathscr{N}} E$, then, by Proposition 7.18, there is a probabilistic computation T with root E such that $\mathsf{W}(T) < \mathsf{W}(R)$ and $\mathcal{P}(T,C) \geq \mathcal{P}(R,C)$ for every $C \in \mathsf{NF}$. By the inductive hypothesis applied to S and T, there is a sub-probabilistic computation U of S such that $\mathcal{P}(U,C) \geq \mathcal{P}(T,C)$ and $\mathcal{N}(U,C) \geq \mathcal{N}(T,C)$ for every $C \in \mathsf{NF}$. Now, consider the probabilistic computation $[D,U]$. This is clearly a sub-probabilistic computation of P. Moreover, for every $C \in \mathsf{NF}$:

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

  - If $P = [D,S]$, E is the root of S and $D \rightarrow_{\mathscr{K}} E$, then, by Proposition 7.18, there is a probabilistic computation T with root E such that $\mathsf{W}(T) \leq \mathsf{W}(R)$ and $\mathcal{P}(T,C) \geq \mathcal{P}(R,C)$ for every $C \in \mathsf{NF}$. Now, observe we can apply the inductive hypothesis to S and T, because $\mathsf{W}(T) \leq \mathsf{W}(R)$ and $n_S < n_P$. So, there is a sub-probabilistic computation U of S such that $\mathcal{P}(U,C) \geq \mathcal{P}(T,C)$ and $\mathcal{N}(U,C) \geq \mathcal{N}(T,C)$ for every $C \in \mathsf{NF}$. Now, consider the probabilistic computation $[D,U]$. This is clearly a sub-probabilistic computation of P. Moreover, for every $C \in \mathsf{NF}$:

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

  - $P = [(p,q,D),S_1,S_2]$, E_1 is the root of S_1 and E_2 is the root of S_2, then, by Proposition 7.18, there are probabilistic computations T_1 and T_2 with root E_1 and E_2 such that $\mathsf{W}(T_1), \mathsf{W}(T_2) < \mathsf{W}(R)$ and $p\mathcal{P}(T_1,C)+q\mathcal{P}(T_2,C) \geq \mathcal{P}(R,C)$ for every $C \in \mathsf{NF}$. By the inductive hypothesis applied to S_1 and T_1 (to S_2 and T_2, respectively), there is a sub-probabilistic computation U_1 of S_1 (a sub-probabilistic computation U_2 of S_2, respectively) such that $\mathcal{P}(U_1,C) \geq \mathcal{P}(T_1,C)$ for every $C \in \mathsf{NF}$ ($\mathcal{P}(U_2,C) \geq \mathcal{P}(T_2,C)$ for every $C \in \mathsf{NF}$, respectively). Now, consider the probabilistic computation $[(p,q,D),U_1,U_2]$. This is clearly a sub-probabilistic computation of P. Moreover, for every $C \in \mathsf{NF}$:

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

This concludes the proof.

The main theorem is the following:

Theorem 7.20 (Strong Confluence). *For every maximal probabilistic computation P, for every maximal probabilistic computation R such that P and R have the same root, and for every $C \in \mathsf{NF}$, $\mathcal{P}(P,C) = \mathcal{P}(R,C)$ and $\mathcal{N}(P,C) = \mathcal{N}(R,C)$. Moreover, $\mathcal{P}(P) = \mathcal{P}(R)$ and $\mathcal{N}(P) = \mathcal{N}(R)$.*

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

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

Now, consider the two sets $A = \{\mathcal{P}(Q,C)\}_{Q\sqsubseteq P}$ and $B = \{\mathcal{P}(S,C)\}_{S\sqsubseteq R}$. We claim the two sets have the same upper bounds. Indeed, if $x \in \mathbb{R}$ is an upper bound on A and $S \sqsubseteq R$, by Proposition 7.19 there is $Q \sqsubseteq P$ such that $\mathcal{P}(Q,C) \geq \mathcal{P}(S,C)$, and so $x \geq \mathcal{P}(S,C)$. As a consequence, x is an upper bound on B. Symmetrically, if x is an upper bound on B, it is an upper bound on A. Since A and B have the same upper bounds, they have the same least upper bound, and $\mathcal{P}(P,C) = \mathcal{P}(R,C)$. The other claims can be proved exactly in the same way. This concludes the proof.

7.6 Computing with Mixed States

Definition 7.21 (Mixed State). *A mixed state is a function $\mathscr{M} : \mathcal{C} \to \mathbb{R}_{[0,1]}$ such that there is a finite set $S \subseteq \mathscr{C}$(,w)ith $\mathscr{M}(C) = 0$ except when $C \in S$ and, moreover, $\sum_{C\in S}\mathscr{M}(C) = 1$.* **Mix** *is the set of mixed states.*

In this thesis, a mixed state $\mathscr{M}$ will be denoted with the linear notation $\{p_1 : C_1,\ldots,p_k : C_k\}$ or as $\{p_i : C_i\}_{1\leq i\leq k}$, where p_i is the probability $\mathscr{M}(C_i)$ associated to the configuration C_i.

Definition 7.22 (Reduction). *The reduction relation $\Longmapsto$ between mixed states is defined in the following way: $\{p_1 : C_1,\ldots,p_m : C_m\} \Longmapsto \mathscr{M}$ iff there exist m mixed states $\mathscr{M}_1 = \{q_1^i : D_1^i\}_{1\leq i\in n_1},\ldots,\mathscr{M}_m = \{q_m^i : D_m^i\}_{1\leq i\leq n_m}$ such that:*

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

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

Given a mixed state $\mathscr{M}$ and a configuration $C \in \mathsf{NF}$, the *probability* of observing C in $\mathscr{M}$ is defined as $\mathscr{M}(C)$ and is denoted as $\mathcal{P}(\mathscr{M},C)$. Observe that if $\mathscr{M} \Longmapsto \mathscr{M}'$ and $C \in \mathsf{NF}$, then $\mathcal{P}(\mathscr{M},C) \leq \mathcal{P}(\mathscr{M}',C)$. If $\{\mathscr{M}_i\}_{i<\varphi}$ is a mixed computation, then

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

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

Please notice that a maximal mixed computation is always infinite. Indeed, if $\mathscr{M} = \{p_i : C_i\}_{1\leq i\leq n}$ and for every $i \in [1, n], C_i \in \mathsf{NF}$, then $\mathscr{M} \Longmapsto \mathscr{M}$.

Proposition 7.23. *Let $\{\mathscr{M}_i\}_{i<\omega}$ be a maximal mixed computation and let $C_1, \ldots, C_n$ be the configurations on which $\mathscr{M}_0$ evaluates to a positive real. Then there are maximal probabilistic computations $P_1, \ldots, P_n$ with roots $C_1, \ldots, C_n$ such that $\sup_{j<\varphi} \mathscr{M}_j(D) = \sum_{i=1}^{n} (\mathscr{M}_0(C_i)\mathcal{P}(P_i, D))$ for every D.*

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

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

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

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

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

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

This concludes the proof.

Theorem 7.24. *For any two maximal mixed computations $\{\mathscr{M}_i\}_{i<\omega}$ and $\{\mathscr{M}'_i\}_{i<\omega}$ such that $\mathscr{M}_0 = \mathscr{M}'_0$, the following condition holds: for every $C \in \mathsf{NF}$, $\mathcal{P}(\{\mathscr{M}_i\}_{i<\omega}, C) = \mathcal{P}(\{\mathscr{M}'_i\}_{i<\omega}, C)$*

Proof. A trivial consequence of Proposition 7.23.

Part III

A Variation on the Theme

8

Modal Labeled Deduction Systems for Quantum State Transformations

In this chapter we propose the formalization of a modal natural deduction system [79, 95] called MSQS, in order to represent (in an abstract, qualitative, way) the fundamental operations on quantum states: unitary transformations and total measurements. Unitary transformations model the internal evolution of a quantum system, whereas measurements correspond to the results of the interaction between a quantum system and an observer. The outcome of an observation can be either the reduction to some quantum state or the reduction to a classical state, where we say that a state w is *classical* iff w is invariant with respect to measurement, i.e. each measurement of w has w as outcome. We call a measurement *total* when the outcome of the measurement is a classical state. Note that we are not interested in the structure of quantum states, but rather in the way quantum states are transformed. Hence, we will abstract away from the internals of quantum states and we represent them in a generic way in order to describe how operations transform a state into another one. We propose to model measurement and unitary transformations by means of suitable modal operators, we give a suitable Kripke style semantics and we prove that MSQS is sound and complete with respect to it. Then we also study normalization properties, a subformula property, and as a consequence we show that is possible to give a purely syntactical proof of consistency for the system.

A variant of MSQS, called MSpQS, is also defined: in MSpQS we represent the case of generic, not necessary total measurements, and we prove the same result as for MSQS.

8.1 Modal Logic, Quantum Logic and Quantum Computing

Modal logic, as a logic of possible worlds, is a natural way to represent descriptions of a quantum system: the worlds model the quantum states and the relations of accessibility between worlds model the dynamical behavior of the system, as a consequence of the application of measurements and unitary transformations. To emphasize this semantic view of modal logic, we give our deduction system in the style of *labeled deduction* [42, 90, 98], a framework for giving uniform presentations of different non-classical logics (see also Section 2.3.1).

It is important to observe that our logical systems are not a quantum logic. Since the work of Birkhoff and von Neumann in 1936 [23], various logics have been investigated as a means to formalize reasoning about propositions taking into account the principles of quantum theory, e.g. [31, 32]. In general, it is possible to view quantum logic as a logical axiomatization of quantum theory [4, 14, 15, 70].

Our proposal moves from quite a different point of view: we do not aim to present a general logical formalization of quantum theory, rather we describe how it is possible to use modal logic to reason in a simple way about quantum state transformations. Informally, in our proposal, a modal world represents (an abstraction of) a quantum state. The discrete temporal evolution of a quantum state is controlled and determined by a sequence of unitary transformations and measurements that can change the description of a quantum state into other descriptions. So, the evolution of a quantum state can be viewed as a graph, where the nodes are the (abstract) quantum states and the arrows represent quantum transformations. The arrows give us the so-called accessibility relations of Kripke models and two nodes linked by an arrow represent two related quantum states: the target node is obtained from the source node by means of the operation specified in the decoration of the arrow.

8.2 The deduction system MSQS

8.2.1 The language of MSQS

Our labeled modal natural deduction system MSQS, which formally represents unitary transformations and total measurements of quantum states, consists of rules that derive formulas of two kinds: modal formulas and relational formulas. We thus define a modal language and a relational language.

The alphabet of the *relational language* consists of:

- the binary symbols U and M,
- a denumerable set $x_0, x_1, \ldots$ of *labels*.

Metavariables x, y, z, possibly annotated with subscripts and superscripts, range over the set of labels. For brevity, we will sometimes speak of a "world" x meaning that the label x stands for a world $\mathscr{I}(x)$, where $\mathscr{I}$ is an interpretation function mapping labels into worlds as formalized in Definition 8.2 below.

The set of *relational formulas* (*r-formulas*) is given by expressions of the form $x\mathsf{U}y$ and $x\mathsf{M}y$. We write $x\mathsf{R}y$ to denote a generic r-formula, with $\mathsf{R} \in \{\mathsf{U}, \mathsf{M}\}$.

The alphabet of the *modal language* consists of:

- a denumerable set $r, r_0, r_1, \ldots$ of *propositional symbols*,
- the standard *propositional connectives* $\perp$ and $\supset$,
- the unary *modal operators* $\square$ and $\blacksquare$.

The set of *modal formulas* (*m-formulas*) is the least set that contains $\perp$ and the propositional symbols, and is closed under $\supset$ and the modal operators. Metavariables A, B, C, possibly indexed, range over modal formulas. Other connectives can be defined in the usual manner, e.g. $\neg A \equiv A \supset \perp$, $A \wedge B \equiv \neg(A \supset \neg B)$, $A \leftrightarrow B \equiv (A \supset B) \wedge (B \supset A)$, $\Diamond A \equiv \neg\square\neg A$, $\blacklozenge A \equiv \neg\blacksquare\neg A$, etc.

Let us give, in a rather informal way, the intuitive meaning of the modal operators of our language:

$$
\dfrac{\begin{matrix}[x:A]\\ \vdots \\ x:B\end{matrix}}{x:A\supset B}\supset I
\qquad
\dfrac{x:A\supset B \quad x:A}{x:B}\supset E
\qquad
\dfrac{\begin{matrix}[x:\neg A]\\ \vdots \\ y:\bot\end{matrix}}{x:A}\,RAA
\qquad
\dfrac{x:\bot}{\alpha}\,\bot E
$$

$$
\dfrac{\begin{matrix}[x\mathsf{R}y]\\ \vdots \\ y:A\end{matrix}}{x:\bigstar A}\,\bigstar I
\qquad
\dfrac{x:\bigstar A \quad x\mathsf{R}y}{y:A}\,\bigstar E
$$

$$
\dfrac{}{x\mathsf{U}x}\,\mathsf{U}\mathit{refl}
\qquad
\dfrac{x\mathsf{U}y}{y\mathsf{U}x}\,\mathsf{U}\mathit{symm}
\qquad
\dfrac{x\mathsf{U}y \quad y\mathsf{U}z}{x\mathsf{U}z}\,\mathsf{U}\mathit{trans}
\qquad
\dfrac{x\mathsf{M}y}{x\mathsf{U}y}\,\mathsf{U}I
\qquad
\dfrac{\begin{matrix}[x\mathsf{M}y]\\ \vdots \\ \alpha\end{matrix}}{\alpha}\,\mathsf{M}\mathit{ser}
$$

$$
\dfrac{x\mathsf{M}y}{y\mathsf{M}y}\,\mathsf{M}\mathit{srefl}
\qquad
\dfrac{\alpha(x) \quad x\mathsf{M}x \quad x\mathsf{M}y}{\alpha(y/x)}\,\mathsf{M}\mathit{sub1}
\qquad
\dfrac{\alpha(y) \quad x\mathsf{M}x \quad x\mathsf{M}y}{\alpha(x/y)}\,\mathsf{M}\mathit{sub2}
$$

In RAA, $A \neq \bot$.
In $\bigstar I$, y is fresh: it is different from x and does not occur in any assumption on which $y : A$ depends other than $x\mathsf{R}y$.
In $\mathsf{M}\mathit{ser}$, y is fresh: it is different from x and does not occur in α nor in any assumption on which α depends other than $x\mathsf{M}y$.
We refer to the fresh y in $\bigstar I$ and $\mathsf{M}\mathit{ser}$ as the *parameter* of the rule.

Fig. 8.1. The rules of MSQS

- $\Box A$ means: A is true after the application of any unitary transformation.
- $\blacksquare A$ means: A is true in each quantum state obtained by a total measurement.

A *labeled formula* (*l-formula*) is an expression $x : A$, where x is a label and A is an m-formula. A *formula* is either an r-formula or an l-formula. The metavariable α, possibly indexed, ranges over formulas. We write $\alpha(x)$ to denote that the label x occurs in the formula α, so that $\alpha(y/x)$ denotes the substitution of the label y for all occurrences of x in α. Furthermore, we say that an l-formula $x : A$ is *atomic* when A is atomic, which is the case when A is a propositional symbol or $\bot$. Finally, we define the *grade* of an l-formula $x : A$, in symbols $grade(x : A)$, to be the number of times $\supset$ and $\bigstar$ occur in A, so that $grade(x : A) = 0$ for an atomic A.

8.2.2 The rules of MSQS

Figure 8.1 shows the rules of MSQS, where the notion of *discharged/open assumption* is standard [79, 95], e.g. the formula $[x : A]$ is discharged in the rule $\supset I$:

Propositional rules: The rules $\supset I$, $\supset E$ and RAA are just the labeled version of the standard ([79, 95]) natural deduction rules for implication introduction and elimination and for *reductio ad absurdum*, where we enforce Prawitz's side condition that $A \neq \bot$. The "mixed" rule $\bot E$ allows us to derive a generic formula α whenever we have obtained a contradiction $\bot$ at a world x; in this case, if α is $z : A$ (with $z \neq x$), we do not enforce the side condition that $A \neq \bot$ but allow the rule to derive $y : \bot$ for some y from $x : \bot$.[1]

Modal rules: We give the rules for a generic modal operator $\bigstar$, with a corresponding generic relation R, since all the modal operators share the structure of these basic introduction/elimination rules; this holds because, for instance, we express $x : \Box A$ as the metalevel implication $x\mathsf{U}y \Longrightarrow y : A$ for an arbitrary y accessible from x. In particular:

- if $\bigstar$ is $\Box$ then R is U,
- if $\bigstar$ is $\blacksquare$ then R is M.

