Toward a Quantum Process Algebra

Philippe Jorrand*, Marie Lalire†
Leibniz Laboratory
46, avenue Félix Viallet 38000 Grenoble, France

February 1, 2008

Abstract

Quantum computations operate in the quantum world. For their results to be useful in any way, there is an intrinsic necessity of cooperation and communication controlled by the classical world. As a consequence, full formal descriptions of algorithms making use of quantum principles must take into account both quantum and classical computing components and assemble them so that they communicate and cooperate. This paper aims at defining a high level language allowing the description of classical and quantum programming, and their cooperation. Since process algebras provide a framework to model cooperating computations and have well defined semantics, they have been chosen as a basis for this language. Starting with a classical process algebra, this paper explains how to transform it for including quantum computation. The result is a quantum process algebra with its operational semantics, which can be used to fully describe quantum algorithms in their classical context.

1 Introduction

Quantum algorithms are often described by means of quantum gate networks. See [14] for an introduction to quantum computing, and [9] or [12] for a full account of quantum computing and quantum information. This has several drawbacks, for instance, gate networks do not allow descriptions of loops nor conditional execution of parts of networks. So as to overcome these difficulties, a few quantum programming languages have been developed, such as: QCL [13], an imperative language which aims at simulating quantum programs, qGCL [17] which allows the construction of proved correct quantum programs through a refinement method, and QPL [15], a functional language with a denotational semantics. There is also the work of Alessandra Di Pierro and Herbert Wiklicky

*Philippe.Jorrand@imag.fr
†Marie.Lalire@imag.fr

[7] who adapted constraint programming to quantum computation, with the purpose of defining a semantical framework for quantum programming. More recently, André Van Tonder has developed a quantum lambda calculus [16], based on a simplified linear lambda calculus.

Cooperation between quantum and classical computations is inherent in quantum algorithmics. For example, the quantum computation part is in general probabilistic: it produces a result which is checked by a classical part and, if this result is not correct, the quantum computation has to be repeated. Teleportation of a qubit state from Alice to Bob [3] is another good example of this cooperation. Indeed, Alice carries out a measurement, the classical result of which (two bits) is sent to Bob, and Bob uses this result to determine which quantum transformation he must apply. Moreover, initial preparation of quantum states and measurement of quantum results are two essential forms of interactions between the classical and quantum kinds of computations which the language must be able to express. Process algebras are a good candidate for such a language since they provide a framework for modeling cooperating computations. In addition, they have well defined semantics and permit the transformation of programs as well as the formal study and analysis of their properties.

Process algebras have already been used in the context of quantum programming in [11], where the authors have modeled a quantum cryptographic protocol and verified its correctness with a classical process algebra. Starting with a classical process algebra in section 2, this paper explains the essential features of quantum computation in section 3 and "quantumizes" the initial process algebra in section 4. Examples of short quantum programs are given in section 5.

2 A Classical Process Algebra

The classical process algebra chosen here is quite similar to CCS [10] and Lotos [4]. Its syntax and semantics are given in appendix A. In this process algebra, communication among processes is the only basic action. There is a distinction between value emission denoted $g\ !v$, where g is a communication gate and v a value, and value reception denoted $g\ ?x$, where g is a gate and x a variable which receives the value. To create a process from basic actions, the prefix operator "." is used: if α is an action and P, a process, $\alpha.P$ is a new process which performs α first, then behaves as P.

There are two predefined processes. The first one is *nil*, the process that cannot perform any transition, and the other one is *end*, which performs a "δ-transition" for signaling successful termination, and becomes *nil* ("δ-transitions" are necessary in the semantics of sequential composition of processes).

The operators of the process algebra are: sequential composition ($P\ ;Q$), parallel composition ($P \parallel Q$), conditional choice ($[]_i\ \ c_i \rightarrow P_i$) and restriction

$(P|L\ )$. As for sequential composition, process Q is executed if process P terminates successfully, that is to say if P performs a δ-transition. The process $[]_i\ \ c_i \rightarrow P_i$, where c_i is a condition and P_i a process, evolves as a process chosen nondeterministically among the processes P_j such that c_j is true. Restriction is useful for disallowing the use of some gates (the gates listed in L), thus forcing internal communication within process P. Communication can occur between two parallel processes whenever a value emission in one of them and a value reception in the other one use the same gate name. For instance, a communication can occur in the process $g\ !v\ .P \parallel g\ ?x\ .Q$ on gate g. After the communication has occurred, this process becomes $P \parallel Q[x \leftarrow v]$ where $Q[x \leftarrow v]$ is Q where all occurrences of x have been replaced by v.

3 Quantum Computing

3.1 Qubits and Registers

Whereas classical computing is based on bits taking values in $\{0, 1\}$, quantum computing is based on qubits (quantum bits). The states of qubits are normalized vectors in a two dimensional space $\mathbb{C}^2$, where $\mathbb{C}$ are the complex numbers.

In the following, we will use Dirac's notation $|.\rangle$ for vectors. Let $|0\rangle$ and $|1\rangle$ be two normalized vectors forming an orthonormal basis of $\mathbb{C}^2$:

$$|0\rangle = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad |1\rangle = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

The state of a qubit can be written $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ where $\alpha, \beta \in \mathbb{C}$ and $|\alpha|^2 + |\beta|^2 = 1$ so that $|\psi\rangle$ is normalized.

