Math. Struct. in Comp. Science (2006), *vol.* 16, *pp.* 407–428. © 2006 Cambridge University Press
doi:10.1017/S096012950600524X Printed in the United Kingdom

Relations among quantum processes: bisimilarity and congruence

MARIE LALIRE

Leibniz Laboratory – Grenoble, France

Received 25 January 2005; revised 23 November 2005

Full formal descriptions of algorithms making use of quantum principles must take into account both quantum and classical computing components, as well as communications between these components. Moreover, to model concurrent and distributed quantum computations and quantum communication protocols, communications over quantum channels that move qubits physically from one place to another must also be taken into account.

Inspired by classical process algebras, which provide a framework for modelling cooperating computations, a process algebraic notation is defined. This notation provides a homogeneous style for formal descriptions of concurrent and distributed computations comprising both quantum and classical parts. Based upon an operational semantics that makes sure that quantum objects, operations and communications operate according to the postulates of quantum mechanics, an equivalence is defined among process states considered as having the same behaviour. This equivalence is a probabilistic branching bisimulation. From this relation, an equivalence on processes is defined. However, it is not a congruence because it is not preserved by parallel composition.

1. Introduction

The number of quantum programming languages is growing rapidly. These languages can be classified into three families: imperative, functional, and parallel and distributed. Among imperative programming languages, there are QCL (Ömer 2000), designed by Ömer, which aims at simulating quantum programs, and qGCL (Zuliani 2001) by Zuliani, which allows the construction by refinement of quantum programs that have been proved to be correct. QPL (Selinger 2004) is a functional language designed by Selinger with a denotational semantics. Several quantum λ-calculi have also been developed: see, for example, van Tonder(2003), where the calculus is based on a simplified linear λ-calculus, and Arrighi and Dowek (2004), where the calculus is a 'linear-algebraic λ-calculus'. Gay and Nagarajan have developed CQP (Nagarajan and Gay 2004), which is a language to describe communicating quantum processes and is based on π-calculus. An important point in their work is the definition of a type system, and the proof that the operational semantics preserves typing.

Cooperation between quantum and classical computations is inherent in quantum algorithmics. Teleportation of a qubit state from Alice to Bob (Bennett *et al.* 1993) is a 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

https://doi.org/10.1017/S096012950600524X Published online by Cambridge University Press

quantum transformation he must apply. Moreover, initial preparation of quantum states and measurement of quantum results are two essential forms of interaction between the classical and quantum parts of computations that a language must be able to express. Process algebras are a good candidate for such a language since they provide a framework for modelling 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. Their semantics give rise to an equivalence relation on processes. Bisimilarity is an adequate equivalence relation to deal with communicating processes since it relates processes that can execute the same flows of actions while having the same branching structure.

This paper first presents the main points of the definition and semantics of a Quantum Process Algebra (QPAlg). Then, a probabilistic branching bisimilarity is defined among process states (Section 3), and this relation is proved to be an equivalence. As an example, in Section 4, the application of the Hadamard unitary transformation is proved to be bisimilar with its simulation with measurement only, based on state transfer. Finally, in Section 5, an equivalence among processes is defined. This relation is preserved by all the operators of QPAlg except parallel composition.

2. Definition of QPAlg

The process algebra QPAlg is based on process algebras such as CCS (Milner 1989) and Lotos (Bolognesi and Brinksma 1987). The key aspects of QPAlg are developed in this section. The precise syntax and the main inference rules of the semantics are given in appendix A. For more details and more examples, see Lalire and Jorrand (2004).

2.1. *Variables*

For the purposes of this paper, we consider two types of variables, one classical, `Nat`, for variables taking natural 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 with value passing (Milner 1989; Bolognesi and Brinksma 1987; Roscoe 1998), 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.

Here, quantum variables represent physical qubits. Applying a transformation to a variable that represents a qubit modifies the state of that qubit. This means that values of variables are modified. For that reason, a process state must keep track of both variable names and variable states: this will be achieved through the context (*cf.* Section 2.5).

Variables are declared using the syntax $\mathbf{[}\ x_1 : t_1, \ldots, x_n : t_n\ .\ P\ \mathbf{]}$ where $x_1, \ldots, x_n$ is a list of variables, $t_1, \ldots, t_n$ are their types, and P is a process that can make use of these classical and quantum variables.

https://doi.org/10.1017/S096012950600524X Published online by Cambridge University Press

2.2. *Expressions*

The quantum expressions are quantum variables or tensor products of quantum variables, denoted $x_1 \otimes \cdots \otimes x_n$.