Other rules:

- In order to axiomatize $\Box$, we add rules $\mathsf{U}\mathit{refl}$, $\mathsf{U}\mathit{symm}$, and $\mathsf{U}\mathit{trans}$, formalizing that U is an equivalence relation.
- In order to axiomatize $\blacksquare$, we add rules formalizing the following properties:
  - If $x\mathsf{M}y$ then there is a specific unitary transformation (depending on x and y) that generates y from x: rule $\mathsf{U}I$.
  - The total measurement process is serial: rule $\mathsf{M}\mathit{ser}$ says that if from the assumption $x\mathsf{M}y$ we can derive α for a *fresh* y (i.e. y is different from x and does not occur in α nor in any assumption on which α depends other than $x\mathsf{M}y$), then we can discharge the assumption (since there always is some y such that $x\mathsf{M}y$) and conclude α.
  - The total measurement process is shift-reflexive: rule $\mathsf{M}\mathit{srefl}$.
  - Invariance with respect to classical worlds: rules $\mathsf{M}\mathit{sub1}$ and $\mathsf{M}\mathit{sub2}$ say that if $x\mathsf{M}x$ and $x\mathsf{M}y$, then y must be equal to x and so we can substitute the one for the other in any formula α.

Definition 8.1 (Derivations and proofs). *A* derivation *of a formula α from a set of formulas Γ in* MSQS *(an* MSQS*-derivation, for short, or just "derivation" when* MSQS *is clear from context or is not needed) is a tree formed using the rules in* MSQS*, ending with α and depending only on a finite subset of Γ. We write $\Gamma \vdash \alpha$ to denote that there exists an* MSQS*-derivation of α from Γ, and denote such a derivation Π graphically as*

$$\begin{array}{c}\Gamma\\ \Pi\\ \alpha\end{array}$$

A derivation in MSQS *of α depending on the empty set is called a* proof *of α and we then write $\vdash \alpha$ as an abbreviation of $\emptyset \vdash \alpha$ and say that α is a theorem of* MSQS.

For instance, the following labeled formula schemata are all provable in MSQS (where, in parentheses, we give the intuitive meaning of each formula in terms of quantum state transformations):

1. $x : \Box A \supset A$
   (the identity transformation is unitary).
2. $x : A \supset \Box\Diamond A$
   (each unitary transformation is invertible).

[1] See [98] for a detailed discussion of the rules for $\bot$, which in particular explains how, in order to maintain the duality of modal operators like $\Box$ and $\Diamond$, it must be possible to propagate a $\bot$ at a world x to any other different world y.

3. $x : \Box A \supset \Box\Box A$
   (unitary transformations are composable).
4. $x : \blacksquare A \supset \blacklozenge A$
   (it is always possible to perform a total measurement of a quantum state).
5. $x : \blacksquare(A \leftrightarrow \blacksquare A)$
   (it is always possible to perform a total measurement with a complete reduction of a quantum state to a classical one).
6. $x : \blacksquare A \supset \blacksquare\blacksquare A$
   (total measurements are composable).

As concrete examples, Figure 8.2 contains the proofs of the formulas 5 and 6, where, for simplicity, here and in the following (cf. Figure 8.5), we employ the rules for equivalence ($\leftrightarrow I$) and for negation ($\neg I$ and $\neg E$), which are derived from the propositional rules as is standard. For instance,

$$\dfrac{\begin{array}{c}[x : A]^1 \\ \mathit{\Pi} \\ y : \bot\end{array}}{x : \neg A}\ \neg I^1 \quad \text{abbreviates} \quad \dfrac{\dfrac{\begin{array}{c}[x : A]^1 \\ \mathit{\Pi} \\ y : \bot\end{array}}{x : \bot}\ \bot E}{x : A \supset \bot}\ \supset I^1$$

We can similarly derive rules about r-formulas. For instance, we can derive a rule for the transitivity of M as shown at the top of the proof of the formula 6 in Figure 8.2:

$$\dfrac{x\mathsf{M}y \quad y\mathsf{M}z}{x\mathsf{M}z}\ \mathsf{M}trans$$

abbreviates

$$\dfrac{x\mathsf{M}y \quad \dfrac{x\mathsf{M}y}{y\mathsf{M}y}\ \mathsf{M}srefl \quad y\mathsf{M}z}{x\mathsf{M}z}\ \mathsf{M}sub1$$

8.3 A semantics for unitary transformations and total measurements

We give a semantics that formally describes unitary transformations and total measurements of quantum states in terms of accessibility relations between worlds, and then prove that MSQS is sound and complete with respect to this semantics. Together with the corresponding result for generic measurements in MSpQS described in Section 8.4, this means that our modal systems indeed provide a representation of quantum states and operations on them, which was one of the main goals of the thesis.

Definition 8.2 (Frames, models, structures). *A* frame *is a tuple $\mathscr{F} = \langle W, U, M\rangle$, where:*

- W *is a non-empty set of* worlds
  (representing abstractly the quantum states);
- $U \subseteq W \times W$ *is an equivalence relation*
  (vUw means that w is obtained by applying a unitary transformation to v; U is an equivalence relation since identity is a unitary transformation, each unitary transformation must be invertible, and unitary transformations are composable);

$$
\dfrac{\dfrac{\dfrac{\dfrac{[y:A]^2 \quad \dfrac{[x\mathsf{M}y]^1}{y\mathsf{M}y}\,\mathsf{M}\mathit{srefl} \quad [y\mathsf{M}z]^3}{z:A}\,\mathsf{M}\mathit{sub1}}{y:\blacksquare A}\,\blacksquare I^3}{y:A\supset\blacksquare A}\,\supset I^2 \qquad \dfrac{\dfrac{[y:\blacksquare A]^4 \quad \dfrac{[x\mathsf{M}y]^1}{y\mathsf{M}y}\,\mathsf{M}\mathit{srefl}}{y:A}\,\blacksquare E}{y:\blacksquare A\supset A}\,\supset I^4}{\dfrac{y:A\leftrightarrow\blacksquare A}{x:\blacksquare(A\leftrightarrow\blacksquare A)}\,\blacksquare I^1}\,\leftrightarrow I
$$

$$
\dfrac{\dfrac{\dfrac{\dfrac{[x:\blacksquare A]^1 \quad \dfrac{[x\mathsf{M}y]^2 \quad \dfrac{[x\mathsf{M}y]^2}{y\mathsf{M}y}\,\mathsf{M}\mathit{srefl} \quad [y\mathsf{M}z]^3}{x\mathsf{M}z}\,\mathsf{M}\mathit{sub1}}{z:A}\,\blacksquare E}{y:\blacksquare A}\,\blacksquare I^3}{x:\blacksquare\blacksquare A}\,\blacksquare I^2}{x:\blacksquare A\supset\blacksquare\blacksquare A}\,\supset I^1
$$

Fig. 8.2. Examples of proofs in MSQS

- $M \subseteq W \times W$
  (vMw means that w is obtained by means of a total measurement of v);

with the following properties:

(i) $\forall v, w.\ vMw \Longrightarrow vUw$
(ii) $\forall v.\ \exists w.\ vMw$
(iii) $\forall v, w.\ vMw \Longrightarrow wMw$
(iv) $\forall v, w.\ vMv\ \&\ vMw \Longrightarrow v = w$

(i) *means that although it is not true that measurement is a unitary transformation, locally for each v, if vMw then there is a particular unitary transformation, depending on v and w, that generates w from v; the vice versa cannot hold, since in quantum theory measurements cannot be used to obtain the unitary evolution of a quantum system.* (ii) *means that each quantum state is totally measurable.* (iii) *and* (iv) *together mean that after a total measurement we obtain a classical world. Figure 8.3 shows properties* (ii)*,* (iii) *and* (iv)*, respectively, as well as the combination of* (iii) *and* (iv).[2]

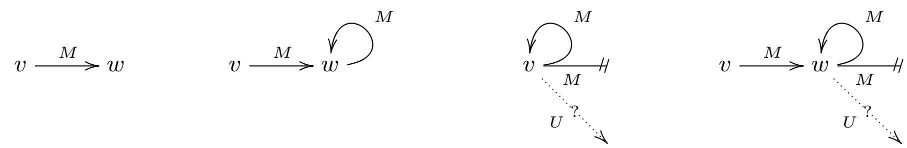
A model *is a pair $\mathscr{M} = \langle \mathscr{F}, V \rangle$, where $\mathscr{F}$ is a frame and $V : W \to 2^{Prop}$ is an interpretation function mapping worlds into sets of formulas.*

A structure *is a pair $\mathscr{S} = \langle \mathscr{M}, \mathscr{I} \rangle$, where $\mathscr{M}$ is a model and $\mathscr{I} : Var \to W$ is an interpretation function mapping variables (labels) into worlds in W.*

We write R to denote a generic frame relation, i.e. $R \in \{U, M\}$, and, slightly abusing notation, we write $\mathscr{I}(\mathsf{R})$ to denote the corresponding R.

Given this semantics, we can define what it means for formulas to be true, and then prove the soundness and completeness of MSQS.

[2] Note that while (iv) says that v is invariant with respect to M, a unitary transformation U could still be applied to v (and hence the dotted arrow decorated with a "?" for U).



(ii) (iii) (iv) (iii) and (iv)

Fig. 8.3. Some properties of the relation M

Definition 8.3 (Truth). Truth *for an m-formula at a world w in a model $\mathscr{M} = \langle W, U, M, V\rangle$ is the smallest relation $\vDash$ satisfying:*

$$
\begin{array}{ll}
\vDash^{\mathscr{M},w} r & \textit{iff } r \in V(w) \\
\vDash^{\mathscr{M},w} A \supset B & \textit{iff } \vDash^{\mathscr{M},w} A \Longrightarrow \vDash^{\mathscr{M},w} B \\
\vDash^{\mathscr{M},w} \Box A & \textit{iff } \forall w'.\ wUw' \Longrightarrow \vDash^{\mathscr{M},w'} A \\
\vDash^{\mathscr{M},w} \blacksquare A & \textit{iff } \forall w'.\ wMw' \Longrightarrow \vDash^{\mathscr{M},w'} A
\end{array}
$$

Hence, $\nvDash^{\mathscr{M},w} \bot$ for any $\mathscr{M}$ and w. For an m-formula A, we write $\vDash^{\mathscr{M}} A$ iff $\vDash^{\mathscr{M},w} A$ for all w.

Truth *for a formula α in a structure $\mathscr{S} = \langle \mathscr{M}, \mathscr{I}\rangle$ is then the smallest relation $\vDash$ satisfying:*

$$
\begin{array}{ll}
\vDash^{\mathscr{M},\mathscr{I}} x\mathsf{M}y & \textit{iff } \mathscr{I}(x)M\mathscr{I}(y) \\
\vDash^{\mathscr{M},\mathscr{I}} x\mathsf{U}y & \textit{iff } \mathscr{I}(x)U\mathscr{I}(y) \\
\vDash^{\mathscr{M},\mathscr{I}} x : A & \textit{iff } \vDash^{\mathscr{M},\mathscr{I}(x)} A
\end{array}
$$

Hence, $\vDash^{\mathscr{M},\mathscr{I}} x\mathsf{R}y$ iff $\mathscr{I}(x)\mathscr{I}(\mathsf{R})\mathscr{I}(y)$ iff $\mathscr{I}(x)R\mathscr{I}(y)$. Moreover, $\nvDash^{\mathscr{M},\mathscr{I}} x : \bot$ for any x, $\mathscr{M}$ and $\mathscr{I}$.

By extension, $\vDash^{\mathscr{M},\mathscr{I}} \Gamma$ iff $\vDash^{\mathscr{M},\mathscr{I}} \alpha$ for all α in the set of formulas Γ. Thus, for a set of formulas Γ and a formula α,

$$
\begin{array}{rl}
\Gamma \vDash \alpha & \text{iff } \forall \mathscr{S}.\ \Gamma \vDash^{\mathscr{S}} \alpha \\
& \text{iff } \forall \mathscr{M}, \mathscr{I}.\ \Gamma \vDash^{\mathscr{M},\mathscr{I}} \alpha \\
& \text{iff } \forall \mathscr{M}, \mathscr{I}.\ \vDash^{\mathscr{M},\mathscr{I}} \Gamma \Longrightarrow \vDash^{\mathscr{M},\mathscr{I}} \alpha
\end{array}
$$

We omit $\mathscr{M}$ when it is not relevant and, for example, write $\Gamma \vDash^{\mathscr{I}} \alpha$ when $\vDash^{\mathscr{I}} \Gamma$ implies $\vDash^{\mathscr{I}} \alpha$.

By adapting standard proofs to the case of labeled deduction (see, e.g., [42, 79, 90, 95, 98]), we can show:

Theorem 8.4 (Soundness and completeness of MSQS). $\Gamma \vdash \alpha$ *iff* $\Gamma \vDash \alpha$. $\Box$

Theorem 8.4 follows from Theorems 8.5 and 8.10 below.

Theorem 8.5 (Soundness of MSQS). $\Gamma \vdash \alpha$ *implies* $\Gamma \vDash \alpha$.

Proof. We let $\mathscr{M}$ be an arbitrary model and prove that if $\Gamma \vdash \alpha$ then $\Gamma \models^{\mathscr{I}} \alpha$ for any $\mathscr{I}$, i.e. $\models^{\mathscr{I}} \Gamma$ implies $\models^{\mathscr{I}} \alpha$ for any $\mathscr{I}$. The proof proceeds by induction on the structure of the derivation of α from Γ. The base case, where $\alpha \in \Gamma$, is trivial. There is one step case for each rule of MSQS.

Consider an application of the rule RAA,

$$\dfrac{\begin{array}{c}[x : \neg A]\\ \vdots \\ y : \bot\end{array}}{x : A}\ RAA$$

where $\Gamma' \models y : \bot$ with $\Gamma' = \Gamma \cup \{x : \neg A\}$. By the induction hypothesis, $\Gamma' \vdash y : \bot$ implies $\Gamma' \models^{\mathrm{i}} y : \bot$ for any $\mathscr{I}$. We assume $\models^{\mathscr{I}} \Gamma$ and prove $\models^{\mathscr{I}} x : A$. Since $\not\models^{w} \bot$ for any world w, from the induction hypothesis we obtain $\not\models^{\mathscr{I}} \Gamma'$, and thus $\not\models^{\mathscr{I}} x : \neg A$, i.e. $\models^{\mathscr{I}} x : A$ and $\not\models^{\mathscr{I}} x : \bot$.

Consider an application of the rule $\bot E$,

$$\dfrac{x : \bot}{\alpha}\ \bot E$$

with $\Gamma \vdash x : \bot$. By the induction hypothesis, $\Gamma \vdash x : \bot$ implies $\Gamma \models^{\mathscr{I}} x : \bot$ for any $\mathscr{I}$. We assume $\models^{\mathscr{I}} \Gamma$ and prove $\models^{\mathscr{I}} \alpha$ for an arbitrary formula α. If $\models^{\mathscr{I}} \Gamma$ then $\models^{\mathscr{I}} x : \bot$ by the induction hypothesis, i.e. $\models^{\mathscr{I}(x)} \bot$. But since $\not\models^{w} \bot$ for any world w, then $\not\models^{\mathscr{I}} \Gamma$ and thus $\models^{\mathscr{I}} \alpha$ for any α.