Like in classical computing where bits are organized into registers, there are also registers of qubits. The state $|\psi\rangle$ of a register made of two qubits in states $|\psi_1\rangle$ and $|\psi_2\rangle$ respectively, is the tensor product of the states of these qubits, that is to say $|\psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$. The state of a two qubit register is thus a normalized vector in a 4 dimensional space $\mathbb{C}^4$ with basis $\{|i\rangle \otimes |j\rangle\},\ i, j \in \{0, 1\}$, usually denoted $\{|00\rangle, |01\rangle, |10\rangle, |11\rangle\}$: $|\psi\rangle = \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$. This can be generalized to registers of n qubits: their states are normalized vectors in a 2^n dimensional space $\mathbb{C}^{2^n}$.

It is important to note that the state of a n qubit register cannot in general be written as a tensor product of the states of the qubits which compose the register. Such states are "entangled" states. For instance, the famous state $|EPR\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$[1]is an entangled state because for all $\alpha, \beta, \gamma, \delta$ in $\mathbb{C}$, $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \neq (\alpha|0\rangle + \beta|1\rangle) \otimes (\gamma|0\rangle + \delta|1\rangle)$.

[1]EPR comes from Einstein, Podolsky and Rosen [8], who questioned the completeness of quantum theory because of the existence of such states. Later, theoretical results [2] followed by experiments [1] proved them wrong.

It must be noted that entangled states are an essential difference between the classical and the quantum worlds: in the quantum world, it is not true in general that the state of a system composed of n sub-systems can be reduced to an n-tuple of the states of its components. This will have to be taken into account in the semantics of the process algebra presented here.

3.2 Deterministic Evolution

According to the postulates of quantum mechanics, the evolution of a closed quantum system — *i.e.* which is not observed — can be described by a unitary transformation. A unitary transformation on n qubits can be represented by a complex and unitary $2^n \times 2^n$ matrix U. The unitarity condition maintains the normalization and can be written $UU^\dagger = U^\dagger U = I$ where $U^\dagger$ is the adjoint (*i.e.* conjugate transpose) of U. This condition implies that the evolution of a closed quantum system is reversible.

3.3 Probabilistic Measurement

Measurement corresponds to the observation of a quantum system by a classical system. Contrary to the classical world, where reading a bit does not change its state, the observation of qubits is destructive: in general, measuring a qubit modifies its state irreversibly.

Moreover, measurement is probabilistic: measuring two qubits initially prepared in identical states, will not necessarily produce identical results. Let $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ be the state of a qubit. Measurement of a qubit is performed relatively to a basis of $\mathbb{C}^2$. Measuring this qubit in the standard basis $\{|0\rangle, |1\rangle\}$, yields $|0\rangle$ with probability $|\alpha|^2$, and $|1\rangle$ with probability $|\beta|^2$. For this reason, unlike unitary transformations, measurement is not reversible.

Only so-called "projective" measurements will be considered here. A projective measurement is described by an hermitian matrix M (*i.e.* $M = M^\dagger$), called an observable. The set of observables is denoted $\mathcal{O}$. Quantum registers of n qubits measured with observable M (a $2^n \times 2^n$ matrix) have their state projected onto one of the eigenspaces of M and renormalized. Indeed, since M is hermitian, it has a spectral decomposition, $M = \sum_m mP_m$ where P_m is the projector onto the eigenspace of M corresponding to the eigenvalue m (which is real since M is hermitian). So, if a quantum register in state $|\psi\rangle$ is measured with observable M, its state is transformed to:

$$\frac{P_m|\psi\rangle}{\sqrt{p_m}}$$

with probability p_m. $P_m|\psi\rangle$ is the projection of $|\psi\rangle$ onto the eigenspace corresponding to m, and $\sqrt{p_m}$ is the renormalization factor. The probability p_m is $\langle\psi|P_m|\psi\rangle$, the scalar product of $|\psi\rangle$ and $P_m|\psi\rangle$. The result of the measurement, as viewed by the classical observer, is the value m.

For example, the observable M_{std} for measuring one qubit in the standard basis, is :

$$M_{std} = 0P_0 + 1P_1 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

where P_0 and P_1 are the outer products $|0\rangle\langle 0| = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ and $|1\rangle\langle 1| = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$ respectively (with Dirac's notation, $\langle 0| = \begin{pmatrix} 1 & 0 \end{pmatrix}$, $\langle 1| = \begin{pmatrix} 0 & 1 \end{pmatrix}$).

3.4 Two Important Quantum Features

3.4.1 Entangled States.

Entangled states show amazing properties, especially when measurements are performed on parts of them. For instance, with a two qubit register in state $|EPR\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, measuring one of the two qubits in the standard basis yields:

- either 0, with probability 1/2, and the state becomes $|00\rangle$,
- or 1, with probability 1/2, and the state becomes $|11\rangle$.

After that, if the other qubit is also measured in the standard basis, the same result as for the first measurement is obtained with probability 1: the results of the two measurements are correlated and this is independent of the spatial arrangement of the qubits and the distance separating them.

3.4.2 No Cloning Theorem.

An other important difference between quantum computing and classical computing is that in the quantum world, it is impossible to copy the state of quantum systems. This is known as the *no cloning theorem*: an unknown quantum state cannot be cloned. In the context of languages and programming, this means that it is impossible to copy the value of a quantum variable into another quantum variable.

4 "Quantumized" Processes

4.1 Quantum Variables

There are two types of variables in the "quantumized" process algebra, one classical: *Nat*, for variables taking integer values, and one quantum: *Qubit* for variables standing for qubits.

In classical process algebras, variables are instantiated when communications between processes occur and cannot be modified after their instantiation. As a consequence, it is not necessary to store their values. In fact, when a variable is instantiated, all its occurrences are replaced by the value received (see the semantics of communication in parallel composition, as given in appendix A).

