Proc. QPL 2004, pp. 109–126

A process algebraic approach to concurrent and distributed quantum computation: operational semantics

Marie Lalire* Philippe Jorrand†

Leibniz Laboratory
46, avenue Félix Viallet 38000 Grenoble, France

Abstract

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. Moreover, to model concurrent and distributed quantum computations, as well as quantum communication protocols, quantum to quantum communications which move qubits physically from one place to another must also be taken into account.

Inspired by classical process algebras, which provide a framework for modeling cooperating computations, a process algebraic notation is defined, named QPAlg for Quantum Process Algebra, which provides a homogeneous style to formal descriptions of concurrent and distributed computations comprising both quantum and classical parts. On the quantum side, QPAlg provides quantum variables, operations on quantum variables (unitary operators and measurement observables), as well as new forms of communications involving the quantum world. The operational semantics make sure that these quantum objects, operations and communications operate according to the postulates of quantum mechanics.

1 Introduction

Quantum algorithms are frequently described by means of quantum gate networks. 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 [8], an imperative language designed by Bernhard Ömer which aims at simulating quantum programs, qGCL [11] by Paolo Zuliani which allows the construction of proved correct quantum programs through a refinement method, and QPL [9], a functional language designed by

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

Peter Selinger with a denotational semantics. A quantum lambda calculus [10], based on a simplified linear lambda calculus, as also been developed by André van Tonder.

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 [2] is another good example of this cooperation. Indeed, Alice carries out a measurement, the result of which (two bits) is sent to Bob, and Bob uses this classical 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 [7], where Simon Gay and Rajagopal Nagarajan have modeled a quantum cryptographic protocol and verified its correctness with a classical process algebra. Starting with a classical process algebra described in appendix A, this paper explains how to "quantumize" it in section 2. Examples of short quantum programs are given in section 3.

2 "Quantumized" Processes

2.1 Quantum Variables

For the purpose of this paper, we consider that 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. An extended version of the process algebra would of course also include quantum registers and other types of variables.

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: $[\, x_1 : t_1, \ldots, x_n : t_n \,.\, P \,]$ where $x_1, \ldots, x_n$ is a list of variables, $t_1, \ldots, t_n$ are their types, and P is a process which can make use of these classical and quantum variables. To simplify the rest of this paper, the names of variables will always be considered distinct.

In the inference rules which describe the semantics of processes, the states of processes can no longer be process terms only, as 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 : *names* $\rightarrow$ *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 Var(s), "." adds an element on top of a stack, and "|" concatenates two stacks.

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:

$$\overline{[\, x_1 : t_1, \ldots, x_n : t_n \,.\, P \,]/C \quad \longrightarrow \quad [\, P \,]/C'}$$

with $C =< s, q = |\psi\rangle, f >$, $C' =< s', q = |\psi\rangle, f >$
and $s' = \{(x_1, t_1), \ldots, (x_n, t_n)\}.s$

This rule adds the new variable names and types on top of 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:

$$\frac{P/C \quad \dashrightarrow \quad P'/C'}{[\, P \,]/C \quad \dashrightarrow \quad [\, P' \,]/C'}$$

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 $[\, P \,]$ 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:

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

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 ($f\backslash_E$ means f restricted to E). As regards 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.

2.2 Basic Actions

The classical basic actions are classical to classical communications. Classical to quantum communications and quantum to quantum communications are introduced for respectively initializing qubits and allowing the description of communication protocols. 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:

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

$$\frac{}{g\,!x\,.P/C \quad \xrightarrow{g\,!x} \quad P/C'}$$

where

- $C =< s, q = |\psi\rangle, f >$ and $C' =< s\backslash\{x\}, q[x \leftarrow *] = |\psi\rangle, f >$
- $x \in \text{Var}(s)$ and $x \in q$

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

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

The first rule deals with classical value sending, the second one, with qubit sending, and the last one, with value reception. For qubit sending (second rule), because of the no-cloning theorem, the sent qubit must be removed from the context. It should be noted that in the third rule, the variable x can be classical or quantum but, if it is quantum, it must not have already been initialized.