The classical expressions are the usual classical expressions, and application of an admissible transformation to a quantum expression. Admissible transformations are also called general quantum measurements, and include unitary transformations. For more details, see Nielsen and Chuang (2000).

Let $\mathscr{A}$ be a set of predefined admissible transformations. The application of the admissible transformation $A = \{A_{\tau_1}, \ldots, A_{\tau_m}\} \in \mathscr{A}$, of dimension $2^n \times 2^n$, to the register of qubits $x_1 \otimes \cdots \otimes x_n$ is denoted $A[x_1 \otimes \cdots \otimes x_n]$.

The quantum memory is stored in the context in the form $q = \rho$ where q is the list of quantum variable names and ρ is a density matrix representing their quantum state (*cf.* Section 2.5). If the classical result of $A[x_1 \otimes \cdots \otimes x_n]$ is τ_i, then $\mathscr{T}_{A_{\tau_i}}$ is the super-operator that must be applied to the density matrix ρ to describe the evolution of the quantum memory $q = \rho$:

$$\mathscr{T}_{A_{\tau_i}} : \rho \mapsto \Pi^\dagger.(A_{\tau_i} \otimes I^{\otimes k}).\Pi.\rho.\Pi^\dagger.(A_{\tau_i}^\dagger \otimes I^{\otimes k}).\Pi$$

where

— Π is the permutation matrix that places the x_i's at the head of q
— $k = \mathtt{size}(q) - n$
— $I^{\otimes k} = \underbrace{I \otimes \cdots \otimes I}_{k}$, where I is the identity matrix on $\mathbb{C}^2$.

Since the admissible transformation A_{τ_i} may be applied to qubits that 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 A_{τ_i} can be applied to the first n elements and I to the remainder. Finally, the last operation is the inverse of the permutation Π so that at the end, the arrangement of the elements in ρ is consistent with the order of the elements in q.

$A[x_1 \otimes \cdots \otimes x_n]$ is not probabilistic, but its evaluation produces a result with a probabilistic value. This value is stored in the context, which becomes probabilistic (*cf.* Section 2.5).

In the examples given in this paper, the set $\mathscr{A}$ of admissible transformations is

$$\mathscr{A} = \{H, CNot, I, \sigma_X, \sigma_Y, \sigma_Z, M_{std,1}, M_{std,2}, X, Z \otimes X\}$$

where H is a Hadamard transformation, $CNot$ is 'controlled not', I is the identity, and σ_X, σ_Y, σ_Z are Pauli operators. $M_{std,1}$ and $M_{std,2}$ correspond to measurement in the standard basis of one and two qubits, respectively. X and $Z \otimes X$ are the admissible transformations corresponding to the measurements with the Pauli observables X and $Z \otimes X$.

2.3. *Basic actions*

The basic actions of QPAlg are classical expressions and communications. A classical expression is interesting as a basic action if it has side effects, as is the case for the application of an admissible transformation.

https://doi.org/10.1017/S096012950600524X Published online by Cambridge University Press

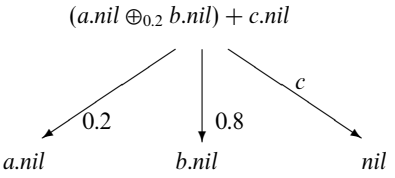


Figure 1. Nondeterminism solved before probabilistic choice

There are several kinds of communications, depending on the type of the expression sent and the type of the receiving variable. The different kinds of communications are: classical to classical communications; classical to quantum communications for initialising qubits; and quantum to quantum communications for allowing the description of quantum communication protocols. Communication gates are not typed, but we can imagine a subsequent version of QPAlg where communication gates would be declared with a fixed type like the other variables.

Emission of an expression e from a gate g is denoted $g\ !e$, and reception in a variable x is denoted $g\ ?x$. In the operational semantics of parallel composition (Rules 15 to 20 of the semantics given in appendix A.2), the combination of the rules for emission and reception defines communication. In a classical to quantum communication (Rule 17), the qubit is initialised in the basis state $|v\rangle\langle v|$, where v is the classical value sent (in this case, v must be 0 or 1). In a quantum to quantum communication (Rule 18), the name of the sent qubit is replaced in the context by the name of the receiving qubit.

2.4. *Composition operators*

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.