Here, quantum variables stand for physical qubits. Applying a unitary transformation to a variable which represents a qubit modifies the state of that qubit. This means that values of variables are modified. For that reason, it is necessary to keep track of both variable names and variable states.

Since variables are no longer just names standing for communicated values, they have to be declared. The syntax of declarations is:

$$\kappa[x_1, \ldots, x_n]\theta[y_1, \ldots, y_m] : P \bullet$$

where $x_1, \ldots, x_n$ is a list of classical variables, $y_1, \ldots, y_n$ is a list of quantum variables, and P is a process which can make use of these classical and quantum variables. Variables have their scope limited on the left by their declaration and on the right by the postfixed operator $\bullet$. To simplify the rest of this paper, the names of variables will always be considered distinct.

As already said, it is necessary to store the states and the names of variables during the execution of a process. Consequently, in the inference rules which describe the semantics of processes, the states of processes can no longer be process terms only, as it was the case for the classical process algebra, they have to be process terms P together with contexts C, of the form P/C. The main purpose of a context is to maintain the quantum state, stored as $q = |\psi\rangle$ where q is a sequence of quantum variable names and $|\psi\rangle$ their quantum state. Moreover, in order to treat classical variables in a similar way, modifications of classical variables are also allowed. So, for the same reason as in the case of quantum variables, classical values are stored in the context. Storing and retrieving classical values is represented by functions $f : \mathit{names} \rightarrow \mathit{values}$. The context must also keep track of the embedding of variable scopes. To keep track of parallel composition, this is done via a "cactus stack" structure of sets of variables, called the environment stack (s), which stores variable scopes and types. The set of all the variables in s is denoted $\mathrm{Var}(s)$.

In summary, the context has three components $< s, q = |\psi\rangle, f >$, where:

- s is the environment stack;
- q is a sequence of quantum variable names;
- $|\psi\rangle$ is the quantum state of the variables in q;
- f is the function which associates values to classical variables.

The rules for declaration and liberation of variables are the following:

Declaration:

$$\frac{}{\kappa[x_1,\ldots,x_n]\ \theta[y_1,\ldots,y_m] : P\ \bullet/C \quad \longrightarrow \quad P\ \bullet/C'}$$

with $C =< s, q = |\psi\rangle, f >$, $C' =< s', q = |\psi\rangle, f >$ and
$s' = \{(x_1, \text{Nat}), \ldots, (x_n, \text{Nat}), (y_1, \text{Qubit}), \ldots, (y_m, \text{Qubit})\}.s$

This rule adds the new variable names and types to the stack s. Because the variables do not have values yet, the quantum state and the classical function do not have to be modified at this point.

Evolution of a process within the scope of declared variables:

$$\frac{P/C \quad \dashrightarrow \quad P'/C'}{P\ \bullet/C \quad \dashrightarrow \quad P'\ \bullet/C'}$$

where $\dashrightarrow$ stands for any of the transitions: $\xrightarrow{\alpha}$ with α an action, $\xrightarrow{\tau}$ with τ the "silent" action, and the declaration transition $\longrightarrow$.

In short: if the process P can perform a transition, then the process $P\ \bullet$ can perform the same transition, provided that the action of the transition is not δ.

Termination of a process with exit from a scope and liberation of the variables:

$$\frac{P/C \quad \xrightarrow{\delta} \quad P'/ < e.s, q = |\psi\rangle, f >}{P\ \bullet/C \quad \xrightarrow{\delta} \quad nil\ / < s, q[e \leftarrow *] = |\psi\rangle, f|_{\text{Var}(s)} >}$$

If the action is δ, this means that P has successfully terminated, so the context must be cleaned up by eliminating the variables having their scope limited to that process.

Cleaning up the context means eliminating the head of the stack and restricting the function f to the variables remaining in the stack ($f|_E$ means f restricted to E). As regards to the quantum part of the context, because of possible entanglement among local variables and other more global ones, qubits corresponding to these local variables cannot be removed. Only their variable names are erased and replaced by a "$*$" in the sequence q ($q[e \leftarrow *]$ is q in which all the names listed in e have been replaced by $*$). The quantum state is not modified.

4.2 Basic Actions

The classical basic actions are classical to classical communications. Classical to quantum communications are introduced for initializing qubits. Quantum to classical communications are part of measurement and are dealt with in the next paragraph.

The semantics of communications is based upon the following rules:

$$\frac{}{g\ !v\ .P/C \quad \xrightarrow{g\ !v} \quad P/C} \quad v \in I\!N$$

$$\frac{}{g\ ?x\ .P/C \quad \xrightarrow{g\ ?x} \quad P/C}$$

with $C =< s, q = |\psi\rangle, f >$, $x \in \mathrm{Var}(s)$, and $x \notin q$.

The first rule deals with classical value sending, and the second one, with value reception. It should be noted that in the second rule, the variable x can be classical or quantum but, if it is quantum, it must not have already been initialized. In the semantics of parallel composition, the combination of these rules defines communication. If x is a qubit, the communication initializes it in the basis state $|v\rangle$, where v is the classical value sent (in this case, v must be 0 or 1).

The second kind of basic actions is unitary transformations which perform the unitary evolution of qubit states. Given a set $\mathcal{U}$ of predefined unitary transformations, the action corresponding to the application of $U \in \mathcal{U}$ to a list of quantum variables is denoted by $U[x_1, \ldots, x_n]$.