In the operational semantics of parallel composition, the combination of these rules defines communication. In a classical to quantum communication, the qubit is initialized in the basis state $|v\rangle$, where v is the classical value sent (in this case, v must be 0 or 1). In a quantum to quantum communication, the name of the sent qubit is replaced in q by the name of the receiving qubit.

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

- $C =< s, q = |\psi\rangle, f >$, $C' =< s, q = |\psi'\rangle, f >$
- $U \in \mathcal{U}$, $x_1, \ldots, x_n \in \text{Var}(s)$, and $x_1, \ldots, x_n \in q$
- $\forall\, i, j \in \{0, \ldots, n\}$ such that $i \neq j :\ x_i \neq x_j$
- $|\psi'\rangle = \Pi^t.(U \otimes I^{\otimes k}).\Pi|\psi\rangle$
- Π is the permutation matrix which places the x_i's at the head of q and Π^t is the transpose of Π
- $k = \text{size}(q) - n$
- $I^{\otimes k} = \underbrace{I \otimes \cdots \otimes I}_{k}$, 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 fourth 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 placed back in the same order.

2.3 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 distinct 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.

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 adding to 1.

As explained in [4] and [5], 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, a probabilistic transition must be performed first.

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 $C \downarrow$)
- $x_1, \ldots, x_n \in \text{Var}(s)$ and $x_1, \ldots, x_n \in q$
- $\forall\, i, j \in \{0, \ldots, n\}$ such that $i \neq j :\ x_i \neq x_j$
- $M \in \mathcal{O}$ with $\sum_i \lambda_i P_i$ as spectral decomposition
- $p_i = \langle\psi|\Pi^t(P_i \otimes I^{\otimes k})\Pi|\psi\rangle$ and $|\psi_i\rangle = \dfrac{\Pi^t(P_i \otimes I^{\otimes k})\Pi|\psi\rangle}{\sqrt{p_i}}$
- 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.

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

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

where

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

The only remaining point is the evolution of processes within 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.

3 Examples

In the following examples, the set $\mathcal{U}$ of unitary transformations is:

$$\mathcal{U} = \{H, CNot, I, X, Y, Z\}$$

where H is Hadamard transformation, $CNot$ is the "controlled not" operation, I is the identity, and X, Y, Z are Pauli matrices. The set $\mathcal{O}$ of observables contains the observables corresponding to measurement of one and two qubits in the standard basis, denoted respectively $M_{std,1}$ and $M_{std,2}$, and the observable corresponding to measurement of a qubit in the basis $\{|+\rangle, |-\rangle\}$, denoted M_{+-}.

3.1 Construction of an EPR pair

$$\begin{aligned}\textbf{BuildEPR} \stackrel{\text{def}}{=} \; & [\, x : \text{Qubit}, y : \text{Qubit} \,. \\ & \quad ((g_1\, ?x \,.g_2\, ?y \,.H[x].CNot[x,y].\textit{end}) \\ & \quad \parallel (g_1\, !0 \,.g_2\, !0 \,.\textit{end}))\backslash\{g_1, g_2\} \\ &]\end{aligned}$$

This process puts the pair of qubits x, y in the state $|EPR\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. To check that the order of measurement of the two qubits does not matter, it is possible, using the inference rules, to analyze the behavior 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.

$$\begin{aligned}\textbf{CheckEPR}_\textbf{1} \stackrel{\text{def}}{=} \; & [\, a : \text{Qubit}, b : \text{Qubit} \,. \\ & \quad \textbf{BuildEPR}[a,b] \,;\, M_{std,1}[a].M_{std,1}[b].\textit{end} \\ &]\end{aligned}$$

$$\begin{aligned}\textbf{CheckEPR}_\textbf{2} \stackrel{\text{def}}{=} \; & [\, a : \text{Qubit}, b : \text{Qubit} \,. \\ & \quad \textbf{BuildEPR}[a,b] \,;\, M_{std,1}[b].M_{std,1}[a].\textit{end} \\ &]\end{aligned}$$