The predefined process *nil* cannot perform any transition.

The operators of the process algebra are: parallel composition ($P \parallel Q$); nondeterministic choice ($P + Q$); probabilistic choice ($P \oplus_p Q$); conditional choice ($[\![c_1 \rightarrow P_1, \ldots, c_n \rightarrow P_n]\!]$); and restriction ($P \backslash L$). 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 use the same gate name.

The process $P \oplus_p Q$ behaves like P with probability p and like Q with probability $1 - p$. As explained in Cazorla *et al.* (2001; 2003), and shown in the example of Figure 1, if a process contains both a probabilistic and a nondeterministic choice, then the probabilistic choice must always be solved first. Otherwise, a probabilistic transition labelled with a probability p does not mean that this transition will be executed with probability p.

To solve probabilistic choices before nondeterministic ones, we define the notion of stability for processes.

https://doi.org/10.1017/S096012950600524X Published online by Cambridge University Press

Definition 1. Probabilistic stability is defined by induction:

1 *nil*, $a.P$, $[\, x_1 : t_1, \ldots x_n : t_n \,.\, P \,]$ are stable.
2 $P\backslash L$, $[\, x_1 : t_1, \ldots, x_n : t_n \,] \bullet P[a_1, \ldots, a_n]$ are stable if and only if P is stable.
3 $P + Q$, $P \parallel Q$ are stable if and only if P and Q are stable.
4 $[\, c_1 \rightarrow P_1 \ldots c_n \rightarrow P_n \,]$ is stable if and only if for all i, P_i is stable.

P stable is denoted $P \downarrow$.

In the examples, another operator on processes is used, *viz*. ';', for sequential composition. $P\ ;Q$ behaves like Q once P has terminated. This require the introduction of a predefined process *end*, for signaling successful termination. The operator ';' can be simulated with '$\parallel$': that is, $P\ ;Q$ behaves as $(P \parallel \delta?.Q)\backslash\{\delta\}$ where δ is a fresh gate name and with $end \stackrel{\text{def}}{=} \delta!.nil$.

2.5. *Contexts and process states*

In the inference rules that describe the semantics of processes, the states of processes are 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 $q = \rho$ where q is a sequence of quantum variable names and ρ is a density matrix representing their current quantum state. In order to treat classical variables in a similar way, modifications of classical variables are allowed. So, for the same reason as for quantum variables, classical values are stored in the context. Storing and retrieving classical values is represented by functions $f : names \rightarrow values$. The context keeps track of the embedding of variable scopes. To keep track of parallel composition, we use 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 $\mathtt{Var}(s)$, and '.' adds an element on top of a stack, and '|' concatenates two stacks.

Definition 2. A **context** is a tuple $< s, q = \rho, f >$, where:

— s is the environment stack
— q is a sequence of quantum variable names
— ρ is a density matrix representing the quantum state of the variables in q
— f is the function that associates values with classical variables.

The evaluation of an admissible transformation (Rule 2 of the semantics) produces a probabilistic result. This requires the introduction of a probabilistic composition operator for contexts. This operator is denoted $\boxplus_p$, so the state $P/C_1 \boxplus_p C_2$ is P/C_1 with probability p and P/C_2 with probability $1-p$. In general, a context is either of the form $< s, q = \rho, f >$ or $\boxplus_{p_i} < s_i, q_i = \rho_i, f_i >$ where the p_i's are probabilities adding to 1.

As in the case of probabilities introduced by the operator $\oplus_p$, 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 $< s, q = \rho, f >$. If the context of a process state is not stable, a probabilistic transition must be performed first (Rule 10 of the semantics).

https://doi.org/10.1017/S096012950600524X Published online by Cambridge University Press

2.6. *Example: teleportation*

The teleportation protocol (Bennett *et al.* 1993) transfers the state of a qubit in a place A into a qubit in a place B with only two classical bits sent from place A to place B:

> *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.*

This protocol is described with QPAlg in program 1.