The inference rule for unitary transformations is:

$$\frac{}{U[x_1, \ldots, x_n].P/C \quad \xrightarrow{\tau} \quad P/C'}$$

where

- $C =< s, q = |\psi\rangle, f >$, $C' =< s, q = |\psi'\rangle, f >$
- $U \in \mathcal{U}$, $x_1, \ldots, x_n \in \mathrm{Var}(s)$, and $x_1, \ldots, x_n \in q$
- $|\psi'\rangle = \Pi^t.(U \otimes I^{\otimes k}).\Pi|\psi\rangle$
- Π is the permutation matrix which places the x_i's at the head of q and Π^t is the transpose of Π
- $k = \mathrm{size}(q) - n$
- $I^{\otimes k} = \underbrace{I \otimes \cdots \otimes I}_{k}$, where I is the identity matrix on $\mathbb{C}^2$

The condition $x_1, \ldots, x_n \in q$ prevents from applying a unitary transformation to qubits which have not been initialized. The third point deals with the evolution from a quantum state initially equal to $|\psi\rangle$. Since the unitary transformation U may be applied to qubits which are anywhere within the list q, a permutation Π must be applied first. This permutation moves the x_i's so that they are placed at the head of q in the order specified by $[x_1, \ldots, x_n]$. Then U can be applied to the first n elements and I to the remainder. Finally, the last operation is the inverse of the permutation Π ($\Pi^{-1} = \Pi^t$) so that at the end, the elements in q and $|\psi\rangle$ are put back in the same order.

4.3 Measurement and Probabilistic Processes

A last but essential basic action has to be introduced into the process algebra: quantum measurement. Let $M \in \mathcal{O}$ be an observable, $x_1, \ldots, x_n$ a list of quantum variables and g a gate. Then, the syntax for measurement is the following:

- $M[x_1, \ldots, x_n]$ is a measurement of the n qubits of the list with respect to observable M, but the classical result is neither stored nor transmitted.
- $g\ !M[x_1, \ldots, x_n]$ is a measurement of the n qubits of the list with respect to observable M, followed by sending the classical result through gate g.

As said in paragraph 3.3, measurement is probabilistic: more precisely, the classical result and the quantum state after measurement are probabilistic. This requires the introduction of a probabilistic composition operator for contexts. This operator is denoted $\boxplus_p$: the state $P/C_1 \boxplus_p C_2$ is P/C_1 with probability p and P/C_2 with probability $1-p$.

This implies that, in general, the context is either of the form $< s, q = |\psi\rangle, f >$, or of the form $\boxplus_{p_i} < s_i, q_i = |\psi_i\rangle, f_i >$ where the p_i's are probabilities.

As explained in [5, 6], if a process contains both a probabilistic and a nondeterministic choice, the probabilistic choice must always be solved first. In the process algebra presented here, nondeterminism appears with parallel composition and conditional choice. So as to guarantee that probabilistic choice is always solved first, the notion of probabilistic stability for contexts is introduced: a context C is probabilistically stable, which is denoted $C \downarrow$, if it is of the form $< s, q = |\psi\rangle, f >$. If the context of a process state is not stable, this state must perform a probabilistic transition.

The semantic rule for measurement without communication is:

$$\frac{}{M[x_1, \ldots, x_n].P/C \quad \xrightarrow{\tau} \quad P/\boxplus_{p_i} < s, q = |\psi_i\rangle, f >}$$

with

- $C =< s, q = |\psi\rangle, f >$ (which implies $C \downarrow$)
- $x_1, \ldots, x_n \in \mathrm{Var}(s)$ and $x_1, \ldots, x_n \in q$
- $M \in \mathcal{O}$ with $\sum_i \lambda_i P_i$ as spectral decomposition
- $p_i = \langle\psi|\Pi^t(P_i \otimes I^{\otimes k})\Pi|\psi\rangle$
- $|\psi_i\rangle = \dfrac{\Pi^t(P_i \otimes I^{\otimes k})\Pi|\psi\rangle}{\sqrt{p_i}}$
- Π is the permutation matrix which places the x_i's at the head of q and Π^t is the transpose of Π

- $k = \text{size}(q) - n$

As in the case of unitary transformations, a permutation Π rearranges the qubits so that projectors apply only to measured qubits. The computations of $|\psi_i\rangle$ and p_i stem from the projective measurement postulate of quantum mechanics as summarized in paragraph 3.3.

When the value coming out of the measurement is sent out, the rule is:

$$\frac{}{g\ !M[x_1,\ldots,x_n]\ .P/C \quad \xrightarrow{\tau} \quad (g\ !y\ .end\ \bullet)\ ;P/\boxplus_{p_i} C_i}$$

where

- y is a new variable (implicitly declared as $\kappa[y]$, see below)
- $C =< s, q = |\psi\rangle, f >$ (which implies $C\downarrow$)
- $C_i =< \{(y, \text{Nat})\}.s, q = |\psi_i\rangle, f \lhd \{y \mapsto \lambda_i\} >$
- and the conditions are the same as in the rule without communication.

The only remaining point is the evolution of processes with probabilistic contexts. It is necessary to introduce probabilistic transitions for describing this evolution: $S_1 \quad \longrightarrow_p \quad S_2$ means that state S_1 becomes S_2 with probability p. This is used in the following rule:

$$\frac{}{P/\boxplus_{p_i} C_i \quad \longrightarrow_{p_i} \quad P/C_i} \text{ where } \sum_j p_j = 1$$

The syntax and the main inference rules of this quantum process algebra are presented in appendix B.