3.2 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 obtain information about α and β. Nevertheless, Alice succeeded thanks to the EPR pair and the teleportation protocol [2]:

$$\begin{array}{l}
\textbf{Alice} \stackrel{\text{def}}{=} [\, x : \text{Qubit}, y : \text{Qubit} \,. \\
\qquad\quad CNot[x,y].H[x].\textit{meas}\,!M_{std,2}[x,y]\ .\textit{end} \\
\quad\ ]
\end{array}$$

$$\begin{array}{l}
\textbf{Bob} \stackrel{\text{def}}{=} [\, z : \text{Qubit} \,. \\
\qquad [\, k : \text{Nat} \,. \\
\qquad\quad \textit{meas}\ ?k \,. \\
\qquad\quad [\, k = 0 \rightarrow I[z].\textit{end}, \\
\qquad\quad\ k = 1 \rightarrow X[z].\textit{end}, \\
\qquad\quad\ k = 2 \rightarrow Z[z].\textit{end}, \\
\qquad\quad\ k = 3 \rightarrow Y[z].\textit{end}\,] \\
\qquad\ ] \\
\quad\ ]
\end{array}$$

$$\begin{array}{l}
\textbf{Teleport} \stackrel{\text{def}}{=} [\, \psi : \text{Qubit} \,. \\
\qquad [\, a : \text{Qubit}, b : \text{Qubit} \,. \\
\qquad\quad \textbf{BuildEPR}[a,b]\ ; \\
\qquad\quad (\textbf{Alice}[\psi,a]\ \|\ \textbf{Bob}[b]) \backslash \{\textit{meas}\} \\
\qquad\ ] \\
\quad\ ]
\end{array}$$

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.

3.3 Communication protocols

Alice sends qubits to Bob through a non secure channel and Eve eavesdrops this channel to get information on the qubits sent by Alice. In the following example **A**, **B**, and **E** are processes modeling whatever Alice, Bob, and Eve may respectively apply to their qubits. The actions of these processes, which are not made explicit here, will be specified in the next example of the BB84 protocol.

The communication protocols which are described here could be used to model cryptographic protocols so as to check if they are secure.

Eve intercepts all qubits

Eve intercepts qubits because of a flaw in the channel that Alice and Bob use to communicate.

$$\textbf{Alice} \stackrel{\text{def}}{=} [\, a : \text{Qubit} \,.\, \mathbf{A}[a] \,;\, \textit{fill}\ !a\ \textit{.end} \,] \,;\, \textbf{Alice}$$

$$\textbf{Bob} \stackrel{\text{def}}{=} [\, b : \text{Qubit} \,.\, \textit{empty}\ ?b\ .\mathbf{B}[b] \,] \,;\, \textbf{Bob}$$

$$\textbf{Eve} \stackrel{\text{def}}{=} [\, e : \text{Qubit}, f : \text{Qubit} \,.\, \textit{emptyFlaw}\ ?e\ .\mathbf{E}[e, f] \,;\, \textit{fillFlaw}\ !f\ \textit{.end} \,] \,;\, \textbf{Eve}$$

$$\textbf{Flaw} \stackrel{\text{def}}{=} [\, u : \text{Qubit}, v : \text{Qubit} \,.\, \textit{emptyFlaw}\ !u\ .\textit{fillFlaw}\ ?v\ \textit{.end} \,]$$

$$\textbf{Channel} \stackrel{\text{def}}{=} [\, x : \text{Qubit}, y : \text{Qubit} \,.\, \textit{fill}\ ?x\ .\mathbf{Flaw}[x, y] \,;\, \textit{empty}\ !y\ \textit{.end} \,] \,;\, \textbf{Channel}$$

$$\textbf{Protocol} \stackrel{\text{def}}{=} (\textbf{Alice} \parallel \textbf{Bob} \parallel \textbf{Eve} \parallel \textbf{Channel}) \setminus \{\, \textit{fill}, \textit{empty}, \textit{fillFlaw}, \textit{emptyFlaw} \}$$

Eve intercepts some of the qubits

This part requires that a nondeterministic process composition $P + Q$ be introduced in the process algebra. This can be done, provided that probabilistic choices are always solved first (this operator is not presented in the operational semantics in appendix B).