Program 1 Implementation of the teleportation protocol

$$
\begin{array}{rl}
\textbf{BuildEPR} \stackrel{\text{def}}{=} & [\ x : \texttt{Qubit}, y : \texttt{Qubit}\ ]\bullet \\
& ((g_1\ ?x\ .g_2\ ?y\ .H[x].CNot[x,y].end) \\
& \parallel (g_1\ !0\ .g_2\ !0\ .end))\backslash\{g_1, g_2\} \\
\\
\textbf{Alice} \stackrel{\text{def}}{=} & [\ x : \texttt{Qubit}, y : \texttt{Qubit}\ ]\bullet \\
& CNot[x,y].H[x].g\ !M_{std,2}[x,y]\ .end \\
\\
\textbf{Bob} \stackrel{\text{def}}{=} & [\ z : \texttt{Qubit}\ ]\bullet \\
& [\ k : \texttt{Nat}\ . \\
& \quad g\ ?k\ . \\
& \quad [\ k = 0 \rightarrow I[z].end, \\
& \quad k = 1 \rightarrow \sigma_X[z].end, \\
& \quad k = 2 \rightarrow \sigma_Z[z].end, \\
& \quad k = 3 \rightarrow \sigma_Y[z].end\ ] \\
&] \\
\\
\textbf{Teleport} \stackrel{\text{def}}{=} & [\ \psi : \texttt{Qubit}\ ]\bullet \\
& [\ a : \texttt{Qubit}, b : \texttt{Qubit}\ . \\
& \quad \textbf{BuildEPR}[a,b]\ ; \\
& \quad (\textbf{Alice}[\psi, a] \parallel \textbf{Bob}[b])\backslash\{g\} \\
&]
\end{array}
$$

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. Probabilistic branching bisimilarity

The operational semantics associates a process graph with a process state. A process graph is a set of transitions between process states, and an initial state (Fokkink 2000).

https://doi.org/10.1017/S096012950600524X Published online by Cambridge University Press

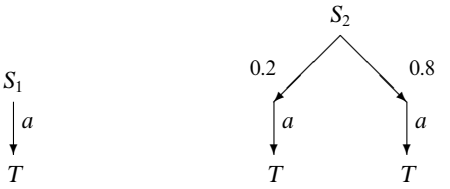


Figure 2. Equivalent states

The transitions are: action transitions, $S \xrightarrow{a} T$, where S, T are states, and a is an action (possibly the internal action τ); or probabilistic transitions, $S \longrightarrow_p T$, where p is a probability.

In this section, an equivalence relation on process states is defined. This is probabilistic branching bisimilarity, which identifies states when they are associated with process graphs having the same branching structure. The bisimilarity defined here is probabilistic because of the probabilistic transitions introduced by quantum measurement and by the operator of probabilistic choice. The choice of a branching bisimilarity comes from the fact that it abstracts from silent transitions (contrary to strong bisimilarity), but is finer than any other equivalence taking into account silent steps (van Glabbeek 1993).

This definition is inspired from the definitions in Fokkink (2000) and Andova (1999).

3.1. *Preliminary definitions and notation*

Process states The set of all possible process states is denoted $\mathcal{S}$. Let $S, T \in \mathcal{S}$. Then S can be written P/C_P and T can be written Q/C_Q, where P, Q are process terms and C_P, C_Q contexts (possibly probabilistic).

Assuming that $S = P/C_P$ and $C_P =< s, q = \rho, f >$, if x is an initialised qubit in S, that is, $(x, \texttt{Qubit}) \in s$ and $x \in q$, then ρ_x^S is the state of x and this state can be obtained with a trace out operation on ρ:

$$\rho_x^S = Tr_{\{x\}/q}(\rho).$$

Silent transitions Internal transitions ($\xrightarrow{\tau}$) are, of course, considered to be silent, but so too are probabilistic transitions. The reason is that we want, for example, the states S_1 and S_2 described in Figure 2 to be equivalent.

Silent transitions will be denoted $\rightsquigarrow$. We use $\rightsquigarrow^*$ to stand for a sequence (possibly empty) of silent transitions, and $\rightsquigarrow^{0..1}$ for zero or one silent transitions.

Function μ Let $\mathscr{R}$ be an equivalence on process states, S be a process state and $\bar{S}$ be its equivalence class with respect to $\mathscr{R}$. If M is a set of process states, then $S \triangleright M$ means that there remains in $M \cup \bar{S}$ a sequence of transitions from S to a state in M.

A function $\mu_{\mathscr{R}} : \mathcal{S} \times \mathscr{P}(\mathcal{S}) \to [0, 1]$ is defined by $\mu_{\mathscr{R}} : (S, M) \mapsto p$, where p is the probability to reach a state in the set M from a state S without leaving $\bar{S} \cup M$.