Consider an application of the rule $\bigstar I$

$$\dfrac{\begin{array}{c}[x\mathsf{R}y]\\ \vdots \\ y : A\end{array}}{x : \bigstar A}\ \bigstar I$$

where $\Gamma' \vdash y : A$ with y fresh and with $\Gamma' = \Gamma \cup \{x\mathsf{R}y\}$. By the induction hypothesis, for all interpretations $\mathscr{I}$, if $\models^{\mathscr{I}} \Gamma$ then $\models^{\mathscr{I}} y : A$. We let $\mathscr{I}$ be any interpretation such that $\models^{\mathscr{I}} \Gamma$, and show that $\models^{\mathscr{I}} x : \bigstar A$. Let w be any world such that $\mathscr{I}(x)\mathscr{I}(\mathsf{R})w$. Since $\mathscr{I}$ can be trivially extended to another interpretation (still called $\mathscr{I}$ for simplicity) by setting $\mathscr{I}(y) = w$, the induction hypothesis yields $\models^{\mathscr{I}} y : A$, i.e. $\models^{w} A$, and thus $\models^{\mathscr{I}(x)} \bigstar A$, i.e. $\models^{\mathscr{I}} x : \bigstar A$.

Consider an application of the rule $\bigstar E$

$$\dfrac{x : \bigstar A \quad x\mathsf{R}y}{y : A}\ \bigstar E$$

with $\Gamma_1 \vdash x : \bigstar A$ and $\Gamma_2 \vdash x\mathsf{R}y$, and $\Gamma \supseteq \Gamma_1 \cup \Gamma_2$. We assume $\models^{\mathscr{I}} \Gamma$ and prove $\models^{\mathscr{I}} y : A$. By the induction hypothesis, for all interpretations $\mathscr{I}$, if $\models^{\mathscr{I}} \Gamma_1$ then $\models^{\mathscr{I}} x : \bigstar A$ and if $\models^{\mathscr{I}} \Gamma_2$ then $\mathscr{I}(x)\mathscr{I}(\mathsf{R})\mathscr{I}(y)$. If $\models^{\mathscr{I}} \Gamma$, then $\models^{\mathscr{I}} x : \bigstar A$ and $\mathscr{I}(x)\mathscr{I}(\mathsf{R})\mathscr{I}(y)$, and thus $\models^{\mathscr{I}} y : A$.

The rules $\mathsf{U}\mathit{refl}$, $\mathsf{U}\mathit{symm}$, and $\mathsf{U}\mathit{trans}$ are sound by the properties of U.

The rule $\mathsf{U}I$ is sound by property (i) in Definition 8.2.

Consider an application of the rule $\mathsf{M}\mathit{ser}$

$$\dfrac{\begin{array}{c}[x\mathsf{M}y]\\ \vdots\\ \alpha\end{array}}{\alpha}\ \mathsf{M}ser$$

with $\Gamma' = \Gamma \cup \{x\mathsf{M}y\}$, for y fresh. By the induction hypothesis, $\Gamma' \vdash \alpha$ implies $\Gamma' \models^{\mathscr{I}} \alpha$ for any $\mathscr{I}$. Let us suppose that there is an $\mathscr{I}'$ such that $\models^{\mathscr{I}'} \Gamma'$ and $\not\models^{\mathscr{I}'} \alpha$. Let us consider an $\mathscr{I}''$ such that $\mathscr{I}''(z) = \mathscr{I}'(z)$ for all z such that $z \neq y$ and $\mathscr{I}''(y)$ is the world w such that $\mathscr{I}''(y)Mw$, which exists by property (ii) in Definition 8.2. Since y does not occur in Γ nor in α, we then have that $\models^{\mathscr{I}''} \Gamma'$ and $\not\models^{\mathscr{I}''} \alpha$, contradicting the universality of the consequence of the induction hypothesis. Hence, $\mathsf{M}ser$ is sound.

The rule $\mathsf{M}srefl$ is sound by property (iii) in Definition 8.2.

Consider an application of the rule $\mathsf{M}sub1$

$$\frac{\alpha(x) \quad x\mathsf{M}x \quad x\mathsf{M}y}{\alpha(y/x)}\ \mathsf{M}sub1$$

with $\Gamma_1 \vdash \alpha(x)$, $\Gamma_2 \vdash x\mathsf{M}x$, $\Gamma_3 \vdash x\mathsf{M}y$, and $\Gamma \supseteq \Gamma_1 \cup \Gamma_2 \cup \Gamma_3$. We assume $\models^{\mathscr{I}} \Gamma$ and prove $\models^{\mathscr{I}} \alpha(y/x)$. By the induction hypothesis, $\Gamma_1 \vdash \alpha(x)$ implies $\Gamma_1 \models^{\mathscr{I}} \alpha(x)$, $\Gamma_2 \vdash x\mathsf{M}x$ implies if $\models^{\mathscr{I}} \Gamma_2$ then $\mathscr{I}(x)M\mathscr{I}(x)$, and $\Gamma_3 \vdash x\mathsf{M}y$ implies if $\models^{\mathscr{I}} \Gamma_3$ then $\mathscr{I}(x)M\mathscr{I}(y)$. By property *(iv)* in Definition 8.2, we then have $\mathscr{I}(x) = \mathscr{I}(y)$ and thus $\models^{\mathscr{I}} \alpha(y/x) : A$. The case for rule $\mathsf{M}sub2$ follows analogously.

To prove completeness (Theorem 8.10), we give some preliminary definitions and results. For simplicity, we split each set of formulas Γ into a pair (LF, RF) of the subsets of l-formulas and r-formulas of Γ, and then prove $(LF, RF) \models \alpha$ implies $(LF, RF) \vdash \alpha$. We call (LF, RF) a *context* and, slightly abusing notation, we write $\alpha \in (LF, RF)$ whenever $\alpha \in LF$ or $\alpha \in RF$, and write $x \in (LF, RF)$ whenever the label x occurs in some $\alpha \in (LF, RF)$. We say that a context (LF, RF) is *consistent* iff $(LF, RF) \nvdash x : \bot$ for every x, so that we have:

Fact 1 *If (LF, RF) is consistent, then for every x and every A, either $(LF \cup \{x : A\}, RF)$ is consistent or $(LF \cup \{x : \neg A\}, RF)$ is consistent.*

Let $\overline{(LF, RF)}$ be the *deductive closure* of (LF, RF) for r-formulas under the rules of MSQS, i.e.

$$\overline{(LF, RF)} \equiv \{x\mathsf{R}y \mid (LF, RF) \vdash x\mathsf{R}y\}$$

for $\mathsf{R} \in \{\mathsf{U}, \mathsf{M}\}$. We say that a context (LF, RF) is *maximally consistent* iff

1. it is consistent,
2. it is deductively closed for r-formulas, i.e. $(LF, RF) = \overline{(LF, RF)}$, and
3. for every x and every A, either $x : A \in (LF, RF)$ or $x : \neg A \in (LF, RF)$.

Completeness follows by a Henkin–style proof, where a canonical structure

$$\mathscr{S}^c = \langle \mathscr{M}^c, \mathscr{I}^c \rangle = \langle W^c, U^c, M^c, V^c, \mathscr{I}^c \rangle$$

is built to show that $(LF, RF) \nvdash \alpha$ implies $(LF, RF) \not\models^{\mathscr{S}^c} \alpha$, i.e. $\models^{\mathscr{S}^c} (LF, RF)$ and $\not\models^{\mathscr{S}^c} \alpha$.

In standard proofs for unlabeled modal logics (e.g. [26]) and for other non-classical logics, the set W^c is obtained by progressively building maximally consistent sets of formulas, where consistency is locally checked within each set. In our case, given the presence of l-formulas and r-formulas, we modify the Lindenbaum lemma to extend (LF, RF) to one single maximally consistent context (LF^*, RF^*), where consistency is "globally" checked also against the additional assumptions in RF.[3] The elements of W^c are then built by partitioning LF^* and RF^* with respect to the labels, and the relations R between the worlds are defined by exploiting the information in RF^*.

In the Lindenbaum lemma for predicate logic, a maximally consistent and ω-complete set of formulas is inductively built by adding for every formula $\neg\forall x.A$ a *witness* to its truth, namely a formula $\neg A[c/x]$ for some new individual constant c. This ensures that the resulting set is ω-complete, i.e. that if, for every closed term t, $A[t/x]$ is contained in the set, then so is $\forall x.A$. A similar procedure applies here in the case of l-formulas of the form $x : \neg\bigstar A$. That is, together with $x : \neg\bigstar A$ we consistently add $y : \neg A$ and $x\mathsf{R}y$ for some new y, which acts as a *witness world* to the truth of $x : \neg\bigstar A$. This ensures that the maximally consistent context (LF^*, RF^*) is such that if $x\mathsf{R}z \in (LF^*, RF^*)$ implies $z : B \in (LF^*, RF^*)$ for every z, then $x : \bigstar B \in (LF^*, RF^*)$, as shown in Lemma 8.7 below. Note that in the standard completeness proof for unlabeled modal logics, one instead considers a canonical model $\mathscr{M}^c$ and shows that if $w \in W^c$ and $\models^{\mathscr{M}^c,w} \neg\bigstar A$, then W^c also contains a world w' accessible from w that serves as a witness world to the truth of $\neg\bigstar A$ at w, i.e. $\models^{\mathscr{M}^c,w'} \neg A$.

Lemma 8.6. *Every consistent context* (LF, RF) *can be extended to a maximally consistent context* (LF^*, RF^*).

Proof. We first extend the language of MSQS with infinitely many new constants for witness worlds. Systematically let b range over labels, c range over the new constants for witness worlds, and a range over both. All these may be subscripted. Let $l_1, l_2, \ldots$ be an enumeration of all l-formulas in the extended language; when l_i is $a : A$, we write $\neg l_i$ for $a : \neg A$. Starting from $(LF_0, RF_0) = (LF, RF)$, we inductively build a sequence of consistent contexts by defining (LF_{i+1}, RF_{i+1}) to be:

- (LF_i, RF_i), if $(LF_i \cup \{l_{i+1}\}, RF_i)$ is inconsistent; else
- $(LF_i \cup \{l_{i+1}\}, RF_i)$, if l_{i+1} is not $a : \neg\bigstar A$; else
- $(LF_i \cup \{a : \neg\bigstar A, c : \neg A\}, RF_i \cup \{a\mathsf{R}c\})$ for a $c \notin (LF_i \cup \{a : \neg\bigstar A\}, RF_i)$, if l_{i+1} is $a : \neg\bigstar A$.

Every (LF_i, RF_i) is consistent. To show this we show that if $(LF_i \cup \{a : \neg\bigstar A\}, RF_i)$ is consistent, then so is $(LF_i \cup \{a : \neg\bigstar A, c : \neg A\}, RF_i \cup \{a\mathsf{R}c\})$ for a $c \notin (LF_i \cup \{a : \neg\bigstar A\}, RF_i)$; the other cases follow by construction. We proceed by contraposition. Suppose that

$$(LF_i \cup \{a : \neg\bigstar A, c : \neg A\}, RF_i \cup \{a\mathsf{R}c\}) \vdash a_j : \bot$$

where $c \notin (LF_i \cup \{a : \neg\bigstar A\}, RF_i)$. Then, by RAA,

$$(LF_i \cup \{a : \neg\bigstar A\}, RF_i \cup \{a\mathsf{R}c\}) \vdash c : A\,,$$

[3] We consider only consistent contexts. If (LF, RF) is inconsistent, then $LF, RF \vdash x : A$ for all $x : A$, and thus completeness immediately holds for l-formulas. Our language does not allow us to define inconsistency for a set of r-formulas, but, whenever (LF, RF) is inconsistent, the canonical model built in the following is nonetheless a counter-model to non-derivable r-formulas.

and $\bigstar I$ yields

$$(LF_i \cup \{a : \neg\bigstar A\}, RF_i) \vdash a : \bigstar A\,.^4$$

Since also

$$(LF_i \cup \{a : \neg\bigstar A\}, RF_i) \vdash a : \neg\bigstar A\,,$$

by $\neg E$ we have

$$(LF_i \cup \{a : \neg\bigstar A\}, RF_i) \vdash a : \bot\,,$$

i.e. $(LF_i \cup \{a : \neg\bigstar A\}, RF_i)$ is inconsistent. Contradiction.

Now define

$$(LF^*, RF^*) = \overline{(\bigcup_{i\geq 0} LF_i, \bigcup_{i\geq 0} RF_i)}$$

We show that (LF^*, RF^*) is maximally consistent, by showing that it satisfies the three conditions in the definition of maximal consistency. For the first condition, note that if

$$(\bigcup_{i\geq 0} LF_i, \bigcup_{i\geq 0} RF_i)$$

is consistent, then so is

$$\overline{(\bigcup_{i\geq 0} LF_i, \bigcup_{i\geq 0} RF_i)}\,.$$

Now suppose that (LF^*, RF^*) is inconsistent. Then for some finite (LF', RF') included in (LF^*, RF^*) there exists an a such that $(LF', RF') \vdash a : \bot$. Every l-formula $l \in (LF', RF')$ is in some (LF_j, RF_j). For each $l \in (LF', RF')$, let i_l be the least j such that $l \in (LF_j, RF_j)$, and let $i = max\{i_l \mid l \in (LF', RF')\}$. Then $(LF', RF') \subseteq (LF_i, RF_i)$, and (LF_i, RF_i) is inconsistent, which is not the case.

The second condition is satisfied by definition of (LF^*, RF^*).

For the third condition, suppose that $l_{i+1} \notin (LF^*, RF^*)$. Then $l_{i+1} \notin (LF_{i+1}, RF_{i+1})$ and $(LF_i \cup \{l_{i+1}\}, RF_i)$ is inconsistent. Thus, by Fact 1, $(LF_i \cup \{\neg l_{i+1}\}, RF_i)$ is consistent, and $\neg l_{i+1}$ is consistently added to some (LF_j, RF_j) during the construction, and therefore $\neg l_{i+1} \in (LF^*, RF^*)$.

The following lemma states some properties of maximally consistent contexts.

Lemma 8.7. *Let (LF^*, RF^*) be a maximally consistent context. Then*

1. $(LF^*, RF^*) \vdash a_i \mathsf{R} a_j$ *iff* $a_i \mathsf{R} a_j \in (LF^*, RF^*)$.
2. $(LF^*, RF^*) \vdash a : A$ *iff* $a : A \in (LF^*, RF^*)$.
3. $a : B \supset C \in (LF^*, RF^*)$ *iff* $a : B \in (LF^*, RF^*)$ *implies* $a : C \in (LF^*, RF^*)$.

[4] Note that if $A = \bot$, then we cannot apply RAA. But in that case, if

$$(LF_i \cup \{a : \neg\bigstar\bot, c : \neg\bot\}, RF_i \cup \{a\mathsf{R}c\}) \vdash a_j : \bot$$

then also

$$(LF_i \cup \{a : \neg\bigstar\bot\}, RF_i \cup \{a\mathsf{R}c\}) \vdash a_j : \bot\,,$$

which can only be the case if either LF_i contains for some B both $a : \bigstar\neg B$ and $a : \bigstar B$, which give rise to a $\bot$ at c via $a\mathsf{R}c$, or LF_i contains $a : \bigstar A$, i.e. $a : \bigstar\bot$. In both such cases, it must be that $(LF_i \cup \{a : \neg\bigstar A\}, RF_i)$ is inconsistent, which contradicts the assumption.

4. $a_i : \bigstar B \in (LF^*, RF^*)$ *iff* $a_i \mathsf{R} a_j \in (LF^*, RF^*)$ *implies* $a_j : B \in (LF^*, RF^*)$ *for all* a_j.

Proof. 1 and *2* follow immediately by definition. We only treat *4* as *3* follows analogously. For the left-to-right direction, suppose that $a_i : \bigstar B \in (LF^*, RF^*)$. Then, by (ii), $(LF^*, RF^*) \vdash a_i : \bigstar B$, and, by $\bigstar E$, we have $(LF^*, RF^*) \vdash a_i \mathsf{R} a_j$ implies $(LF^*, RF^*) \vdash a_j : B$ for all a_j. By *1* and *2*, conclude $a_i \mathsf{R} a_j \in (LF^*, RF^*)$ implies $a_j : B \in (LF^*, RF^*)$ for all a_j. For the converse, suppose that $a_i : \bigstar B \notin (LF^*, RF^*)$. Then $a_i : \neg \bigstar B \in (LF^*, RF^*)$, and, by the construction of (LF^*, RF^*), there exists an a_j such that $a_i \mathsf{R} a_j \in (LF^*, RF^*)$ and $a_j : B \notin (LF^*, RF^*)$.