To model the fact that Eve does not succeed in intercepting all qubits, the flaw in the channel is made nondeterministic:

$$\begin{aligned}\textbf{Channel} \stackrel{\text{def}}{=}\ & [\, x : \text{Qubit} \,. \\ & \quad \textit{fill}\ ?x\ . \\ & \quad (\\ & \qquad [\, y : \text{Qubit} \,.\, \mathbf{Flaw}[x, y] \,;\, \textit{empty}\ !y\ \textit{.end} \,] + \\ & \qquad (\textit{empty}\ !x\ \textit{.end}) \\ & \quad) \\ &] \,;\, \textbf{Channel}\end{aligned}$$

3.4 The BB84 Protocol

The BB84 protocol [1] is a protocol of quantum key distribution: Alice and Bob must agree on a private key, i.e. a list of bits that should remain secret. To communicate, they can send qubits through a non secure channel. In fact, the processes **A** and **B** left unspecified in the previous paragraph can be used to model this protocol. The process **Alice** is redefined and the process **B** used by **Bob** is made explicit. In addition, another process is defined: the process **Random** which initializes a bit randomly at 0 or 1. The gates *keepDataA* and

keepDataB are used by Alice and Bob respectively to send the bits that they want to keep. In this example, we take the liberty of using identical names for variables having distinct scopes.

$$\textbf{Alice} \stackrel{\text{def}}{=} [\, a : \text{Qubit}, \mathit{dataA} : \text{Nat}, \mathit{baseA} : \text{Nat} \,.\; \mathbf{A}_1[a, \mathit{dataA}, \mathit{baseA}] \,;\, \mathit{fill}\ !a\ .\mathbf{A}_2[\mathit{dataA}, \mathit{baseA}] \,] \,;\, \textbf{Alice}$$

$$\textbf{Random} \stackrel{\text{def}}{=} [\, r : \text{Nat} \,.\; [\, x : \text{Qubit} \,.\; (g\ !0\ .\mathit{end} \parallel g\ ?x\ .\mathit{end}) \backslash \{g\}\ ;\; H[x].\; (h\ !M_{std,1}[x]\ .\mathit{end} \parallel h\ ?r\ .\mathit{end}) \backslash \{h\} \,] \,]$$

$$\mathbf{A}_1 \stackrel{\text{def}}{=} [\, a : \text{Qubit}, \mathit{dataA} : \text{Nat}, \mathit{baseA} : \text{Nat} \,.\; \textbf{Random}[\mathit{dataA}][a]\ ;\; \textbf{Random}[\mathit{baseA}]\ ;\; [\, \mathit{baseA} = 1 \rightarrow H[a].\mathit{end} \,] \,]$$

$$\mathbf{A}_2 \stackrel{\text{def}}{=} [\, \mathit{dataA} : \text{Nat}, \mathit{baseA} : \text{Nat} \,.\; [\, \mathit{bool} : \text{Nat}, \mathit{ok} : \text{Nat} \,.\; \mathit{received}\ ?\mathit{ok}\ .\; \mathit{base}\ !\mathit{baseA}\ .\; \mathit{keep}\ ?\mathit{bool}\ .\; [\, \mathit{bool} = 1 \rightarrow \mathit{keepDataA}\ !\mathit{dataA}\ .\mathit{end} \,] \,] \,]$$