5 Examples

A few unitary transformations are often used in quantum algorithms:

- *Hadamard* is a transformation on one qubit denoted H. Its action is the creation of uniform superpositions:
  $H : |0\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$ and $|1\rangle \mapsto \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$
- *Controlled Not* is a transformation on two qubits denoted $CNot$. If the first qubit is in state $|1\rangle$, it flips the state of the second qubit:
  $CNot : |00\rangle \mapsto |00\rangle, |01\rangle \mapsto |01\rangle, |10\rangle \mapsto |11\rangle, |11\rangle \mapsto |10\rangle$

- *Pauli matrices* are four transformations on one qubit:

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},$$
$$Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

5.1 Construction of an EPR pair

$$\begin{array}{lll} \text{BuildEPR} & \stackrel{\text{def}}{=} & \theta[x,y]: \\ & & ((g_1\ ?x\ .g_2\ ?y\ .H[x].CNot[x,y].end) \\ & & \|\ (g_1\ !0\ .g_2\ !0\ .end)) \\ & & |\{g_1,g_2\}\ \bullet \end{array}$$

This process puts the pair of qubits x, y in the state $|EPR\rangle$ (see paragraph 3.4.1). To check that the order of measurement of the two qubits does not matter, it is possible, using the inference rules, to analyze the behaviour of the following two processes: in both of them, the first measurement produces 0 (1) with probability 0.5 and the second measurement produces 0 (1) with probability 1.

$$\begin{array}{lll} \text{CheckEPR}_1 & \stackrel{\text{def}}{=} & \theta[a,b]: \quad \text{BuildEPR}[a,b]\ ; \\ & & \qquad\qquad M_{std}[a].M_{std}[b].end\ \bullet \end{array}$$

$$\begin{array}{lll} \text{CheckEPR}_2 & \stackrel{\text{def}}{=} & \theta[a,b]: \quad \text{BuildEPR}[a,b]\ ; \\ & & \qquad\qquad M_{std}[b].M_{std}[a].end\ \bullet \end{array}$$

5.2 Teleportation

Once upon a time, there were two friends, Alice and Bob who had to separate and live away from each other. Before leaving, each one took a qubit of the same EPR pair. Then Bob went very far away, to a place that Alice did not know. Later on, someone gave Alice a mysterious qubit in a state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, with a mission to forward this state to Bob. Alice could neither meet Bob and give him the qubit, nor clone it and broadcast copies everywhere, nor measure it to know α and β. Nevertheless, Alice succeeded thanks to the EPR pair and the teleportation protocol [3]:

$$\text{Alice} \stackrel{\text{def}}{=} \theta[x,y] : CNot[x,y].H[x].meas\ !M[x,y]\ .end \bullet$$

$$\begin{aligned}\text{Bob} \stackrel{\text{def}}{=}\ & \theta[z] : (\kappa[k] : meas\ ?k\ .[\!][k=0 \rightarrow I[z].end \\ & \qquad [\!][k=1 \rightarrow X[z].end \\ & \qquad [\!][k=2 \rightarrow Z[z].end \\ & \qquad [\!][k=3 \rightarrow Y[z].end \bullet) \bullet\end{aligned}$$

$$\begin{aligned}\text{Teleport} \stackrel{\text{def}}{=}\ & \theta[\psi] : (\theta[a,b] : \text{BuildEPR}[a,b]\ ; \\ & \qquad (\text{Alice}[\psi,a] \parallel \text{Bob}[b])|\{meas\}\ \ \bullet) \bullet\end{aligned}$$

M is the observable corresponding to measuring two qubits in the standard basis of $\mathbb{C}^4$:

$$M = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 3 \end{pmatrix}$$

The inference rules can be used to show that this protocol results in Bob's z qubit having the state initially possessed by the x qubit of Alice, with only two classical bits sent from Alice to Bob.

6 Conclusion

This paper has presented a process algebra for quantum programming. One of its advantages is that it can describe classical and quantum programming, and their cooperation. Without this cooperation, the implementation of the teleportation protocol, for instance, is not possible. Another feature of this language is that measurement and initialization of quantum registers appear through communications between quantum and classical parts of the language, which happens to be a faithful model of physical reality.

Moreover, a thorough semantics has been defined, thus allowing the study and analysis of programs. One peculiarity of this semantics is the introduction of probabilistic processes, due to quantum measurement. Probabilistic processes perform probabilistic transitions. As a consequence, the execution tree obtained from a process presents action and probabilistic branches.

Several extensions are possible. Firstly, quantum to quantum communications could be added to allow the modeling of cryptographic protocols. Another track that could be followed is the use of density matrices, which are a more general description of quantum states than vectors in $\mathbb{C}^{2^n}$. Density matrices notably permit to describe states of parts of entangled registers. They would give a more abstract semantics to this process algebra and open the way to a semantic analysis similar to abstract interpretation.

7 Acknowledgment

The authors thank Frédéric Prost for having suggested nice improvements to a previous version of this work.

References

[1] A. Aspect, J. Dalibard, and G. Roger. Experimental tests of Bell's inequalities using time-varying analysers. *Physical Review Letters*, 49:1804–1807, 1982.

[2] J. S. Bell. On the Einstein-Podolsky-Rosen paradox. *Physics*, 1:195–200, 1964.

[3] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. Wootters. Teleporting an unknown quantum state via dual classical and EPR channels. *Physical Review Letters*, 70:1895–1899, 1993.

[4] T. Bolognesi and E. Brinksma. Introduction to the ISO specification language LOTOS. *Computer Networks and ISDN Systems*, 14(1):25–59, 1987.

[5] D. Cazorla, F. Cuartero, V. Valero, and F. L. Pelayo. A process algebra for probabilistic and nondeterministic processes. *Information Processing Letters*, 80(1):15–23, 2001.

[6] D. Cazorla, F. Cuartero, V. Valero, F. L. Pelayo, and J. Pardo. Algebraic theory of probabilistic and nondeterministic processes. *The Journal of Logic and Algebraic Programming*, 55(1–2):57–103, 2003.