We can now define the canonical structure

$$\mathscr{S}^c = \langle \mathscr{M}^c, \mathscr{I}^c \rangle = \langle W^c, U^c, M^c, V^c, \mathscr{I}^c \rangle .$$

Definition 8.8. *Given a maximal consistent context* (LF^*, RF^*)*, we define the* canonical structure $\mathscr{S}^c$ *as follows:*

- $W^c = \{a \mid a \in (LF^*, RF^*)\}$,
- $(a_i, a_j) \in U^c$ *iff* $a_i \mathsf{U} a_j \in (LF^*, RF^*)$,
- $(a_i, a_j) \in M^c$ *iff* $a_i \mathsf{M} a_j \in (LF^*, RF^*)$,
- $V^c(r) = a$ *iff* $a : r \in (LF^*, RF^*)$,
- $\mathscr{I}^c(a) = a$.

Note that the standard definition of R^c adopted for unlabeled modal logics, i.e.

$$(a_i, a_j) \in R^c \text{ iff } \{A \mid \Box A \in a_i\} \subseteq a_j ,$$

is not applicable in our setting, since $\{A \mid \Box A \in a_i\} \subseteq a_j$ does *not* imply $\vdash a_i \mathsf{R} a_j$. We would therefore be unable to prove completeness for r-formulas, since there would be cases, e.g. when $RF = \{\}$, where $\nvdash a_i \mathsf{R} a_j$ but $(a_i, a_j) \in R^c$ and thus $\vDash^{\mathscr{S}^c} a_i \mathsf{R} a_j$. Hence, we instead define $(a_i, a_j) \in R^c$ iff $a_i \mathsf{R} a_j \in (LF^*, RF^*)$; note that therefore $a_i \mathsf{R} a_j \in (LF^*, RF^*)$ implies $\{A \mid \Box A \in a_i\} \subseteq a_j$. As a further comparison with the standard definition, note that in the canonical model the label a can be identified with the set of formulas $\{A \mid a : A \in (LF^*, RF^*)\}$. Moreover, we immediately have:

Fact 2 $a_i \mathsf{R} a_j \in (LF^*, RF^*)$ *iff* $(LF^*, RF^*) \vDash^{\mathscr{S}^c} a_i \mathsf{R} a_j$.

The deductive closure of (LF^*, RF^*) for r-formulas ensures not only completeness for r-formulas, as shown in Theorem 8.10 below, but also that the conditions on R^c are satisfied, so that $\mathscr{S}^c$ is really a structure for MSQS. More concretely:

- U^c is an equivalence relation by construction and rules $\mathsf{U}\mathit{refl}$, $\mathsf{U}\mathit{symm}$, and $\mathsf{U}\mathit{trans}$. For instance, for transitivity, consider an arbitrary context (LF, RF) from which we build $\mathscr{S}^c$. Assume $(a_i, a_j) \in U^c$ and $(a_j, a_k) \in U^c$. Then $a_i \mathsf{U} a_j \in (LF^*, RF^*)$ and $a_j \mathsf{U} a_k \in (LF^*, RF^*)$. Since (LF^*, RF^*) is deductively closed, by *1* in Lemma 8.7 and rule $\mathsf{U}\mathit{trans}$, we have $a_i \mathsf{U} a_k \in (LF^*, RF^*)$. Thus, $(a_i, u_k) \in U^c$ and U^c is indeed transitive.
- $\forall v, w \in W^c.\ vMw \Longrightarrow vUw$ holds by construction and rule $\mathsf{U}I$.

- $\forall v \in W^c.\ \exists w \in W^c.\ vMw$ holds by construction and rule $\mathsf{M}ser$. For the sake of contradiction, consider an arbitrary a_i and a variable a'_j that do not satisfy the property. Define $(LF', RF') = (LF^*, RF^*) \cup \{a_i \mathsf{M} a'_j\}$. Then it cannot be the case that $(LF', RF') \vdash \alpha$, for otherwise $(LF^*, RF^*) \vdash \alpha$ would be derivable by an application of the rule $\mathsf{M}ser$. Thus, $(LF', RF') \nvdash \alpha$. But then (LF', RF') must be in the chain of contexts built in Lemma 8.7. So, by the maximality of (LF^*, RF^*), we have that $(LF', RF') = (LF^*, RF^*)$, contradicting our assumption. Hence, for some a_j, the r-formula $a_i \mathsf{M} a_j$ is in (LF^*, RF^*), which is what we had to show.
- $\forall v, w \in W^c.\ vMw \Longrightarrow wMw$ holds by construction and rule $\mathsf{M}srefl$.
- $\forall v, w \in W^c.\ vMv\ \&\ vMw \Longrightarrow v = w$ holds by construction and rules $\mathsf{M}sub1$ and $\mathsf{M}sub2$ since v is a classical world. Consider an arbitrary context (LF, RF) from which we build $\mathscr{S}^c$ and assume $(a_i, a_i) \in M^c$ and $(a_i, a_j) \in M^c$. Then $a_i \mathsf{M} a_i \in (LF^*, RF^*)$ and $a_i \mathsf{M} a_j \in (LF^*, RF^*)$. Thus, for each $a_i : A \in (LF^*, RF^*)$, we also have $a_j : A \in (LF^*, RF^*)$; otherwise, since (LF^*, RF^*) is deductively closed, we would have $a_j : \neg A \in (LF^*, RF^*)$ and also $a_j : A \in (LF^*, RF^*)$ by *1* in Lemma 8.7 and rule $\mathsf{M}sub1$, and thus a contradiction. Similarly, if $a_j : A \in (LF^*, RF^*)$ then $a_i : A \in (LF^*, RF^*)$ by rule $\mathsf{M}sub2$. Hence, for each m-formula A, we have that $a_i : A \in (LF^*, RF^*)$ iff $a_j : A \in (LF^*, RF^*)$, which means that a_i and a_j are equal with respect to m-formulas.

  Under the same assumptions, we can similarly show that a_i and a_j are equal with respect to r-formulas, i.e. that whenever (LF^*, RF^*) contains an r-formula that includes a_i then it also contains the same r-formula with a_j substituted for a_i, and vice versa. To this end, we must consider eight different cases corresponding to eight different r-formulas.

  1. If $a_k \mathsf{U} a_i \in (LF^*, RF^*)$ for some a_k, then from the assumption that $a_i \mathsf{M} a_j \in (LF^*, RF^*)$ we have $a_i \mathsf{U} a_j \in (LF^*, RF^*)$, by *1* in Lemma 8.7 and rule $\mathsf{U}I$. Therefore, $a_k \mathsf{U} a_j \in (LF^*, RF^*)$ by rule $\mathsf{U}trans$.
  2. We can reason similarly for $a_j \mathsf{U} a_k \in (LF^*, RF^*)$ and also apply rules $\mathsf{U}I$ and $\mathsf{U}trans$ to conclude that then also $a_i \mathsf{U} a_k \in (LF^*, RF^*)$.
  3. If $a_i \mathsf{U} a_k \in (LF^*, RF^*)$ for some a_k, then from the assumption that $a_i \mathsf{M} a_j \in (LF^*, RF^*)$ we have $a_i \mathsf{U} a_j \in (LF^*, RF^*)$, by *1* in Lemma 8.7 and rule $\mathsf{U}I$, and thus $a_j \mathsf{U} a_i \in (LF^*, RF^*)$, by rule $\mathsf{U}symm$. Therefore, $a_j \mathsf{U} a_k \in (LF^*, RF^*)$ by rule $\mathsf{U}trans$.
  4. We can reason similarly for $a_k \mathsf{U} a_j \in (LF^*, RF^*)$ and also apply rules $\mathsf{U}I$, $\mathsf{U}symm$, and $\mathsf{U}trans$ to conclude that then also $a_k \mathsf{U} a_i \in (LF^*, RF^*)$.
  5. If $a_k \mathsf{M} a_i \in (LF^*, RF^*)$ for some a_k, then from the assumption that $a_i \mathsf{M} a_j \in (LF^*, RF^*)$ we have $a_k \mathsf{M} a_j \in (LF^*, RF^*)$, by *1* in Lemma 8.7 and the derived rule $\mathsf{M}trans$.
  6. We can reason similarly for $a_j \mathsf{M} a_k \in (LF^*, RF^*)$ and also apply rule $\mathsf{M}trans$ to conclude that then also $a_i \mathsf{U} a_k \in (LF^*, RF^*)$.
  7. If $a_i \mathsf{M} a_k \in (LF^*, RF^*)$ for some a_k, then from the assumptions that $a_i \mathsf{M} a_i \in (LF^*, RF^*)$ and $a_i \mathsf{M} a_j \in (LF^*, RF^*)$ we have $a_j \mathsf{M} a_k \in (LF^*, RF^*)$, by *1* in Lemma 8.7 and rule $\mathsf{M}sub1$.
  8. We can reason similarly for $a_k \mathsf{M} a_j \in (LF^*, RF^*)$ and apply rule $\mathsf{M}sub2$ to conclude that then also $a_k \mathsf{M} a_i \in (LF^*, RF^*)$.

  Hence, a_i and a_j are equal also with respect to r-formulas, and thus $a_i = a_j$ whenever $(a_i, a_i) \in M^c$ and $(a_i, a_j) \in M^c$, which is what we had to show.

By Lemma 8.7 and Fact 2, it follows that:

Lemma 8.9. $a : A \in (LF^*, RF^*)$ *iff* $(LF^*, RF^*) \vDash^{\mathscr{S}^c} a : A$.

Proof. We proceed by induction on the grade of $a : A$, and we treat only the step case where $a : A$ is $a_i : \bigstar B$; the other cases follow analogously. For the left-to-right direction, assume $a_i : \bigstar B \in (LF^*, RF^*)$. Then, by Lemma 8.7, $a_i \mathsf{R} a_j \in (LF^*, RF^*)$ implies $a_j : B \in (LF^*, RF^*)$, for all a_j. Fact 2 and the induction hypothesis yield that $(LF^*, RF^*) \vDash^{\mathscr{S}^c} a_j : B$ for all a_j such that $(LF^*, RF^*) \vDash^{\mathscr{S}^c} a_i \mathsf{R} a_j$, i.e. $(LF^*, RF^*) \vDash^{\mathscr{S}^c} a_i : \bigstar B$ by Definition 8.3. For the converse, assume $a_i : \neg\bigstar B \in (LF^*, RF^*)$. Then, by Lemma 8.7, $a_i \mathsf{R} a_j \in (LF^*, RF^*)$ and $a_j : \neg B \in (LF^*, RF^*)$, for some a_j. Fact 2 and the induction hypothesis yield $(LF^*, RF^*) \vDash^{\mathscr{S}^c} a_i \mathsf{R} a_j$ and $(LF^*, RF^*) \vDash^{\mathscr{S}^c} a_j : \neg B$, i.e. $(LF^*, RF^*) \vDash^{\mathscr{S}^c} a_i : \neg\bigstar B$ by Definition 8.3.

We can now finally show:

Theorem 8.10 (Completeness of MSQS). *$\Gamma \vDash \alpha$ implies $\Gamma \vdash \alpha$.*

Proof. If $(LF, RF) \nvdash b_i \mathsf{R} b_j$, then $b_i \mathsf{R} b_j \notin (LF^*, RF^*)$, and thus $(LF^*, RF^*) \nvDash^{\mathscr{S}^c} b_i \mathsf{R} b_j$ by Fact 2.

If $(LF, RF) \nvdash b : A$, then $(LF \cup \{b : \neg A\}, RF)$ is consistent; otherwise there exists a b_i such that $(LF \cup \{b : \neg A\}, RF) \vdash b_i : \bot$, and then $(LF, RF) \vdash b : A$. Therefore, by Lemma 8.6, $(LF \cup \{b : \neg A\}, RF)$ is included in a maximally consistent context $((LF \cup \{b : \neg A\})^*, RF^*)$. Then, by Lemma 8.9, $((LF \cup \{b : \neg A\})^*, RF^*) \vDash^{\mathscr{S}^C} b : \neg A$, i.e. $((LF \cup \{b : \neg A\})^*, RF^*) \nvDash^{\mathscr{S}^c} b : A$, and thus $(LF, RF) \nvDash^{\mathscr{S}^c} b : A$.

8.4 Generic measurements

In quantum computing, not all measurements are required to be total: think, for example, of the case of observing only the first qubit of a quantum state. To this end, in this section, we propose MSpQS, a variant of MSQS that provides a modal system representing all the possible (thus not necessarily total) measurements. We obtain MSpQS from MSQS by means of the following changes:

- The alphabet of the modal language contains the unary modal operator $\boxdot$ instead of $\blacksquare$, with corresponding $\diamonddot$, where $\boxdot A$ intuitively means that A is true in each quantum state obtained by a measurement.
- The set of relational formulas contains expressions of the form $x\mathsf{P}y$ instead of $x\mathsf{M}y$, and we now write $x\mathsf{R}y$ to denote a generic r-formula, with $\mathsf{R} \in \{\mathsf{U}, \mathsf{P}\}$.
- The rules of MSpQS are given in Figure 8.4. In particular, $\bigstar$ is either $\Box$ (as before) or $\boxdot$, for which then R is P, and whose properties are formalized by the following additional rules:
  - If $x\mathsf{P}y$ then there is a specific unitary transformation (depending on x and y) that generates y from x: rule $\mathsf{PU}I$.
  - The measurement process is transitive: rule $\mathsf{P}trans$.
  - There are (always reachable) classical worlds: $class$ says that y is a classical world reachable from world x by a measurement.
  - Invariance with respect to classical worlds for measurement: rules $\mathsf{P}sub1$ and $\mathsf{P}sub2$.

$\supset I,\ \supset E,\ RAA,\ \bot E,\ \bigstar I,\ \bigstar E,\ \mathsf{U}\mathit{refl},\ \mathsf{U}\mathit{symm},\ \mathsf{U}\mathit{trans},$

$$\frac{x\mathsf{P}y}{x\mathsf{U}y}\ \mathsf{PU}I \qquad \frac{x\mathsf{P}y \quad y\mathsf{P}z}{x\mathsf{P}z}\ \mathsf{P}\mathit{trans} \qquad \begin{array}{c}[x\mathsf{P}y]\,[y\mathsf{P}y]\\ \vdots\\ \dfrac{\alpha}{\alpha}\ \mathit{class}\end{array}$$

$$\frac{\alpha(x) \quad x\mathsf{P}x \quad x\mathsf{P}y}{\alpha(y/x)}\ \mathsf{P}\mathit{sub1} \qquad \frac{\alpha(y) \quad x\mathsf{P}x \quad x\mathsf{P}y}{\alpha(x/y)}\ \mathsf{P}\mathit{sub2}$$

In $\bigstar I$, y is fresh: it is different from x and does not occur in any assumption on which $y : A$ depends other than $x\mathsf{R}y$.
In class, y is fresh: it is different from x and does not occur in α nor in any assumption on which α depends other than $x\mathsf{P}y$ and $y\mathsf{P}y$.
We refer to the fresh y in $\bigstar I$ and class as the *parameter* of the rule.