It should be noted that, for this function to yield a probability, nondeterminism must be eliminated in a way that allows the computation of μ. Here, nondeterminism is treated

https://doi.org/10.1017/S096012950600524X Published online by Cambridge University Press

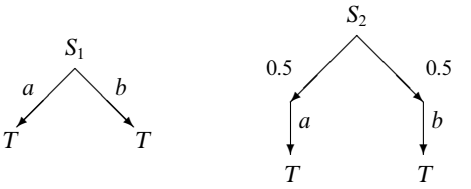
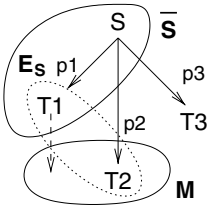


Figure 3. Non equivalent states

as equiprobability, but this is just a convention for the definition of $\mu_{\mathscr{R}}$. For example, this does not imply the equivalence of the two process states S_1 and S_2 described in Figure 3.

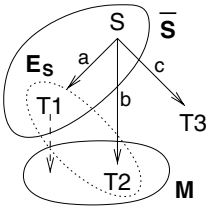
The function $\mu_{\mathscr{R}}$ is defined as follows:

— If $S \in M$, then $\mu_{\mathscr{R}}(S, M) = 1$,



— else, if $\exists\ T \in M \cup \bar{S}$ such that $S \longrightarrow_p T \triangleright M$, then let $E_S = \{R \in M \cup \bar{S} \mid S \longrightarrow_{p_R} R \triangleright M\}$ in

$$\mu_{\mathscr{R}}(S, M) = \sum_{R \in E_S} p_R \ \mu_{\mathscr{R}}(R, M)$$



— else, if $\exists\ T \in M \cup \bar{S}$ such that $S \xrightarrow{a} T \triangleright M$, then let $E_S = \{R \in M \cup \bar{S} \mid S \xrightarrow{a_R} R \triangleright M\}$ in

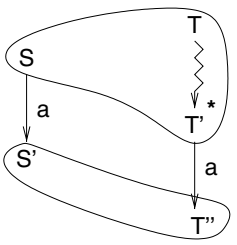
$$\mu_{\mathscr{R}}(S, M) = \frac{1}{|E_S|} \sum_{R \in E_S} \mu_{\mathscr{R}}(R, M)$$

— else $\mu_{\mathscr{R}}(S, M) = 0$.

3.2. *Probabilistic branching bisimulation*

To define bisimilarity, the first step is the definition of a relation of bisimulation on process states. Unlike the case with the usual definitions of bisimulation, here, a bisimulation must be an equivalence relation, because of the last point of the definition, which ensures that the probability to reach an equivalence class is constant on each equivalence class.

Definition An equivalence relation is a probabilistic branching bisimulation if and only if:



— If S and T are equivalent and if an action a can be performed from S, then the same action can be performed from T, possibly after several silent transitions.
— The reached states (S' and T'') are equivalent.
— The action a must occur with the same probability in both cases. The probability to perform an action is the sum for all the branches leading to the action of the product of the probabilities of each branch. It is calculated using the function μ defined in the previous section.

In the following, x and y denote variables and v denotes a classical value.

https://doi.org/10.1017/S096012950600524X Published online by Cambridge University Press

Let $\mathscr{B}$ be an equivalence relation on process states. $\mathscr{B}$ is a probabilistic branching bisimulation if and only if it satisfies:

— **Value sending**
If $S\mathscr{B}T$ and $S \xrightarrow{g\ !v} S'$, then $\exists\ T', T''$ such that

$$(T \rightsquigarrow^* T' \xrightarrow{g\ !v} T'' \ \wedge\ S\mathscr{B}T' \ \wedge\ S'\mathscr{B}T'').$$

— **Qubit sending**
If $S\mathscr{B}T$ and $S \xrightarrow{g\ !x} S'$, then $\exists\ T', T''$ such that

$$(T \rightsquigarrow^* T' \xrightarrow{g\ !y} T'' \ \wedge\ \rho_x^S = \rho_y^{T'} \ \wedge\ S\mathscr{B}T' \ \wedge\ S'\mathscr{B}T'').$$

— **Value reception**
If $S\mathscr{B}T$ and $S \xrightarrow{g\ ?v} S'$, then $\exists\ T', T''$ such that

$$(T \rightsquigarrow^* T' \xrightarrow{g\ ?v} T'' \ \wedge\ S\mathscr{B}T' \ \wedge\ S'\mathscr{B}T'').$$