[7] A. Di Pierro and H. Wiklicky. Quantum constraint programming. In L. M. Pereira and P. Quaresma, editors, *Proceedings of APPIA-GULP-PRODE'01 - Joint Conference on Declarative Programming*, pages 113–130, Evora, Portugal, 2001.

[8] A. Einstein, B. Podolsky, and N. Rosen. Can quantum-mechanical description of physical reality be considered complete? *Physical Review*, 47:777–780, 1935.

[9] J. Gruska. *Quantum Computing*. McGraw-Hill, June 1999.

[10] R. Milner. *Communication and Concurrency*. Prentice-Hall, London, 1989.

[11] R. Nagarajan and S. Gay. Formal verification of quantum protocols. *Los Alamos arXive e-print quant-ph/0203086*, 2002.

[12] M. A. Nielsen and I. L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.

[13] B. Omer. Quantum programming in QCL. Master's thesis, Institute Information System, Technical University of Vienna, 2000.

[14] E. Rieffel and W. Polak. An introduction to quantum computing for non-physicists. *ACM Computing Surveys*, 32(3):330–335, 2000.

[15] P. Selinger. Towards a quantum programming language. To appear in Mathematical Structures in Computer Science, 2003.

[16] A. Van Tonder. A lambda calculus for quantum computation. *Los Alamos arXive e-print quant-ph/0307150*, 2003.

[17] P. Zuliani. *Quantum Programming*. PhD thesis, St Cross College, University of Oxford, 2001.

A A classical process algebra

A.1 Syntax of process terms

$$
\begin{array}{lcl}
\mathit{elem\_cond} & ::= & \mathit{variable} \mid \mathit{value} \\
\mathit{comp} & ::= & = \mid \neq \mid \leq \mid \geq \mid < \mid > \\
\mathit{cond} & ::= & \mathit{elem\_cond}\ \mathit{comp}\ \mathit{elem\_cond} \\
 & & \\
\mathit{communication} & ::= & \mathit{gate}\ \mathbf{!}\ \mathit{value} \mid \mathit{gate}\ \mathbf{?}\ \mathit{variable} \\
 & & \\
\mathit{process} & ::= & \boldsymbol{nil} \\
 & \mid & \boldsymbol{end} \\
 & \mid & \mathit{process\_name} \\
 & \mid & \mathit{communication}\ .\ \mathit{process} \\
 & \mid & \mathit{process}\ ;\ \mathit{process} \\
 & \mid & \mathit{process} \parallel \mathit{process} \\
 & \mid & \{[\!]\ \mathit{cond} \rightarrow \mathit{process}\ \}^{+} \\
 & \mid & \mathit{process} \mid\{\ \mathit{gate}\ \{\ ,\mathit{gate}\ \}^{*}\}\ \ \mathit{(synchronization)} \\
 & & \\
\mathit{proc\_decl} & ::= & \mathit{process\_name} \stackrel{\boldsymbol{def}}{=} \mathit{process}
\end{array}
$$

A.2 Semantics

The semantics is specified with inference rules which give the evolution of the states of processes. In the classical process algebra considered here, the state of a process is a process term. The inference rules are of the form:

$$\frac{\mathit{Premises}}{\mathit{Conclusion}} \quad \mathit{Condition}$$

which means that if *Premises* have been established and *Condition* holds then the *Conclusion* can be infered.

Premises and *Conclusion* are of the form $P \quad \dashrightarrow \quad P'$, which means that P can execute a transition and become P'. There are three kinds of transitions:

- action transition: $\xrightarrow{\alpha}$ where α is $g\ !v$ or $g\ ?x$;
- silent transition: $\xrightarrow{\tau}$, for internal transition;
- delta transition: $\xrightarrow{\delta}$, for successful termination.

For instance, the rule:

$$\frac{P \quad \xrightarrow{\alpha} \quad P'}{Q \quad \xrightarrow{\beta} \quad Q'} \quad K$$

can be read: if process P can perform action α then become process P', and if condition K holds, then process Q can perform action β and become Q'.

In the following, P, Q, P', Q', P_i and P_i' are processes, α and α_i are actions, g is a communication gate, v is a value, x is a variable, and c_j is a condition.

Successful termination

$$\frac{}{end \quad \xrightarrow{\delta} \quad nil}$$

Action Prefix

$$\frac{}{g\ !v\ .P \quad \xrightarrow{g\ !v} \quad P} \quad v \in I\!N$$

$$\frac{}{g\ ?x\ .P \quad \xrightarrow{g\ ?x} \quad P}$$

Sequential composition

$$\frac{P \quad \xrightarrow{\alpha} \quad P'}{P\ ;Q \quad \xrightarrow{\alpha} \quad P'\ ;Q} \quad \alpha \neq \delta$$

$$\frac{P \quad \xrightarrow{\delta} \quad P'}{P\ ;Q \quad \xrightarrow{\tau} \quad Q}$$

Parallel composition

$$\frac{P \quad \xrightarrow{\alpha} \quad P'}{P \parallel Q \quad \xrightarrow{\alpha} \quad P' \parallel Q} \quad \alpha \neq \delta$$

$$\frac{Q \quad \xrightarrow{\alpha} \quad Q'}{P \parallel Q \quad \xrightarrow{\alpha} \quad P \parallel Q'} \quad \alpha \neq \delta$$

$$\frac{P \xrightarrow{g\,!v} P' \quad Q \xrightarrow{g\,?x} Q'}{P \parallel Q \xrightarrow{\tau} P' \parallel Q'[x \leftarrow v]}$$

$$\frac{P \xrightarrow{g\,?x} P' \quad Q \xrightarrow{g\,!v} Q'}{P \parallel Q \xrightarrow{\tau} P'[x \leftarrow v] \parallel Q'}$$

$$\frac{P \xrightarrow{\delta} P' \quad Q \xrightarrow{\delta} Q'}{P \parallel Q \xrightarrow{\delta} nil}$$