Fig. 8.4. The rules of MSpQS

$$\dfrac{\dfrac{\dfrac{\dfrac{[x : \boxdot\neg(A \supset \boxdot A)]^2 \quad [x\mathsf{P}y]^1}{y : \neg(A \supset \boxdot A)}\ \boxdot E \qquad \dfrac{\dfrac{\dfrac{[y : A]^3 \quad [y\mathsf{P}y]^1 \quad [y\mathsf{P}z]^4}{z : A}\ \mathsf{P}\mathit{sub1}}{y : \boxdot A}\ \boxdot I^4}{y : A \supset \boxdot A}\ \supset I^3}{y : \bot}\ \neg E}{x : \neg\boxdot\neg(A \supset \boxdot A)}\ \neg I^2}{x : \neg\boxdot\neg(A \supset \boxdot A)}\ \mathit{class}^1$$

Fig. 8.5. An example proof in MSpQS

Derivations and proofs in MSpQS are defined as for MSQS. For instance, in addition to the formulas for $\Box$ already listed for MSQS, the following labeled formula schemata are all provable in MSpQS (as shown, e.g., for formula 3 in Figure 8.5):

1. $x : \boxdot A \supset \Diamonddot A$
   (it is always possible to perform a measurement of a quantum state).
2. $x : \boxdot A \supset \boxdot\boxdot A$
   (measurements are composable).
3. $x : \Diamonddot(A \supset \boxdot A)$, i.e. $x : \neg\boxdot\neg(A \supset \boxdot A)$
   (it is always possible to perform a measurement with a complete reduction of a quantum state to a classical one).

The semantics is also obtained by simple changes with respect to the definitions of Section 8.3. A *frame* is a tuple $\mathscr{F} = \langle W, U, P\rangle$, where $P \subseteq W \times W$ and vPw means that w is obtained by means of a measurement of v, with the following properties:

(i) $\forall v, w.\ vPw \Longrightarrow vUw$
(as for (i) in Section 8.3).

(ii) $\forall v, w', w''.\ vPw' \ \&\ w'Pw'' \Longrightarrow vPw''$
(measurements are composable).

(iii) $\forall v.\ \exists w.\ vPw \ \&\ wPw$
(each quantum state v can be reduced to a classical one w by means of a measurement).

(iv) $\forall v, w.\ vPv \ \&\ vPw \Longrightarrow v = w$
(each measurement of a classical state v has v as outcome).

Models and *structures* are defined as before, with $\mathscr{I}(\mathsf{P}) = P$, while the *truth* relation now comprises the clauses

$$\models^{\mathscr{M},w} \boxdot A \ \text{ iff } \forall w'.\ wPw' \Longrightarrow \models^{\mathscr{M},w'} A$$
$$\models^{\mathscr{M},\mathscr{I}} x\mathsf{P}y \text{ iff } \mathscr{I}(x)P\mathscr{I}(y)$$

Finally, MSpQS is also sound and complete.

Theorem 8.11 (Soundness and completeness of MSpQS). $\Gamma \vdash \alpha$ *iff* $\Gamma \models \alpha$. □

We can reason similarly to what we did for MSQS to show the soundness and completeness of MSpQS with respect to the corresponding semantics: Theorem 8.11 follows from Theorems 8.12 and 8.13 below.

Theorem 8.12 (Soundness of MSpQS). $\Gamma \vdash \alpha$ *implies* $\Gamma \models \alpha$.

Proof. We let $\mathscr{M}$ be an arbitrary model and prove that if $\Gamma \vdash \alpha$ then $\models^{\mathscr{I}} \Gamma$ implies $\models^{\mathscr{I}} \alpha$ for any $\mathscr{I}$. The proof proceeds by induction on the structure of the derivation of α from Γ. The base case, where $\alpha \in \Gamma$, is trivial. There is one step case for each rule of MSpQS, where the soundness of the rules $\supset I$, $\supset E$, RAA, $\bot E$, U*refl*, U$symm$, U$trans$ follows exactly like in the proof of Theorem 8.5.

The soundness of the rules $\bigstar I$ and $\bigstar E$ follows exactly like in the proof of Theorem 8.5, with the only difference that when $\bigstar$ is $\boxdot$ then R is P.

The rule PUI is sound by property (i) in the definition of the semantics for MSpQS.

The rule P$trans$ is sound by property (ii) in the definition of the semantics for MSpQS.

The soundness of the rule $class$ follows like for the soundness of the rule Mser in the proof of Theorem 8.5, this time exploiting property (iii) in the definition of the semantics for MSpQS.

The soundness of the rules P$sub1$ and P$sub2$ follows like for the soundness of the rules M$sub1$ and M$sub2$ in the proof of Theorem 8.5, this time exploiting property (iv) in the definition of the semantics for MSpQS.

To prove completeness (Theorem 8.10), we proceed like for MSQS, mutatis mutandis in the construction of the canonical model. In particular, given a maximal consistent context (LF^*, RF^*), we define the canonical structure $\mathscr{S}^c = \langle W^c, U^c, P^c, V^c, \mathscr{I}^c \rangle$ by setting

- $(a_i, a_j) \in P^c$ iff $a_i \mathsf{P} a_j \in (LF^*, RF^*)$.

To show that the conditions on R^c are satisfied, so that $\mathscr{S}^c$ is really a structure for MSpQS, we reuse the results proved for MSQS and in addition show the following:

- $\forall v, w \in W^c.\ vPw \Longrightarrow vUw$ holds by construction and rule PUI.

- $\forall v, w', w'' \in W^c.\ vPw' \ \&\ w'Pw'' \implies vPw''$ holds by construction and rule $\mathsf{P}trans$.
- $\forall v \in W^c.\ \exists w \in W^c.\ vPw \ \&\ wPw$ holds by construction and rule $class$. For the sake of contradiction, consider an arbitrary a_i and a variable a'_j that do not satisfy the property. Define $(LF', RF') = (LF^*, RF^*) \cup \{a_i \mathsf{P} a'_j, a'_j \mathsf{P} a'_j\}$. Then it cannot be the case that $(LF', RF') \vdash \alpha$, for otherwise $(LF^*, RF^*) \vdash \alpha$ would be derivable by an application of the rule $class$. Thus, $(LF', RF') \nvdash \alpha$. But then (LF', RF') must be in the chain of contexts built in Lemma 8.7. So, by the maximality of (LF^*, RF^*), we have that $(LF', RF') = (LF^*, RF^*)$, contradicting our assumption. Hence, for some a_j, the r-formulas $a_i \mathsf{P} a_j$ and $a_j \mathsf{P} a_j$ are both in (LF^*, RF^*), which is what we had to show.
- $\forall v, w \in W^c.\ vPv \ \&\ vPw \implies v = w$ holds by construction and rules $\mathsf{P}sub1$ and $\mathsf{P}sub2$ since v is a classical world. Consider an arbitrary context (LF, RF) from which we build $\mathscr{S}^c$ and assume $(a_i, a_i) \in P^c$ and $(a_i, a_j) \in P^c$. Then $a_i \mathsf{P} a_i \in (LF^*, RF^*)$ and $a_i \mathsf{P} a_j \in (LF^*, RF^*)$. Thus, for each $a_i : A \in (LF^*, RF^*)$, we also have $a_j : A \in (LF^*, RF^*)$; otherwise, since (LF^*, RF^*) is deductively closed, we would have $a_j : \neg A \in (LF^*, RF^*)$ and also $a_j : A \in (LF^*, RF^*)$ by *1* in Lemma 8.7 and rule $\mathsf{P}sub1$, and thus a contradiction. Similarly, if $a_j : A \in (LF^*, RF^*)$ then $a_i : A \in (LF^*, RF^*)$ by rule $\mathsf{P}sub2$. Hence, for each m-formula A, we have that $a_i : A \in (LF^*, RF^*)$ iff $a_j : A \in (LF^*, RF^*)$, which means that a_i and a_j are equal with respect to m-formulas.
  Under the same assumptions, we can similarly show that a_i and a_j are equal with respect to r-formulas, i.e. that whenever (LF^*, RF^*) contains an r-formula that includes a_i then it also contains the same r-formula with a_j substituted for a_i, and vice versa. To this end, we must consider eight different cases corresponding to eight different r-formulas.
  1. If $a_k \mathsf{U} a_i \in (LF^*, RF^*)$ for some a_k, then from the assumption that $a_i \mathsf{P} a_j \in (LF^*, RF^*)$ we have $a_i \mathsf{U} a_j \in (LF^*, RF^*)$, by *1* in Lemma 8.7 and rule $\mathsf{PU}I$. Therefore, $a_k \mathsf{U} a_j \in (LF^*, RF^*)$ by rule $\mathsf{U}trans$.
  2. We can reason similarly for $a_j \mathsf{U} a_k \in (LF^*, RF^*)$ and also apply rules $\mathsf{PU}I$ and $\mathsf{U}trans$ to conclude that then also $a_i \mathsf{U} a_k \in (LF^*, RF^*)$.
  3. If $a_i \mathsf{U} a_k \in (LF^*, RF^*)$ for some a_k, then from the assumption that $a_i \mathsf{P} a_j \in (LF^*, RF^*)$ we have $a_i \mathsf{U} a_j \in (LF^*, RF^*)$, by *1* in Lemma 8.7 and rule $\mathsf{PU}I$, and thus $a_j \mathsf{U} a_i \in (LF^*, RF^*)$, by rule $\mathsf{U}symm$. Therefore, $a_j \mathsf{U} a_k \in (LF^*, RF^*)$ by rule $\mathsf{U}trans$.
  4. We can reason similarly for $a_k \mathsf{U} a_j \in (LF^*, RF^*)$ and also apply rules $\mathsf{PU}I$, $\mathsf{U}symm$, and $\mathsf{U}trans$ to conclude that then also $a_k \mathsf{U} a_i \in (LF^*, RF^*)$.
  5. If $a_k \mathsf{P} a_i \in (LF^*, RF^*)$ for some a_k, then from the assumption that $a_i \mathsf{P} a_j \in (LF^*, RF^*)$ we have $a_k \mathsf{P} a_j \in (LF^*, RF^*)$, by *1* in Lemma 8.7 and the rule $\mathsf{P}trans$.
  6. We can reason similarly for $a_j \mathsf{P} a_k \in (LF^*, RF^*)$ and also apply rule $\mathsf{P}trans$ to conclude that then also $a_i \mathsf{U} a_k \in (LF^*, RF^*)$.
  7. If $a_i \mathsf{P} a_k \in (LF^*, RF^*)$ for some a_k, then from the assumptions that $a_i \mathsf{P} a_i \in (LF^*, RF^*)$ and $a_i \mathsf{P} a_j \in (LF^*, RF^*)$ we have $a_j \mathsf{P} a_k \in (LF^*, RF^*)$, by *1* in Lemma 8.7 and rule $\mathsf{P}sub1$.
  8. We can reason similarly for $a_k \mathsf{P} a_j \in (LF^*, RF^*)$ and apply rule $\mathsf{P}sub2$ to conclude that then also $a_k \mathsf{P} a_i \in (LF^*, RF^*)$.

Hence, a_i and a_j are equal also with respect to r-formulas, and thus $a_i = a_j$ whenever $(a_i, a_i) \in P^c$ and $(a_i, a_j) \in P^c$, which is what we had to show.

Proceeding like for MSQS, we then have:

Theorem 8.13 (Completeness of MSpQS). *$\Gamma \models \alpha$ implies $\Gamma \vdash \alpha$.* □

8.5 Normalization

In this section, we show that each derivation of an l-formula in MSQS and MSpQS can be reduced to a normal form that does not contain unnecessary detours and satisfies a subformula property, from which we then obtain syntactic proofs of the consistency of both MSQS and MSpQS. We first consider MSQS and then discuss the extensions and changes needed in the case of MSpQS.

8.5.1 Normalization for MSQS

We begin by proving a useful lemma about parameters, i.e., as we mentioned above, the fresh variables used in the applications of $\bigstar I$[5] and Mser. By extension, we speak of a parameter y of a derivation if y is the parameter of some application of $\bigstar I$ or Mser in the derivation.

Lemma 8.14 (Parameter condition). *Let Π be an MSQS-derivation of $x : A$ from a set Γ of assumptions. Then we can build an MSQS-derivation Π' of $x : A$ from Γ such that:*

- *each parameter is the parameter of exactly one application of $\bigstar I$ or Mser, and*
- *the parameter of any application of $\bigstar I$ or Mser occurs only in the sub-derivation above that application of the rule.*

Proof. The lemma follows quite straightforwardly by induction on the derivation of $\Gamma \vdash x : A$, where the proof essentially boils down to a systematic renaming of the parameters.

In the remainder of the thesis, we thus assume that all the derivations satisfy the parameter condition.

To show normalization, we follow, where possible, standard presentations such as [79, 80, 95]. We begin by introducing some restrictions to simplify the development; in particular, we restrict applications of RAA and $\bot E$ to the case where the conclusion $x : A$ is atomic, i.e. A is atomic.[6] Moreover, we also restrict applications of M$sub1$, M$sub2$ ans Mser to atomic conclusions.

[5] We recall the convention stated in Section 8.2.2: if $\bigstar$ is $\square$ then R is U, if $\bigstar$ is $\blacksquare$ then R is M.

[6] When presenting classical first-order logic, Prawitz [79] first introduces a natural deduction system consisting of an elimination rule for $\bot$ and introduction and elimination rules for all the other connectives, and then, to show normalization, restricts his attention to the functionally complete $\bot$, $\wedge$, $\supset$, $\forall$ fragment, where RAA is restricted to atomic conclusions (that are also different from $\bot$). In this way, he avoids having to treat the rules for $\vee$ and $\exists$, which behave 'badly' for normalization. Here, since we have already focused on the functionally complete $\bot$, $\supset$, $\bigstar$ system, we do not need further restrictions than the ones on RAA and $\bot E$ (where however we allow the atomic conclusion A to be falsum itself, albeit labeled differently), as well as on M$sub1$, M$sub2$, and Mser.

Lemma 8.15. *If $\Gamma \vdash \alpha$ in MSQS, then there is an MSQS-derivation of α from Γ where the conclusions of applications of RAA, $\bot E$, M$sub1$, M$sub2$, and Mser are atomic.*

Note that we do not need to consider derivations of r-formulas, e.g. by $\bot E$, since in MSQS we only have atomic r-formulas by definition; the same holds for MSpQS. We can then prove the above lemma as follows:

Proof. We first show that any application of RAA with a non-atomic conclusion can be replaced with a derivation in which RAA is applied only to l-formulas of smaller grade. Note that we only show the part of the derivation where the transformation, denoted by $\rightsquigarrow$, actually takes place; the missing parts remain unchanged.

There are two possible cases, depending on whether the conclusion is $x : B \supset C$ or $x : \bigstar B$.

Case 1: We distinguish two subcases, depending on whether C is $\bot$ or not. If $C \neq \bot$, then

$$\dfrac{\begin{array}{c}[x:(B\supset C)\supset\bot]^1\\ \Pi\\ y:\bot\end{array}}{x:B\supset C}\;RAA^1 \quad\rightsquigarrow\quad \dfrac{\dfrac{\begin{array}{c}\dfrac{\dfrac{[x:C\supset\bot]^2 \qquad \dfrac{[x:B\supset C]^1 \quad [x:B]^3}{x:C}\;\supset E}{x:\bot}\;\supset E}{x:(B\supset C)\supset\bot}\;\supset I^1\\ \Pi\\ y:\bot\end{array}}{x:C}\;RAA^2}{x:B\supset C}\;\supset I^3$$

If $C = \bot$, then

$$\dfrac{\begin{array}{c}[x:(B\supset\bot)\supset\bot]^1\\ \Pi\\ y:\bot\end{array}}{x:B\supset\bot}\;RAA^1 \quad\rightsquigarrow\quad \dfrac{\dfrac{\begin{array}{c}\dfrac{\dfrac{[x:B\supset\bot]^1 \quad [x:B]^2}{x:\bot}\;\supset E}{x:(B\supset\bot)\supset\bot}\;\supset I^1\\ \Pi\\ y:\bot\end{array}}{x:\bot}\;\bot E}{x:B\supset\bot}\;\supset I^2$$

Case 2: We distinguish two subcases, depending on whether B is $\bot$ or not. If $B \neq \bot$, then

$$\dfrac{\begin{array}{c}[x:\bigstar B\supset\bot]^1\\ \Pi\\ y:\bot\end{array}}{x:\bigstar B}\;RAA^1 \quad\rightsquigarrow\quad \dfrac{\dfrac{\begin{array}{c}\dfrac{\dfrac{\dfrac{[y:B\supset\bot]^2 \qquad \dfrac{[x:\bigstar B]^1 \quad [x\mathsf{R}y]^3}{y:B}\;\bigstar E}{y:\bot}\;\supset E}{x:\bot}\;\bot E}{x:\bigstar B\supset\bot}\;\supset I^1\\ \Pi\\ y:\bot\end{array}}{y:B}\;RAA^2}{x:\bigstar B}\;\bigstar I^3$$

where, if necessary, we follow Lemma 8.14 to rename the parameters in the derivation to avoid possible clashes due to the new application of $\bigstar I$.