$$\mathbf{B} \stackrel{\text{def}}{=} [\, b : \text{Qubit} \,.\; [\, \mathit{baseA} : \text{Nat}, \mathit{baseB} : \text{Nat}, \mathit{dataB} : \text{Nat} \,.\; \textbf{Random}[\mathit{baseB}]\ ;\; (\; [\, \mathit{baseB} = 0 \rightarrow g\ !M_{std,1}[b]\ .\mathit{end},\; \mathit{baseB} = 1 \rightarrow g\ !M_{+-}[b]\ .\mathit{end} \,] \parallel g\ ?\mathit{dataB}\ .\mathit{end} \;)\backslash\{g\}\ ;\; \mathit{received}\ !1\ .\; \mathit{base}\ ?\mathit{baseA}\ .\; [\, \mathit{baseA} = \mathit{baseB} \rightarrow \mathit{keep}\ !1\ .\mathit{keepDataB}\ !\mathit{dataB}\ .\mathit{end},\; \mathit{baseA} \neq \mathit{baseB} \rightarrow \mathit{keep}\ !0\ .\mathit{end} \,] \,] \,]$$

4 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 above protocols 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 branches and probabilistic branches.

Several extensions are possible. As already mentioned, a nondeterministic process composition operator can be introduced. A probabilistic composition of processes could be added. This would allow, for example, the description of communication protocols in which Eve intercepts qubits with a given probability. Another track that could be followed is the use of density matrices, which are a more general description of quantum states than vectors in $\mathbb{C}^{2^n}$. This may also open the way to a semantic analysis similar to abstract interpretation. Another direction of study concerns the definition of an equivalence among processes, which is necessary for obtaining a more abstract semantics.

References

[1] C. H. Bennett and G. Brassard. Quantum cryptography: Public-key distribution and coin tossing. In *Proceedings of the IEEE International Conference on Computer, Systems and Signal Processing*, pages 175–179, Bangalore, India, December 1984.

[2] 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.

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

[4] 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.

[5] 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.

[6] R. Milner. *Communication and Concurrency*. Prentice-Hall, London, 1999.

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

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

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

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

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

A A Classical Process Algebra

The classical process algebra chosen here is quite similar to CCS [6] and Lotos [3]. 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 ($[\, c_1 \rightarrow P_1, \ldots, c_n \rightarrow P_n \,]$) and restriction ($P \backslash L$). 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 $[\, c_1 \rightarrow P_1, \ldots, c_n \rightarrow P_n \,]$, 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 on gate g in the process $g\,!v\,.P \parallel g\,?x\,.Q$. 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.

A.1 Syntax of Process Terms

process ::= ***nil***
| ***end***
| *communication* **.** *process*
| *process* **;** *process*
| *process* **‖** *process*

	|	*process* \\{ *gate_list* }
	|	[*cond_list*]
	|	*process_name*
communication	::=	*gate* **!** *exp* | *gate* **?** *variable*
cond	::=	*bexp* → *process*
proc_def	::=	*process_name* $\stackrel{\textbf{def}}{=}$ *process*

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, and 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.

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

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

Action Prefix

$$\frac{}{g\,!v\,.P \xrightarrow{g\,!v} P} \quad v \in I\!N \qquad\qquad \frac{}{g\,?x\,.P \xrightarrow{g\,?x} P}$$

Sequential composition

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

Parallel composition

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

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

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

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

Conditional choice

$$\frac{P_i \xrightarrow{\ \alpha_i\ } P_i'}{[\, c_1 \rightarrow P_1, \ldots, c_n \rightarrow P_n \,] \xrightarrow{\ \alpha_i\ } P_i'} \quad c_i$$

$$\frac{}{[\, c_1 \rightarrow P_1, \ldots, c_n \rightarrow P_n \,] \xrightarrow{\ \delta\ } \mathit{nil}} \quad \forall i, \neg c_i$$