Conditional choice

$$\frac{P_i \xrightarrow{\alpha_i} P_i'}{[\!]_j \; c_j \rightarrow P_j \xrightarrow{\alpha_i} P_i'} \; c_i$$

Restriction

$$\frac{P \xrightarrow{\alpha} P'}{P|L \xrightarrow{\alpha} P'|L} \qquad \begin{array}{c} \alpha = \tau \vee \alpha = \delta \\ \vee(\alpha = g[\ !v \text{ or } ?x\ ] \wedge g \notin L) \end{array}$$

B The quantum process algebra

B.1 Syntax

elem_cond	::=	*variable* | *value*
comp	::=	**=** | **≠** | **≤** | **≥** | **<** | **>**
cond	::=	*elem_cond comp elem_cond*
var_list	::=	*variable* {**,** *variable* }*
unitary_operator	::=	$CNot \mid H \mid I \mid X \mid Y \mid Z \mid \ldots$
observable	::=	$M_{std} \mid \ldots$
communication	::=	*gate* **!** *value*
	|	*gate* **!** *variable*
	|	*gate* **!** *observable* **[** *var_list* **]**
	|	*gate* **?** *variable*
unit_transf	::=	*unitary_operator* **[** *var_list* **]**
measurement	::=	*observable* **[** *var_list* **]**
action	::=	*communication*
	|	*unit_transf*
	|	*measurement*

$$
\begin{array}{lcl}
\textit{decl\_var} & ::= & \boldsymbol{\theta}[\ \textit{var\_list}\ ] \mid \boldsymbol{\kappa}[\ \textit{var\_list}\ ] \\
\\
\textit{process} & ::= & \textbf{\textit{nil}} \\
& \mid & \textbf{\textit{end}} \\
& \mid & \textit{process\_name}\ [\![\ \textit{var\_list}\ ]\!] \\
& \mid & \{\textit{decl\_var}\}^{+}\ \textbf{:}\ \textit{process}\ \bullet \\
& \mid & \textit{action}\ \textbf{.}\ \textit{process} \\
& \mid & \textit{process}\ \ \textbf{;}\ \textit{process} \\
& \mid & \textit{process}\ \|\ \textit{process} \\
& \mid & \{[\!]\ \textit{cond} \rightarrow \textit{process}\ \}^{+} \\
& \mid & \textit{process}\ |\{\ \textit{gate}\ \{\ \textit{,gate}\ \}^{*}\} \\
\\
\textit{proc\_decl} & ::= & \textit{process\_name} \stackrel{\textbf{\textit{def}}}{=} \textit{process}
\end{array}
$$

B.2 Main inference rules of the semantics

With respect to appendix A.2, two new kinds of transitions have been added:

- declaration transition: $\longrightarrow$, for variable declaration;
- probabilistic transition: $\longrightarrow_p$, where p is a probability.

In the following, C, C' or C_i represent contexts and S_i, a process state.

Variable declaration

$$\frac{}{\kappa[x_1,\ldots,x_n]\ \theta[y_1,\ldots,y_m] : P\ \bullet / C \quad \longrightarrow \quad P\ \bullet / C'}$$

with $C =< s, q = |\psi\rangle, f >$, $C' =< s', q = |\psi\rangle, f >$ and
$s' = \{(x_1, \mathrm{Nat}), \ldots, (x_n, \mathrm{Nat}), (y_1, \mathrm{Qubit}), \ldots, (y_m, \mathrm{Qubit})\}.s$

Successful termination

$$\frac{}{\textit{end}/C \quad \xrightarrow{\ \delta\ } \quad \textit{nil}/C} \quad C\downarrow$$

End of scope of variables

$$\frac{P/C \quad \dashrightarrow \quad P'/C'}{P\ \bullet / C \quad \dashrightarrow \quad P'\ \bullet / C'}$$

where $\dashrightarrow$ stands for any of the transitions: $\xrightarrow{\ \alpha\ }$ with α an action, $\xrightarrow{\ \tau\ }$, or $\longrightarrow$.

$$\frac{P/C \quad \xrightarrow{\ \delta\ } \quad P' / < e.s, q = |\psi\rangle, f >}{P\ \bullet / C \quad \xrightarrow{\ \delta\ } \quad \textit{nil} / < s, q[e \leftarrow *] = |\psi\rangle, f|_{\mathrm{Var}(s)} >}$$

Action Prefix

$$\frac{}{g\ !v\ .P/C \quad \xrightarrow{\ g\ !v\ } \quad P/C} \quad v \in I\!N,\ C\downarrow$$

$$\overline{g\ !x\ .P/C \quad \xrightarrow{g\ !x} \quad P/C}$$

where $C =< s, q = |\psi\rangle, f >$, $x \in \mathrm{Var}(s)$ and $x \in \mathrm{dom}(f)$

$$\overline{g\ ?x\ .P/C \quad \xrightarrow{g\ ?x} \quad P/C}$$

where $C =< s, q = |\psi\rangle, f >$, $x \in \mathrm{Var}(s)$

$$\overline{U[x_1, \ldots, x_n].P/C \quad \xrightarrow{\tau} \quad P/C'}$$

where

- $C =< s, q = |\psi\rangle, f >$, $C' =< s, q = |\psi'\rangle, f >$
- $U \in \mathcal{U}$, $x_1, \ldots, x_n \in \mathrm{Var}(s)$, and $x_1, \ldots, x_n \in q$
- $|\psi'\rangle = \Pi^t.(U \otimes I^{\otimes k}).\Pi|\psi\rangle$
- Π is the permutation matrix which places the x_i's at the head of q and Π^t is the transpose of Π
- $k = \mathrm{size}(q) - n$
- $I^{\otimes k} = \underbrace{I \otimes \cdots \otimes I}_{k}$