— **Qubit reception**
If $S\mathscr{B}T$ and $S \xrightarrow{g\ ?x} S'$, then $\exists\ T', T''$ such that

$$(T \rightsquigarrow^* T' \xrightarrow{g\ ?y} T'' \ \wedge\ \rho_x^{S'} = \rho_y^{T''} \ \wedge\ S\mathscr{B}T' \ \wedge\ S'\mathscr{B}T'').$$

— **Silent transition**
If $S\mathscr{B}T$ and $S \rightsquigarrow S'$, then $\exists\ T', T''$ such that

$$(T \rightsquigarrow^* T' \rightsquigarrow^{0..1} T'' \ \wedge\ S\mathscr{B}T' \ \wedge\ S'\mathscr{B}T'').$$

— **Probabilities**
If $S\mathscr{B}T$, then $\mu_{\mathscr{B}}(S, M) = \mu_{\mathscr{B}}(T, M),\ \forall M \in \mathcal{S}/\mathscr{B}$.

Bisimulation and recursion Because of recursion in a process definition, the computation of $\mu_{\mathscr{B}}$ in the associated process graph can lead to a linear system of equations. As a consequence, it must be proved that in this case, $\mu_{\mathscr{B}}$ is well defined and that the system has a unique solution.

Let S be a process state and M be a set of process states, the computation of $\mu_{\mathscr{B}}(T, M)$ for all T in $\bar{S}$ leads to a linear system of equations where the $\mu_{\mathscr{B}}(T, M)$ are the unknowns:

$$X = AX + B\,.$$

The i^{th} row in this system can be written $x_i = a_{i1}x_1 + \cdots + a_{in}x_n + b_i$.

The coefficients a_{ij} and b_i are either probabilities or average coefficients in the case of nondeterminism. As a consequence, $0 \leqslant a_{ij} \leqslant 1$ and $0 \leqslant b_i \leqslant 1$. Moreover, in the definition of $\mu_{\mathscr{B}}$, every state in the set E_S is such that there exists a path from that state to the set M. Therefore, the system of equations obtained can be transformed into a system such that $b_i > 0$, $\forall i \in [\![1, n]\!]$. From now on, we will assume that the system verifies this property.

Another property of the system is $0 \leqslant \sum_j a_{ij} + b_i \leqslant 1$, and thus $0 \leqslant \sum_j a_{ij} < 1$.

https://doi.org/10.1017/S096012950600524X Published online by Cambridge University Press

To prove that the system has a unique solution, it is sufficient to prove $\|A\| < 1$ and use the fixpoint theorem. The norms for matrices and vectors are

$$\|A\| = \sup_{\|X\|=1} \|AX\| \qquad \|X\| = \max_i |x_i| .$$

We obtain

$$\|AX\| = \max_i |\sum_j a_{ij} x_i| \leqslant \max_i \sum_j (|a_{ij}||x_i|)$$

and then

$$\|A\| \leqslant \max_i \sum_j |a_{ij}| < 1 .$$

$\|A\| < 1$ implies that the function $f : X \mapsto AX + B$ is strictly contracting, so from the fixpoint theorem, we infer that the equation $X = AX + B$ has a unique solution. Moreover, as $f([0,1]) \subseteq [0,1]$, this solution belongs to $[0,1]$.

As a consequence, the function $\mu_{\mathscr{B}}$ is well defined even in the case of recursive processes.

3.3. *Probabilistic branching bisimilarity*

Definition 3. Two process states S and T are **bisimilar**, denoted $S \leftrightarroweq T$, if and only if there exists a probabilistic branching bisimulation $\mathscr{B}$ such that $S \mathscr{B} T$.

Proposition 1. Probabilistic branching bisimilarity is an equivalence relation.

Proof.

Symmetry. $S \leftrightarroweq T$ implies that there exists a bisimulation $\mathscr{B}$ such that $S \mathscr{B} T$. Since, $\mathscr{B}$ is an equivalence relation, we have $T \mathscr{B} S$, and thus $T \leftrightarroweq S$.

Reflexivity. The identity relation $\mathscr{I}$ is an equivalence relation that verifies all the points of the definition of the probabilistic branching bisimulation. $\forall S \in \mathbb{S}$, $S \mathscr{I} S$, so $S \leftrightarroweq S$.

Transitivity. Given $S \leftrightarroweq T$ and $T \leftrightarroweq V$, does $S \leftrightarroweq V$? To prove this, we have to find a bisimulation $\mathscr{B}$ such that $S \mathscr{B} T$.