Restriction

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

B The Quantum Process Algebra

B.1 Syntax

process	::=	***nil***
	|	***end***
	|	*action* **.** *process*
	|	*process* **;** *process*
	|	*process* **‖** *process*
	|	*process* **\{** *gate_list* **}**
	|	**[** *cond_list* **]**
	|	**[** *var_decl_list* **.** *process* **]**
	|	*process_name* [**[** *var_list* **]**]
action	::=	*communication*
	|	*unit_transf*
	|	*measurement*
communication	::=	*gate* **!** *exp*
	|	*gate* **!** *measurement*
	|	*gate* **?** *variable*
unit_transf	::=	*unitary_operator* **[** *var_list* **]**

measurement ::= *observable* [*var_list*]

var_decl ::= *variable* : *var_type*

proc_def ::= *process_name* $\stackrel{def}{=}$ *process*

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 are contexts and S_i is a process state.

Successful termination

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

Action Prefix

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

$$\frac{}{g\,!x\,.P/C \quad \xrightarrow{g\,!f(x)} \quad P/C}$$

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

$$\frac{}{g\,!x\,.P/C \quad \xrightarrow{g\,!x} \quad P/C'}$$

where

- $C =< s, q = |\psi\rangle, f >$ and $C' =< s\backslash\{x\}, q[x \leftarrow *] = |\psi\rangle, f >$
- $x \in \text{Var}(s)$ and $x \in q$

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

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

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

where

- $C =< s, q = |\psi\rangle, f >$, $C' =< s, q = |\psi'\rangle, f >$
- $U \in \mathcal{U}$, $x_1,\ldots,x_n \in \text{Var}(s)$, and $x_1,\ldots,x_n \in q$

- $\forall\, i,j \in \{0,\ldots,n\}$ such that $i \neq j :\ x_i \neq x_j$
- $|\psi'\rangle = \Pi^t.(U \otimes I^{\otimes k}).\Pi|\psi\rangle$
- Π is the permutation matrix which places the x_i's at the head of q and Π^t is the transpose of Π
- $k = \text{size}(q) - n$ and $I^{\otimes k} = \underbrace{I \otimes \cdots \otimes I}_{k}$

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

with

- $C =< s, q = |\psi\rangle, f >$ (which implies $C\downarrow$)
- $x_1,\ldots,x_n \in \text{Var}(s)$ and $x_1,\ldots,x_n \in q$
- $\forall\, i,j \in \{0,\ldots,n\}$ such that $i \neq j :\ x_i \neq x_j$
- $M \in \mathcal{O}$ with $\sum_i \lambda_i P_i$ as spectral decomposition
- $p_i = \langle\psi|\Pi^t(P_i \otimes I^{\otimes k})\Pi|\psi\rangle$ and $|\psi_i\rangle = \dfrac{\Pi^t(P_i \otimes I^{\otimes k})\Pi|\psi\rangle}{\sqrt{p_i}}$
- Π is the permutation matrix which places the x_i's at the head of q and Π^t is the transpose of Π
- $k = \text{size}(q) - n$

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

where

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

Probabilistic contexts

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

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 2.1). 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

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

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

where

- $x \in \text{Var}(s) \cup \text{Var}(s_P)$, $x \in q$
- $y \in \text{Var}(s) \cup \text{Var}(s_Q)$, $y \notin q$, y of type Qubit
- $C' =< ((s_P \parallel s_Q).s) \backslash \{x\}, q[x \leftarrow y] = |\psi\rangle, f >$

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

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

Variable declaration

$$\frac{}{[\, x_1 : t_1, \ldots, x_1 : t_n \,.\, P \,]/C \quad \longrightarrow \quad [\, P \,]/C'}$$

with $C =< s, q = |\psi\rangle, f >$, $C' =< s', q = |\psi\rangle, f >$
and $s' = \{(x_1, t_1), \ldots, (x_n, t_n)\}.s$

End of scope of variables

$$\frac{P/C \quad \dashrightarrow \quad P'/C'}{[\, P \,]/C \quad \dashrightarrow \quad [\, P' \,]/C'}$$

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

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