$$\overline{M[x_1, \ldots, x_n].P/C \quad \xrightarrow{\tau} \quad P/ \boxplus_{p_i} < s, q = |\psi_i\rangle, f >}$$

with

- $C =< s, q = |\psi\rangle, f >$ (which implies $C \downarrow$)
- $x_1, \ldots, x_n \in \mathrm{Var}(s)$ and $x_1, \ldots, x_n \in q$
- $M \in \mathcal{O}$ with $\sum_i \lambda_i P_i$ as spectral decomposition
- $p_i = \langle\psi|\Pi^t(P_i \otimes I^{\otimes k})\Pi|\psi\rangle$
- $|\psi_i\rangle = \dfrac{\Pi^t(P_i \otimes I^{\otimes k})\Pi|\psi\rangle}{\sqrt{p_i}}$
- Π is the permutation matrix which places the x_i's at the head of q and Π^t is the transpose of Π
- $k = \mathrm{size}(q) - n$

$$\overline{g\ !M[x_1, \ldots, x_n]\ .P/C \quad \xrightarrow{\tau} \quad (g\ !y\ .end\ \bullet)\ ;P/ \boxplus_{p_i} C_i}$$

where

- y is a new variable
- $C =< s, q = |\psi\rangle, f >$ (which implies $C\downarrow$)
- $C_i =< \{(y, \mathrm{Nat})\}.s, q = |\psi_i\rangle, f \Leftarrow \{y \mapsto \lambda_i\} >$
- and the conditions are the same as in the rule without communication.

Sequential composition

$$\frac{P/C \quad \dashrightarrow \quad P'/C'}{P\,;Q/C \quad \dashrightarrow \quad P'\,;Q/C'}$$

where $\dashrightarrow$ stands for any of the transitions : $\xrightarrow{\alpha}$ with α an action different from δ, $\xrightarrow{\tau}$, or $\longrightarrow$.

$$\frac{P/C \quad \xrightarrow{\delta} \quad P'/C'}{P\,;Q/C \quad \xrightarrow{\tau} \quad Q/C'}$$

Parallel composition

In the rules for parallel composition, C, C_P and C_Q are defined as:

- $C =< (s_P \parallel s_Q).s, q = |\psi\rangle, f >$
- $C_P =< s_P.s, q = |\psi\rangle, f >$
- $C_Q =< s_Q.s, q = |\psi\rangle, f >$

In the definition of C, the operator $\parallel$ permits to build a cactus stack (see paragraph 4.1). In the cactus stack $(s_P \parallel s_Q).s$ of the process $P \parallel Q$, the names in s correspond to variables shared by P and Q whereas the names in s_P (resp. s_Q) correspond to variables declared in P (resp. Q).

$$\frac{P/C_P \quad \dashrightarrow \quad P'/C'_P}{P \parallel Q/C \quad \dashrightarrow \quad P' \parallel Q/C'}$$

where

- $\dashrightarrow$ stands for one of those transitions : $\xrightarrow{\alpha}$ with α an action and $\alpha \neq \delta$, $\xrightarrow{\tau}$, $\longrightarrow$
- If $C'_P =< s', q' = |\psi'\rangle, f' >$ then $C' =< (s'_P \parallel s_Q).s, q' = |\psi'\rangle, f' >$ with s'_P such that $s' = s'_P.s$ (P can neither add to nor remove variables from s)
- If $C'_P = \boxplus_{p_i} < s'_i, q'_i = |\psi'_i\rangle, f'_i >$ then $C' = \boxplus_{p_i} < (s_{P'_i} \parallel s_Q).s, q'_i = |\psi'_i\rangle, f'_i >$ with $s_{P'_i}$ such that $s'_i = s_{P'_i}.s$

$$\frac{P/C_P \quad \xrightarrow{g\,!v} \quad P'/C'_P \qquad Q/C_Q \quad \xrightarrow{g\,?x} \quad Q'/C'_Q}{P \parallel Q/C \quad \xrightarrow{\tau} \quad P' \parallel Q'/C'}$$

where

- $x \in \mathrm{Var}(s) \cup \mathrm{Var}(s_Q)$ and $v \in I\!N$
- If x is of type Nat, then:
  $C' =< (s_P \parallel s_Q).s, q = |\psi\rangle, f \leftarrow\!\!+ \{x \mapsto v\} >$
- If x is of type Qubit, then: $x \notin q$, $v \in \{0,1\}$
  and $C' =< (s_P \parallel s_Q).s, x.q = |v\rangle \otimes |\psi\rangle, f >$

$$\frac{P/C_P \quad \xrightarrow{\delta} \quad P'/C'_P \qquad Q/C_Q \quad \xrightarrow{\delta} \quad Q'/C'_Q}{P \parallel Q/C \quad \xrightarrow{\delta} \quad nil/C'}$$

with $C' =< s, q[(\mathrm{Var}(s_P) \cup \mathrm{Var}(s_Q)) \leftarrow *] = |\psi\rangle, f|_{\mathrm{Var}(s)} >$

Probabilistic contexts

$$\frac{}{P/\boxplus_{p_i} C_i \quad \longrightarrow_{p_i} \quad P/C_i} \text{ where } \sum_j p_j = 1$$