If $B = \bot$, then

$$\dfrac{\begin{array}{c}[x : \bigstar\bot \supset \bot]^1 \\ \Pi \\ y : \bot\end{array}}{x : \bigstar\bot}\; RAA^1 \quad \rightsquigarrow \quad \dfrac{\begin{array}{c}\dfrac{\dfrac{\dfrac{[x : \bigstar\bot]^1 \quad [x\mathsf{R}y]^2}{y : \bot}\supset E}{x : \bot}\bot E}{x : \bigstar\bot \supset \bot}\supset I^1 \\ \Pi \\ y : \bot\end{array}}{x : \bigstar B}\; \bigstar I^2$$

We proceed analogously for $\bot E$: we show that any application of $\bot E$ with a non-atomic conclusion can be replaced with a derivation in which $\bot E$ is applied only to l-formulas of smaller grade. Hence, there are again two possible cases, depending on whether the conclusion is $x : B \supset C$ or $x : \bigstar B$.
Case 1:

$$\dfrac{\begin{array}{c}\Pi \\ y : \bot\end{array}}{x : B \supset C}\; \bot E \quad \rightsquigarrow \quad \dfrac{\dfrac{\begin{array}{c}\Pi \\ y : \bot\end{array}}{x : C}\; \bot E}{x : B \supset C}\; \supset I$$

Case 2:

$$\dfrac{\begin{array}{c}\Pi \\ y : \bot\end{array}}{x : \bigstar B}\; \bot E \quad \rightsquigarrow \quad \dfrac{\dfrac{\begin{array}{c}\Pi \\ y : \bot\end{array}}{z : B}\; \bot E}{x : \bigstar B}\; \bigstar I$$

Applications of $\mathsf{M}sub1$ and $\mathsf{M}sub2$ can be reduced to atomic formulas as follows, where we now consider the two subcases for $\Box$ and $\blacksquare$ explicitly:

$$\dfrac{\begin{array}{c}\Pi_1 \\ x : A \supset B\end{array} \quad \begin{array}{c}\Pi_2 \\ x\mathsf{M}x\end{array} \quad \begin{array}{c}\Pi_3 \\ x\mathsf{M}y\end{array}}{y : A \supset B}\; \mathsf{M}sub1 \qquad \rightsquigarrow$$

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\Pi_1 \\ x : A \supset B\end{array} \quad \dfrac{[y : A]^1 \quad \begin{array}{c}\Pi_2 \\ x\mathsf{M}x\end{array} \quad \begin{array}{c}\Pi_3 \\ x\mathsf{M}y\end{array}}{x : A}\; \mathsf{M}sub2}{x : B}\; \supset E \qquad \begin{array}{c}\Pi_2 \\ x\mathsf{M}x\end{array} \quad \begin{array}{c}\Pi_3 \\ x\mathsf{M}y\end{array}}{y : B}\; \mathsf{M}sub1}{y : A \supset B}\; \supset I^1$$

$$\dfrac{\begin{matrix}\Pi_1\\ y : A \supset B\end{matrix} \quad \begin{matrix}\Pi_2\\ x\mathsf{M}x\end{matrix} \quad \begin{matrix}\Pi_3\\ x\mathsf{M}y\end{matrix}}{x : A \supset B}\,\mathsf{M}sub2 \quad \rightsquigarrow$$

$$\dfrac{\dfrac{\dfrac{\begin{matrix}\Pi_1\\ y : A \supset B\end{matrix} \quad \dfrac{[x : A]^1 \quad \begin{matrix}\Pi_2\\ x\mathsf{M}x\end{matrix} \quad \begin{matrix}\Pi_3\\ x\mathsf{M}y\end{matrix}}{y : A}\,\mathsf{M}sub1}{y : B}\,\supset E \quad \begin{matrix}\Pi_2\\ x\mathsf{M}x\end{matrix} \quad \begin{matrix}\Pi_3\\ x\mathsf{M}y\end{matrix}}{x : B}\,\mathsf{M}sub2}{x : A \supset B}\,\supset I^1$$

$$\dfrac{\begin{matrix}\Pi_1\\ x : \Box A\end{matrix} \quad \begin{matrix}\Pi_2\\ x\mathsf{M}x\end{matrix} \quad \begin{matrix}\Pi_3\\ x\mathsf{M}y\end{matrix}}{y : \Box A}\,\mathsf{M}sub1 \quad \rightsquigarrow \quad \dfrac{\dfrac{\begin{matrix}\Pi_1\\ x : \Box A\end{matrix} \quad \dfrac{[y\mathsf{U}z]^1 \quad \begin{matrix}\Pi_2\\ x\mathsf{M}x\end{matrix} \quad \begin{matrix}\Pi_3\\ x\mathsf{M}y\end{matrix}}{x\mathsf{U}z}\,\mathsf{M}sub2}{z : A}\,\Box E}{y : \Box A}\,\Box I^1$$

$$\dfrac{\begin{matrix}\Pi_1\\ y : \Box A\end{matrix} \quad \begin{matrix}\Pi_2\\ x\mathsf{M}x\end{matrix} \quad \begin{matrix}\Pi_3\\ x\mathsf{M}y\end{matrix}}{x : \Box A}\,\mathsf{M}sub2 \quad \rightsquigarrow \quad \dfrac{\dfrac{\begin{matrix}\Pi_1\\ y : \Box A\end{matrix} \quad \dfrac{[x\mathsf{U}z]^1 \quad \begin{matrix}\Pi_2\\ x\mathsf{M}x\end{matrix} \quad \begin{matrix}\Pi_3\\ x\mathsf{M}y\end{matrix}}{y\mathsf{U}z}\,\mathsf{M}sub1}{z : A}\,\Box E}{x : \Box A}\,\Box I^1$$

$$\dfrac{\begin{matrix}\Pi_1\\ x : \blacksquare A\end{matrix} \quad \begin{matrix}\Pi_2\\ x\mathsf{M}x\end{matrix} \quad \begin{matrix}\Pi_3\\ x\mathsf{M}y\end{matrix}}{y : \blacksquare A}\,\mathsf{M}sub1 \quad \rightsquigarrow \quad \dfrac{\dfrac{\begin{matrix}\Pi_1\\ x : \blacksquare A\end{matrix} \quad \dfrac{[y\mathsf{M}z]^1 \quad \begin{matrix}\Pi_2\\ x\mathsf{M}x\end{matrix} \quad \begin{matrix}\Pi_3\\ x\mathsf{M}y\end{matrix}}{x\mathsf{M}z}\,\mathsf{M}sub2}{z : A}\,\blacksquare E}{y : \blacksquare A}\,\blacksquare I^1$$

$$\dfrac{\begin{matrix}\Pi_1\\ y : \blacksquare A\end{matrix} \quad \begin{matrix}\Pi_2\\ x\mathsf{M}x\end{matrix} \quad \begin{matrix}\Pi_3\\ x\mathsf{M}y\end{matrix}}{x : \blacksquare A}\,\mathsf{M}sub2 \quad \rightsquigarrow \quad \dfrac{\dfrac{\begin{matrix}\Pi_1\\ y : \blacksquare A\end{matrix} \quad \dfrac{[x\mathsf{M}z]^1 \quad \begin{matrix}\Pi_2\\ x\mathsf{M}x\end{matrix} \quad \begin{matrix}\Pi_3\\ x\mathsf{M}y\end{matrix}}{y\mathsf{M}z}\,\mathsf{M}sub1}{z : A}\,\blacksquare E}{x : \blacksquare A}\,\blacksquare I^1$$

We proceed in the same way for the $\mathsf{M}ser$ rule.

Case 1:

$$\dfrac{\begin{matrix}[x\mathsf{M}y]^1\\ \Pi\\ u : B \supset C\end{matrix}}{u : B \supset C}\,\mathsf{M}ser^1 \quad \rightsquigarrow \quad \dfrac{\dfrac{\dfrac{\begin{matrix}[x\mathsf{M}y]^1\\ \Pi\\ u : B \supset C\end{matrix} \quad [u : B]^2}{u : C}\,\supset E}{u : C}\,\mathsf{M}ser^1}{u : B \supset C}\,\supset I^2$$

Case 2:

$$\cfrac{\begin{array}{c}[x\mathsf{M}y]^1\\ \varPi\\ u:\bigstar A\end{array}}{x:\bigstar A}\,\mathsf{M}ser^1 \qquad\rightsquigarrow\qquad \cfrac{\cfrac{\cfrac{\begin{array}{c}[x\mathsf{M}y]^1\\ \varPi\\ u:\bigstar A\end{array}\quad [u\mathsf{R}w]^2}{w:A}\,\bigstar E}{w:A}\,\mathsf{M}ser^1}{u:\bigstar A}\,\bigstar I^2$$

where we choose the parameter w so to allow for the application of $\bigstar I$.

By iterating these transformations, we transform an arbitrary MSQS-derivation $\varGamma \vdash \alpha$ into an MSQS-derivation of α from $\varGamma$ where the conclusions of applications of RAA, $\perp E$, $\mathsf{M}sub1$, $\mathsf{M}sub2$, and $\mathsf{M}ser$ are atomic.

An immediate consequence of this lemma is the equivalence of the restricted and the unrestricted natural deduction systems. In the rest of this section, we will therefore assume applications of RAA, $\perp E$, $\mathsf{M}sub1$, $\mathsf{M}sub2$, and $\mathsf{M}ser$ to be restricted in this way.

In a generic derivation, we can have a detour caused by the application of an elimination rule immediately below the application of the corresponding introduction rule. That is, if an l-formula is introduced and then immediately eliminated, then we can avoid introducing it in the first place; recall that in MSQS we only have atomic r-formulas by definition, so we do not need to consider the detours that would arise from non-atomic r-formulas. Formally, since the same formula may appear several times in a derivation, we distinguish these different formula occurrences to define:

Definition 8.16. *An l-formula occurrence $x : A$ is a* cut *in an MSQS-derivation when it is both the conclusion of an introduction rule and the major premise of an elimination rule. We call $x : A$ the* cut-formula *of the cut.*

An MSQS-derivation is in normal form *(is a* normal MSQS-derivation*) iff it contains no cut-formulas.*

Like for any "good" deduction system, we prove a normalization result that shows how to transform (in an effective way) each MSQS-derivation into a normal one. In order to remove cut-formulas, we introduce the notion of *contraction*, where the contraction relation $\triangleright$ is defined as follows:

$$\cfrac{\cfrac{\begin{array}{c}[x:A]\\ \varPi_1\\ x:B\end{array}}{x:A\supset B}\supset I \qquad \begin{array}{c}\varPi_2\\ x:A\end{array}}{x:B}\supset E \quad\triangleright\quad \begin{array}{c}\varPi_2\\ x:A\\ \varPi_1\\ x:B\end{array} \qquad (\triangleright_{\supset})$$

$$\cfrac{\cfrac{\begin{array}{c}[x\mathsf{R}y]\\ \varPi_1\\ y:A\end{array}}{x:\bigstar A}\bigstar I \qquad \begin{array}{c}\varPi_2\\ x\mathsf{R}z\end{array}}{z:A}\bigstar E \quad\triangleright\quad \begin{array}{c}\varPi_2\\ x\mathsf{R}z\\ \varPi_1[z/y]\\ z:A\end{array} \qquad (\triangleright_{\bigstar})$$

where $\varPi'[z/y]$ is obtained from $\varPi$ by systematically substituting z for y. Note that the correctness of the contractions, and also of the substitution $\varPi'[z/y]$, is guaranteed by the assumption that all the derivations satisfy the parameter condition of Lemma 8.14. Note also that it suffices to consider the generic modal operator $\bigstar$ since the two modal operators $\square$ and $\blacksquare$ do not interfere (nor do the corresponding contractions).

Cuts are removed from a derivation by finitely many applications of contractions. Context closure of the contraction relation leads to the formal definition of the notions of *reduction* and *normalization*.

Definition 8.17. *We say that an* MSQS*-derivation Π_1* immediately reduces *to an* MSQS*-derivation Π_2, in symbols $\Pi_1 \succ \Pi_2$, iff there exist* MSQS*-derivations Π_3 and Π_4 such that $\Pi_3 \rhd \Pi_4$ and Π_2 is obtained by replacing Π_3 with Π_4 in Π_1 .*

Hence, if Π is a normal MSQS-derivation (i.e. it contains no cut-formulas), there is no Π' such that $\Pi \succ \Pi'$.

Definition 8.18. *Writing $\succeq$ for the reflexive and transitive closure of $\succ$, we say that an* MSQS*-derivation Π* normalizes *to another* MSQS*-derivation Π' if $\Pi \succeq \Pi'$ and Π' is in normal form.*

Definition 8.19. *We define the* rank *$rank$ of an l-formula as $rank(x : A) = rank(A)$ where*

- $rank(A) = 0$ *if A is atomic;*
- $rank(A \supset B) = max\{rank(A), rank(B)\} + 1$*;*
- $rank(\bigstar A) = rank(A) + 1$.

Then, for Π a derivation in MSQS*,*

- *a* maximal cut-formula *in Π is a cut-formula in Π with maximal rank;*
- $d = max\{rank(x : A) \mid x : A$ *is a cut-formula in* $\Pi\}$*, where* $\max\{\} = 0$*;*
- $cr(\Pi) = (d, n)$ *is the* cut rank *of Π, where n is the number of maximal cut-formulas in Π and where $cr(\Pi) = (0, 0)$ when Π has no cuts.*

The ordering on cut ranks is lexicographic: $(d, n) < (d', n')$ iff $d < d'$ or both $d = d'$ and $n < n'$. To prove our normalization result, we will systematically lower the cut rank of a derivation until all cuts have been eliminated. Before we do that, we prove a useful lemma:

Lemma 8.20. *Let Π be an* MSQS*-derivation with a cut at the bottom, and let this cut have rank q while all the other cuts in Π have rank $< q$. Then the contraction of Π at this lowest cut yields a derivation with only cuts of rank $< q$.*

Proof. Consider all the possible cuts at the bottom of Π and check the ranks of the cuts after the contraction. The proof follows since the two contractions ($\rhd_{\supset}$) and ($\rhd_{\bigstar}$) explicitly give formulas with lower rank, while nothing happens in Π_1 and Π_2, so all the cuts in the derivation resulting from the contraction have rank $< q$.

Lemma 8.21. *Let Π be an* MSQS*-derivation. If $cr(\Pi) > (0, 0)$, then there is an* MSQS*-derivation Π' with $\Pi \rhd \Pi'$ and $cr(\Pi') < cr(\Pi)$.*

Proof. Select a maximal cut-formula in Π such that all cuts above it have lower rank. Apply the appropriate contraction to this maximal cut. Then the part of Π ending in the conclusion of the cut is replaced, by Lemma 8.20, by a sub-derivation in which all cut-formulas have lower rank. If the maximal cut-formula was the only one, then d is lowered by 1, otherwise n is lowered by 1 and d remains unchanged. In both cases, $cr(\Pi)$ gets smaller. (Note that in the first case n may become much larger, but that does not matter in the lexicographic order.)

We are now in a position to give our desired normalization results.

Theorem 8.22. *Every MSQS-derivation of $x : A$ from Γ reduces to an MSQS-derivation in normal form.*

Proof. By Lemma 8.21, the cut rank can be lowered to $(0, 0)$ in a finite number of steps, hence the last derivation in the reduction sequence has no more cuts.

Normal MSQS-derivations possess a well-defined structure that has several desirable properties. Specifically, by analyzing the structure of a normal MSQS-derivation, we can characterize its form: we can identify particular sequences of formulas, and show that in these sequences there is an ordering on inferences. By exploiting this ordering, we can then show a subformula property for MSQS.

Definition 8.23. *A* thread *in an MSQS-derivation Π is a sequence of formulas $\alpha_1, \ldots, \alpha_n$ such that (i) α_1 is an assumption of Π, (ii) α_i stands immediately above α_{i+1}, for $1 \leq i < n-1$, and (iii) α_n is the conclusion of Π.*

We further characterize a thread in terms of the formulas occurring in it: an l-formula-thread *is a thread where $\alpha_1, \ldots, \alpha_n$ are all l-formulas, and an* r-formula-thread *is a thread where $\alpha_1, \ldots, \alpha_n$ are all r-formulas.*

A track *in an MSQS-derivation Π is an initial part of a thread in Π which stops either at the first minor premise of an elimination rule in the thread or at the conclusion of the thread. We call* main track *a track that is also a thread and ends at the conclusion of the derivation.*

Definition 8.24. *B is a* subformula *of A iff (i) A is B; or (ii) A is $A_1 \supset A_2$ and B is a subformula of A_1 or A_2; or (iii) A is $\bigstar A_1$ and B is a subformula of A_1. We say that $y : B$ is a* labeled subformula *(or, slightly abusing notation, simply "subformula") of $x : A$ iff B is a subformula of A.*

One interesting property of normal MSQS-derivations, which can be read off from their structure, is that tracks in a normal MSQS-derivation have a standard form:

Lemma 8.25. *Let Π be a normal MSQS-derivation, and let t be a track $\alpha_1, \ldots, \alpha_n$ in Π. Then t contains a subsequence of formulas $\alpha_i, \ldots, \alpha_k$, called the* minimal part*, which separates two possibly empty parts of t, called the* elimination part *and the* introduction part *of t, where:*

- *each formula α_j in the elimination part, i.e. for $j < i$, is an l-formula and is the major premise of an application of an elimination rule and contains α_{j+1} as a subformula;*
- *each formula α_s in the minimal part except the last one is the premise of an application of RAA, $\bot E$, Msub1, Msub2, Mser, Urefl, Usymm, Utrans, UI, or Msrefl;*
- *each formula α_j in the introduction part except the last one, i.e. for $k < j < n$, is an l-formula, is a premise of an introduction rule, and is a subformula of α_{j+1};*
- *Π has at least one main track, ending in the conclusion.*

The lemma follows quite straightforwardly by observing that in a track in a normal MSQS-derivation no introduction rule application can precede an application of an elimination rule; hence, if the first rule is an elimination, then all eliminations come first.

From these considerations, we can derive some other properties of normal tracks. For example, we can observe that if a thread t has an r-formula as top formula, then t is an *r-formula–thread* and if we extract a track t' from t, then we have empty elimination and introduction parts. Moreover, let $\alpha_1, \ldots, \alpha_n$ be a thread and let $\alpha_1, \ldots, \alpha_i$ be l-formulas; if α_{i+1} is an r-formula, then all α_j, for $i < j \leq n$, are r-formulas.

We can further observe that a "mixed" track (i.e. a track consisting of l-formulas and r-formulas) has the following structure: an introduction part of l-formulas; a minimal part in which an r-formula is introduced by an application of $\perp E$ and a (possibly empty) sequence of applications of RAA, M$sub1$, M$sub2$, Mser, U$refl$, U$symm$, U$trans$, UI, M$srefl$; and an empty introduction part.

The above results allow us to show that normal derivations in MSQS satisfy the following subformula property.

Definition 8.26. *Given an MSQS-derivation Π of $x : A$ from a set Γ of assumptions, let $\mathcal{S}$ be the set of subformulas of the formulas in $\{C \mid z : C \in \Gamma \cup \{x : A\}$ for some $z\}$, i.e. $\mathcal{S}$ is the set consisting of the subformulas of the assumptions Γ and of the conclusion $x : A$.*

We say that Π satisfies the subformula property *iff for each l-formula occurrence $y : B$ in the derivation (i) $B \in \mathcal{S}$; or (ii) B is an assumption $D \supset \perp$ discharged by an application of RAA, where $D \in \mathcal{S}$; or (iii) B is an occurrence of $\perp$ obtained by $\supset E$ from an assumption $D \supset \perp$ discharged by an application of RAA, where $D \in \mathcal{S}$; or (iv) B is an occurrence of $\perp$ obtained by an application of $\perp E$.*

In other words, we define an MSQS-derivation to have the subformula property iff for all $y : B$ in the derivation, either B is a subformula of the assumptions or of the conclusion of the derivation, or B is the negation of such a subformula and is discharged by RAA, or B is an occurrence of $\perp$ immediately below the negation of a subformula, or B is an occurrence of $\perp$ immediately below another occurrence of $\perp$ that is labeled differently.

Theorem 8.27. *Every normal derivation of $x : A$ from Γ in MSQS satisfies the subformula property.*

Proof. We introduce an ordering of the tracks in a normal MSQS-derivation depending on their distance from the main track: the *order* of a track is $o(t_m) = 0$ for a main track t_m, and $o(t) = o(t')$ +1 if the end formula of a generic track t is a minor premise belonging to a major premise in t'.

Consider now an l-formula occurrence $y : B$ in a normal derivation Π of $x : A$ from Γ in MSQS. If $y : B$ occurs in the elimination part of its track t, then it is a subformula of the assumptions at the top of t. If not, then it is a subformula of the l-formula $z : C$ at the end of t. Hence, $z : C$ is a subformula of an l-formula $w : D$ of a track t_1 with $o(t_1)\langle o(t)$. Repeating the argument, we find that $y : B$ is a subformula of an assumption in Γ or of the conclusion $x : A$. This closes the case for all assumptions, so let us now consider the other formulas.

If $y : B$ is a subformula of a discharged assumption, then it must be a subformula of the resulting implicational l-formula in the case of an application of $\supset I$, or of the resulting l-formula in the case of an application of RAA, or (and these are the only exceptions) it is itself discharged by an application of RAA or it is $z : \perp$ (for some z) immediately following such an assumption or an application of $\perp E$.

In proof theory it is standard to give purely syntactical proofs of consistency. The consistency of MSQS follows as a corollary from previous results:

Corollary 8.28. *MSQS is consistent.*

Proof. Suppose, for the sake of contradiction, that $\vdash x : \bot$ in MSQS. Then there is a normal derivation ending in $\vdash x : \bot$ with all assumptions discharged. There is a track through the conclusion; in this track there are no introduction rules, so the top assumption is not discharged. Contradiction.

8.5.2 Normalization for MSpQS

We can again simplify the development by restricting applications of RAA and $\bot E$ to the case where the conclusion $x : A$ is atomic, and we can also restrict applications of $\mathsf{P}sub1$, $\mathsf{P}sub2$, and $class$ to atomic conclusions, where, as for MSQS, we do not need to consider derivations of r-formulas, e.g. by $\bot E$, since also in MSpQS we only have atomic r-formulas by definition.

Lemma 8.29. *If $\Gamma \vdash \alpha$ in MSpQS, then there is an MSpQS-derivation of α from Γ where the conclusions of applications of RAA, $\bot E$, $\mathsf{P}sub1$, $\mathsf{P}sub2$, and class are atomic.*

Recalling that the grade of an l-formula $x : A$ is the number of times $\supset$ and $\bigstar$ occur in A, where $\bigstar$ is either $\Box$ or $\boxdot$ for MSpQS, we can prove the lemma and thus the equivalence of the restricted and the unrestricted system MSpQS as follows.

Proof. By considering the same transformations employed for MSQS in Lemma 8.15, we can replace applications of RAA and $\bot E$ with non-atomic conclusions with derivations in which RAA and $\bot E$ are applied only to l-formulas of smaller grade.

Applications of $\mathsf{P}sub1$ and $\mathsf{P}sub2$ can be reduced to atomic formulas as follows, where we consider only the two subcases for $\Box$ and $\boxdot$, as the subcases for $\supset$ follow like those in Lemma 8.15:

$$\dfrac{\begin{array}{ccc}\Pi_1 & \Pi_2 & \Pi_3\\ x:\Box A & x\mathsf{P}x & x\mathsf{P}y\end{array}}{y:\Box A}\,\mathsf{P}\mathit{sub1} \quad\rightsquigarrow\quad \dfrac{\dfrac{\begin{array}{c}\Pi_1\\ x:\Box A\end{array} \quad \dfrac{[y\mathsf{U}z]^1 \quad \begin{array}{c}\Pi_2\\ x\mathsf{P}x\end{array} \quad \begin{array}{c}\Pi_3\\ x\mathsf{P}y\end{array}}{x\mathsf{U}z}\,\mathsf{P}\mathit{sub2}}{z:A}\,\Box E}{y:\Box A}\,\Box I^1$$

$$\dfrac{\begin{array}{ccc}\Pi_1 & \Pi_2 & \Pi_3\\ y:\Box A & x\mathsf{P}x & x\mathsf{P}y\end{array}}{x:\Box A}\,\mathsf{P}\mathit{sub2} \quad\rightsquigarrow\quad \dfrac{\dfrac{\begin{array}{c}\Pi_1\\ y:\Box A\end{array} \quad \dfrac{[x\mathsf{U}z]^1 \quad \begin{array}{c}\Pi_2\\ x\mathsf{P}x\end{array} \quad \begin{array}{c}\Pi_3\\ x\mathsf{P}y\end{array}}{y\mathsf{U}z}\,\mathsf{P}\mathit{sub1}}{z:A}\,\Box E}{x:\Box A}\,\Box I^1$$

$$\dfrac{\begin{array}{ccc}\Pi_1 & \Pi_2 & \Pi_3\\ x:\boxdot A & x\mathsf{P}x & x\mathsf{P}y\end{array}}{y:\boxdot A}\,\mathsf{P}\mathit{sub1} \quad\rightsquigarrow\quad \dfrac{\dfrac{\begin{array}{c}\Pi_1\\ x:\boxdot A\end{array} \quad \dfrac{[y\mathsf{P}z]^1 \quad \begin{array}{c}\Pi_2\\ x\mathsf{P}x\end{array} \quad \begin{array}{c}\Pi_3\\ x\mathsf{P}y\end{array}}{x\mathsf{P}z}\,\mathsf{P}\mathit{sub2}}{z:A}\,\boxdot E}{y:\boxdot A}\,\boxdot I^1$$

$$\dfrac{\begin{array}{ccc}\Pi_1 & \Pi_2 & \Pi_3\\ y:\boxdot A & x\mathsf{P}x & x\mathsf{P}y\end{array}}{x:\boxdot A}\,\mathsf{P}\mathit{sub2} \quad\rightsquigarrow\quad \dfrac{\dfrac{\begin{array}{c}\Pi_1\\ y:\boxdot A\end{array} \quad \dfrac{[x\mathsf{P}z]^1 \quad \begin{array}{c}\Pi_2\\ x\mathsf{P}x\end{array} \quad \begin{array}{c}\Pi_3\\ x\mathsf{P}y\end{array}}{y\mathsf{P}z}\,\mathsf{P}\mathit{sub1}}{z:A}\,\boxdot E}{x:\boxdot A}\,\boxdot I^1$$

For $class$, similarly to $\mathsf{M}ser$ in Lemma 8.15, we have

$$\dfrac{\begin{array}{c}[x\mathsf{P}y]^1\ [y\mathsf{P}y]^1\\ \Pi\\ u:B\supset C\end{array}}{u:B\supset C}\,class^1 \quad\rightsquigarrow\quad \dfrac{\dfrac{\dfrac{\begin{array}{c}[x\mathsf{P}x]^1\ [y\mathsf{P}y]^1\\ \Pi\\ u:B\supset C\end{array} \qquad [u:B]^2}{u:C}\,\supset E}{u:C}\,class^1}{u:B\supset C}\,\supset I^2$$

$$\dfrac{\begin{array}{c}[x\mathsf{P}y]^1\ [y\mathsf{P}y]^1\\ \Pi\\ u:\bigstar A\end{array}}{u:\bigstar A}\,class^1 \quad\rightsquigarrow\quad \dfrac{\dfrac{\dfrac{\begin{array}{c}[x\mathsf{P}y]^1\ [y\mathsf{P}y]^1\\ \Pi\\ u:\bigstar A\end{array} \qquad [u\mathsf{R}w]^2}{w:A}\,\bigstar E}{w:A}\,class^1}{w:A}\,\bigstar E^2$$

where we choose the parameter w so to allow for the application of $\bigstar I$.

By iterating these transformations, we transform an arbitrary MSpQS-derivation $\Gamma\vdash\alpha$ into an MSpQS-derivation of α from Γ where the conclusions of applications of RAA, $\perp E$, $\mathsf{P}\mathit{sub1}$, $\mathsf{P}\mathit{sub2}$ and $class$ are atomic.

The contractions that remove cut-formulas from a derivation in MSpQS are the same as the ones for MSQS, where in this case $\bigstar$ stands for $\Box$ and $\boxdot$. Hence, proceeding as in the previous section, mutatis mutandis, we obtain a normalization result for MSpQS and the corresponding consequences.

Theorem 8.30. *Every* MSpQS*-derivation of* $x : A$ *from* Γ *reduces to an* MSpQS*-derivation in normal form.* $\square$

Theorem 8.31. *Every normal derivation of* $x : A$ *from* Γ *in* MSpQS *satisfies the subformula property.* $\square$

Corollary 8.32. MSpQS *is consistent.* $\square$

References

1. Samson Abramsky. High-level methods for quantum computation and information. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science (LICS)*. IEEE Computer Society, 2004.
2. Samson Abramsky and Bob Coecke. Physical traces: Quantum vs. classical information processing. In *Proceedings of the 9th Conference on Category Theory and Computer Science (CTCS 2002)*, volume 69 of *Electronic Notes in Theoretical Computer Science*. Elsevier Science, 2003. Also arXiv:cs.CG/0207057.
3. Samson Abramsky and Bob Coecke. A categorical semantics of quantum protocols. In *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science (LICS)*. IEEE Computer Society, 2004. Also arXiv:quant-ph/0402130.
4. Samson Abramsky and Ross Duncan. A categorical quantum logic. *Math. Structures Comput. Sci.*, 16(3):469–489, 2006.
5. P. Adão and P. Mateus. A process algebra for reasoning about quantum security. In P. Selinger, editor, *Proceedings of the 3rd International Workshop on Quantum Programming Languages*, Electronic Notes in Theoretical Computer Science. Elsevier, 2005.
6. Dorit Aharonov, Wim van Dam, Julia Kempe, Zeph Landau, Seth Lloyd, and Oded Regev. Adiabatic quantum computation is equivalent to standard quantum computation. *SIAM J. Comput.*, 37(1):166–194 (electronic), 2007.
7. T. Altenkirch and J. Grattage. A functional quantum programming language. In *Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science*, 2005.
8. Thorsten Altenkirch, Jonathan Grattage, Juliana K. Vizzotto, and Amr Sabry. An algebra of pure quantum programming. In *3rd International Workshop on Quantum Programming Languages*, 2005. to appear in ENTCS.
9. Pablo Arrighi and Gilles Dowek. Linear-algebraic lambda-calculus: higher-order, encodings, and confluence. In Andrei Voronkov, editor, *RTA*, volume 5117 of *Lecture Notes in Computer Science*, pages 17–31. Springer, 2008.
10. A. Asperti. Light affine logic. In *Thirteenth Annual IEEE Symposium on Logic in Computer Science (Indianapolis, IN, 1998)*, pages 300–308. IEEE Computer Soc., Los Alamitos, CA, 1998.
11. Andrea Asperti and Luca Roversi. Intuitionistic light affine logic (proof-nets, normalization complexity, expressive power). *ACM Trans. Comput. Log.*, 3(1):1–38 (electronic), 2002.
12. Patrick Baillot and Virgile Mogbil. Soft lambda-calculus: a language for polynomial time computation. In *Foundations of software science and computation structures*, volume 2987 of *Lecture Notes in Comput. Sci.*, pages 27–41. Springer, Berlin, 2004.
13. Patrick Baillot and Kazushige Terui. A feasible algorithm for typing in elementary affine logic. In *Typed lambda calculi and applications*, volume 3461 of *Lecture Notes in Comput. Sci.*, pages 55–70. Springer, Berlin, 2005.

14. Alexandru Baltag and Sonja Smets. The logic of quantum programs. In *Proceedings of the 2nd QPL*, 2004.
15. Alexandru Baltag and Sonja Smets. LQP: the dynamic logic of quantum information. *Math. Structures Comput. Sci.*, 16(3):491–525, 2006.
16. H. P. Barendregt. *The lambda calculus*, volume 103 of *Studies in Logic and the Foundations of Mathematics*. North-Holland Publishing Co., Amsterdam, revised edition, 1984. Its syntax and semantics.
17. Jean-Louis Basdevant and Jean Dalibard. *Quantum mechanics*. Springer-Verlag, Berlin, 2005. Corrected second printing, With 1 CD-ROM by Manuel Joffre.
18. J. S. Bell. On the einstein-podolsky-rosen paradox. *Physics*, 1:195, 1964.
19. Stephen Bellantoni and Stephen Cook. A new recursion-theoretic characterization of the polytime functions. *Comput. Complexity*, 2(2):97–110, 1992.
20. Paul Benioff. The computer as a physical system: a microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines. *J. Statist. Phys.*, 22(5):563–591, 1980.
21. C. H. Bennett. Logical reversibility of computation. *IBM J. Res. Develop.*, 17:525–532, 1973.
22. Ethan Bernstein and Umesh Vazirani. Quantum complexity theory. *SIAM J. Comput.*, 26(5):1411–1473, 1997.
23. Garrett Birkhoff and John von Neumann. The logic of quantum mechanics. *Ann. of Math. (2)*, 37(4):823–843, 1936.
24. Olivier Bournez and Claude Kirchner. Probabilistic rewrite strategies. Applications to ELAN. In *Rewriting techniques and applications*, volume 2378 of *Lecture Notes in Comput. Sci.*, pages 252–266. Springer, Berlin, 2002.
25. Samuel R. Buss. *Bounded arithmetic*, volume 3 of *Studies in Proof Theory. Lecture Notes*. Bibliopolis, Naples, 1986.
26. Brian F. Chellas. *Modal Logic*. Cambridge University Press, 1980.
27. Alonzo Church. *The Calculi of Lambda-Conversion*. Annals of Mathematics Studies, no. 6. Princeton University Press, Princeton, N. J., 1941.
28. Alan Cobham. The intrinsic computational difficulty of functions. In *Logic, Methodology and Philos. Sci. (Proc. 1964 Internat. Congr.)*, pages 24–30. North-Holland, Amsterdam, 1965.
29. Ugo Dal Lago and Martin Hofmann. Quantitative models and implicit complexity. In *FSTTCS 2005: Foundations of software technology and theoretical computer science*, volume 3821 of *Lecture Notes in Comput. Sci.*, pages 189–200. Springer, Berlin, 2005.
30. Ugo Dal Lago, Andrea Masini, and Margherita Zorzi. On a measurement-free quantum lambda calculus with classical control. *Mathematical Structures in Computer Science*, 19:297–335, 2009.
31. M. L. Dalla Chiara. Quantum logic and physical modalities. *J. Philos. Logic*, 6(4):391–404, 1977. Special issue: Symposium on Quantum Logic (Bad Homburg, 1976).
32. M. L. Dalla Chiara. Quantum logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic: Volume III: Alternatives to Classical Logic*, pages 427–469. Reidel, 1986.
33. Vincent Danos, Elham Kashefi, and Prakash Panangaden. The measurement calculus. *J. ACM*, 54(2):Art. 8, 45 pp. (electronic), 2007.
34. Vincent Danos and Laurent Regnier. The structure of multiplicatives. *Arch. Math. Logic*, 28(3):181–203, 1989.
35. David Deutsch. Quantum theory, the Church-Turing principle and the universal quantum computer. *Proceedings of the Royal Society of London Ser. A*, A400:97–117, 1985.
36. David Deutsch, Adriano Barenco, and Artur Ekert. Universality in quantum computation. *Proc. Roy. Soc. London Ser. A*, 449(1937):669–677, 1995.
37. Alessandra Di Pierro, Chris Hankin, and Herbert Wiklicky. Probabilistic λ-calculus and quantitative program analysis. *J. Logic Comput.*, 15(2):159–179, 2005.

38. P. A. M. Dirac. *The Principles of Quantum Mechanics*. Oxford, at the Clarendon Press, 1947. 3d ed.
39. Jacques Dixmier. *Les algèbres d'opérateurs dans l'espace hilbertien (algèbres de von Neumann)*. Les Grands Classiques Gauthier-Villars. [Gauthier-Villars Great Classics]. Éditions Jacques Gabay, Paris, 1996. Reprint of the second (1969) edition.
40. Richard P. Feynman. Simulating physics with computers. *Internat. J. Theoret. Phys.*, 21(6-7):467–488, 1981/82. Physics of computation, Part II (Dedham, Mass., 1981).
41. Richard P. Feynman. Quantum mechanical computers. *Found. Phys.*, 16(6):507–531, 1986.
42. Dov M. Gabbay. *Labelled Deductive Systems*, volume 1. Clarendon Press, 1996.
43. Marco Gaboardi and Simona Ronchi Della Rocca. From light logics to type assignements: a case study. *Logic Journal of the IGPL*, 2009.
44. S. J. Gay and R. Nagarajan. Communicating quantum processes. In *Proceedings of the 32nd ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages*. ACM Press, 2005.
45. S. J. Gay and R. Nagarajan. Typechecking communicating quantum processes. *Mathematical Structures in Computer Science*, 2007.
46. Simon J. Gay. Quantum programming languages: Survey and bibliography. *Mathematical Structures in Computer Science*, 16(4), 2006.
47. Jean-Yves Girard. Linear logic. *Theoret. Comput. Sci.*, 50(1):101, 1987.
48. Jean-Yves Girard. *Proof theory and logical complexity*, volume 1 of *Studies in Proof Theory. Monographs*. Bibliopolis, Naples, 1987.
49. Jean-Yves Girard. Linear logic: its syntax and semantics. In *Advances in linear logic (Ithaca, NY, 1993)*, volume 222 of *London Math. Soc. Lecture Note Ser.*, pages 1–42. Cambridge Univ. Press, Cambridge, 1995.
50. Jean-Yves Girard. Light linear logic. *Inform. and Comput.*, 143(2):175–204, 1998.
51. Jean-Yves Girard. Between logic and quantic: a tract. In *Linear logic in computer science*, volume 316 of *London Math. Soc. Lecture Note Ser.*, pages 346–381. Cambridge Univ. Press, Cambridge, 2004.
52. Jean-Yves Girard, Andre Scedrov, and Philip J. Scott. Bounded linear logic: a modular approach to polynomial-time computability. *Theor. Comput. Sci.*, 97(1):1–66, 1992.
53. Jean-Yves Girard, Paul Taylor, and Yves Lafont. *Proofs and types*, volume 7 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, 1989.
54. Lov K. Grover. Quantum search on structured problems. In *Quantum computing and quantum communications (Palm Springs, CA, 1998)*, volume 1509 of *Lecture Notes in Comput. Sci.*, pages 126–139. Springer, Berlin, 1999.
55. Mika Hirvensalo. *Quantum computing*. Natural Computing Series. Springer-Verlag, Berlin, second edition, 2004.
56. W. A. Howard. The formulae-as-types notion of construction. In *To H. B. Curry: essays on combinatory logic, lambda calculus and formalism*, pages 480–490. Academic Press, London, 1980.
57. Chris J. Isham. *Lectures on quantum theory*. Imperial College Press, London, 1995. Mathematical and structural foundations.
58. Phillip Kaye, Raymond Laflamme, and Michele Mosca. *An introduction to quantum computing*. Oxford University Press, Oxford, 2007.
59. A. Yu. Kitaev, A. H. Shen, and M. N. Vyalyi. *Classical and quantum computation*, volume 47 of *Graduate Studies in Mathematics*. American Mathematical Society, Providence, RI, 2002. Translated from the 1999 Russian original by Lester J. Senechal.
60. S. C. Kleene. λ-definability and recursiveness. *Duke Math. J.*, 2(2):340–353, 1936.
61. E. Knill. Conventions for quantum pseudocode. Technical Report LAUR-96-2724, Los Alamos National Laboratory, 1996.
62. Vladimir E. Korepin and Lov K. Grover. Simple algorithm for partial quantum search. *Quantum Inf. Process.*, 5(1):5–10, 2006.

63. Yves Lafont. Soft linear logic and polynomial time. *Theoret. Comput. Sci.*, 318(1-2):163–180, 2004.
64. Ugo Dal Lago, Andrea Masini, and Margherita Zorzi. Quantum implicit computational complexity. *submitted*, 2008.
65. Saunders Mac Lane and Garrett Birkhoff. *Algebra*. The Macmillan Co., New York, 1967.
66. Per Martin-Löf. *Intuitionistic type theory*, volume 1 of *Studies in Proof Theory. Lecture Notes*. Bibliopolis, Naples, 1984. Notes by Giovanni Sambin.
67. Andrea Masini, Luca Viganò, and Margherita Zorzi. A Qualitative Modal Representation of Quantum Register Transformations. In G. Dueck, editor, *Proceedings of the 38th IEEE International Symposium on Multiple-Valued Logic (ISMVL 2008)*, pages 131–137. IEEE Computer Society Press, 2008.
68. P. Maymin. Extending the lambda calculus to express randomized and quantu- mized algorithms. Technical Report arXiv:quant-ph/9612052, arXiv, 1996.
69. P. Maymin. The lambda-q calculus can efficiently simulate quantum computers. Technical Report arXiv:quant-ph/9702057, arXiv, 1997.
70. Peter Mittelstaedt. The modal logic of quantum logic. *J. Philos. Logic*, 8(4):479–504, 1979.
71. Takayuki Miyadera and Masanori Ohya. On halting process of quantum turing machine. *Open Systems & Information Dynamics*, 12(3):261–264, 2005.
72. Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. Cambridge University Press, Cambridge, 2000.
73. Harumichi Nishimura and Masanao Ozawa. Computational complexity of uniform quantum circuit families and quantum turing machines. *Theor. Comput. Sci.*, 276(1-2):147–181, 2002.
74. Harumichi Nishimura and Masanao Ozawa. Perfect computational equivalence between quantum turing machines and finitely generated uniform quantum circuit families. Technical report, arXiv:quant-ph/0511117v2, 2008.
75. B. Ömer. *Structured Quantum Programming*. PhD thesis, Technical University of Vienna, 2003.
76. Christos H. Papadimitriou. *Computational complexity*. Addison-Wesley Publishing Company, Reading, MA, 1994.
77. S. Perdrix. Quantum patterns and types for entanglement and separability. In P. Selinger, editor, *Proceedings of the 3rd International Workshop on Quantum Programming Languages*, Electronic Notes in Theoretical Computer Science. Elsevier, 2005.
78. Vaughan Pratt. Linear logic for generalized quantum mechanics. In *Proceedings of the IEEE Workshop on Physics and Computation*, 1992.
79. Dag Prawitz. *Natural deduction. A proof-theoretical study*. Acta Universitatis Stockholmiensis. Stockholm Studies in Philosophy, No. 3. Almqvist & Wiksell, Stockholm, 1965.
80. Dag Prawitz. Ideas and results in proof theory. In *Proceedings of the Second Scandinavian Logic Symposium (Univ. Oslo, Oslo, 1970)*, pages 235–307. Studies in Logic and the Foundations of Mathematics, Vol. 63, Amsterdam, 1971. North-Holland.
81. Steven Roman. *Advanced linear algebra*, volume 135 of *Graduate Texts in Mathematics*. Springer, New York, third edition, 2008.
82. J. W. Sanders and P. Zuliani. Quantum programming. In *Mathematics of Program Construction*, volume 1837 of *Lecture Notes in Computer Science*. Springer, 2000.
83. Helmut Schwichtenberg and Stephen J. Bellantoni. Feasible computation with higher types. In *Proof and system-reliability (Marktoberdorf, 2001)*, volume 62 of *NATO Sci. Ser. II Math. Phys. Chem.*, pages 399–415. Kluwer Acad. Publ., Dordrecht, 2002.
84. Peter Selinger. A brief survey of quantum programming languages. In *Proceedings of the 7th International Symposium on Functional and Logic Programming*, volume 2998 of *Lecture Notes in Computer Science*, pages 1–6. Springer, 2004.
85. Peter Selinger. Towards a quantum programming language. *Mathematical Structures in Computer Science*, 14(4):527–586, 2004.

86. Peter Selinger. Towards a semantics for higher-order quantum computation. In *Proceeding of the 2nd International Workshop on Quantum Programming Languages*, number 33 in TUCS General Publication, pages 127–143. Turku Centre for Comp. Sci. General Publication, 2004.
87. Peter Selinger and Benoit Valiron. A lambda calculus for quantum computation with classical control. *Math. Structures Comput. Sci.*, 16(3):527–552, 2006.
88. Peter W. Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *35th Annual Symposium on Foundations of Computer Science (Santa Fe, NM, 1994)*, pages 124–134. IEEE Comput. Soc. Press, Los Alamitos, CA, 1994.
89. Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Rev.*, 41(2):303–332 (electronic), 1999.
90. Alex Simpson. *The proof theory and semantics of intuitionistic modal logic*. PhD thesis, University of Edinburgh, UK, 1993.
91. Alex Simpson. Reduction in a linear lambda-calculus with applications to operational semantics. In *Term rewriting and applications*, volume 3467 of *Lecture Notes in Comput. Sci.*, pages 219–234. Springer, Berlin, 2005.
92. Kazushige Terui. Light affine lambda calculus and polynomial time strong normalization. *Arch. Math. Logic*, 46(3-4):253–280, 2007.
93. T. Toffoli. Reversible computing. In W. de Bakker and J. van Leeuwen, editors, *Automata, Languages and Programming*, page 632. Springer, New York, 1980. Technical Memo MIT/LCS/TM-151, MIT Lab. for Comput. Sci. (unpublished).
94. A. S. Troelstra. Intuitionistic formal systems. In *Metamathematical investigation of intuitionistic arithmetic and analysis*, pages 1–96. Lecture Notes in Mathematics, Vol. 344. Springer, Berlin, 1973.
95. Anne Sjerp Troelstra and Helmut Schwichtenberg. *Basic proof theory*. Cambridge University Press, 1996.
96. Peter van Emde Boas. Machine models and simulation. In *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity (A)*, pages 1–66. MIT Press, 1990.
97. André van Tonder. A lambda calculus for quantum computation. *SIAM J. Comput.*, 33(5):1109–1135 (electronic), 2004.
98. L. Viganò. *Labelled Non-Classical Logics*. Kluwer Academic Publishers, 2000.
99. John von Neumann. *Mathematical foundations of quantum mechanics*. Princeton Landmarks in Mathematics. Princeton University Press, Princeton, NJ, 1996. Translated from the German and with a preface by Robert T. Beyer, Twelfth printing, Princeton Paperbacks.
100. Philip Wadler. A syntax for linear logic. In *Mathematical foundations of programming semantics (New Orleans, LA, 1993)*, volume 802 of *Lecture Notes in Comput. Sci.*, pages 513–529. Springer, Berlin, 1994.
101. Christopher Wadsworth. Some unusual λ-calculus numeral systems. In J.P. Seldin and J.R. Hindley, editors, *To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*. Academic Press, 1980.
102. Martin Wehr. Quantum computing: a new paradigm and its type theory. Lecture given at the Quantum Computing Seminar, Lehrstuhl Prof. Beth, Universität Karlsruhe, 1996.
103. W.K. Wootters and W.H. Zurek. A single quantum cannot be cloned. *Nature*, (299):802–803, 1982.
104. A. Yao. Quantum circuit complexity. In *Proceedings of the 34th Annual Symposium on Foundations of Computer Science*, pages 352–360, Los Alamitos, California, 1993. IEEE Press.
105. P. Zuliani. Quantum programming with mixed states. In P. Selinger, editor, *Proceedings of the 3rd International Workshop on Quantum Programming Languages*, Electronic Notes in Theoretical Computer Science. Elsevier, 2005.