Since $S \leftrightarroweq T$ and $T \leftrightarroweq V$, there exist two bisimulations $\mathscr{B}_1$ and $\mathscr{B}_2$ such that $S \mathscr{B}_1 V$ and $V \mathscr{B}_2 T$ (denoted $S \mathscr{B}_1 V \mathscr{B}_2 T$).

Let Eq give the equivalence closure of a relation and $\circ$ compose two relations, so $S(\mathscr{B}_1 \circ \mathscr{B}_2) T$ if and only if there exists V such that $S \mathscr{B}_1 V \mathscr{B}_2 T$. Now we have $\mathscr{B} = \mathrm{Eq}(\mathscr{B}_1 \circ \mathscr{B}_2)$ is an equivalence relation such that $S \mathscr{B} T$, and we need to prove that it is a bisimulation.

$$S \mathscr{B} T \Leftrightarrow \left| \begin{array}{l} \text{either } S = T \text{ (reflexive closure)} \\ \text{either } \exists V \mid S \mathscr{B}_1 V \mathscr{B}_2 T \\ \text{either } \exists V \mid S \mathscr{B}_2 V \mathscr{B}_1 T \text{ (symmetric closure)} \\ \text{either } \exists V \mid S \mathscr{B} V \mathscr{B} T \text{ with } S(\mathscr{B}_1 \circ \mathscr{B}_2) V \text{ or } S(\mathscr{B}_2 \circ \mathscr{B}_1) V \\ \qquad \text{(transitive closure)} \end{array} \right.$$

We will only develop here the points of the definition of a bisimulation concerning value sending and probabilities, the other points are similar to these.

https://doi.org/10.1017/S096012950600524X Published online by Cambridge University Press

Value sending. $S\mathscr{B}T$ and $S \xrightarrow{a} S'$ ($a = g\ !v$). We will prove $\exists\ T', T''$ such that ($T \rightsquigarrow^* T' \xrightarrow{a} T'' \ \wedge\ S\mathscr{B}T' \ \wedge\ S'\mathscr{B}T''$).

Case $S = T$: Then $T' = S$ and $T'' = S'$.

Case $\exists V \mid S\mathscr{B}_1 V \mathscr{B}_2 T$: Since $S\mathscr{B}_1 V$, there exist V', V'' such that $V \rightsquigarrow^* V' \xrightarrow{a} V''$ and $S\mathscr{B}_1 V'$ and $S'\mathscr{B}_1 V''$.

If $V = V'$, then $V\mathscr{B}_2 T$ implies there exist T', T'' such that ($T \rightsquigarrow^* T' \xrightarrow{a} T'' \ \wedge\ V\mathscr{B}_2 T' \ \wedge\ V''\mathscr{B}_2 T''$).
$S\mathscr{B}_1 V$ and $V\mathscr{B}_2 T'$, so $S\mathscr{B}T'$. But $S'\mathscr{B}_1 V''$ and $V''\mathscr{B}_2 T''$, so $S'\mathscr{B}T''$.

Otherwise, $V \rightsquigarrow V_1 \rightsquigarrow \cdots \rightsquigarrow V_n \rightsquigarrow V'' \xrightarrow{a} V'$. By applying the rule for silent transitions in the definition of a bisimulation to the relation $\mathscr{B}_2$ with the successive V_i, we obtain $\exists\ T', T''$ such that ($T \rightsquigarrow^* T' \xrightarrow{a} T'' \wedge S\mathscr{B}T' \wedge S'\mathscr{B}T''$).

Case $\exists V \mid S\mathscr{B}_2 V \mathscr{B}_1 T$: This is the same as the previous case.

Case $\exists V \mid S\mathscr{B}V\mathscr{B}T$ with $S(\mathscr{B}_1 \circ \mathscr{B}_2)V$ or $S(\mathscr{B}_2 \circ \mathscr{B}_1)V$: We use induction on the sequence of relations.

Probabilities. Let $\{C_i\}_{i\in I}$, $\{D_j\}_{j\in J}$ and $\{M_k\}_{k\in K}$ be the sets of equivalence classes of $\mathscr{B}_1$, $\mathscr{B}_2$ and $\mathscr{B}$, respectively.
For all k, l in K, we prove that for all $S, T \in M_k$, $\mu_{\mathscr{B}}(S, M_l) = \mu_{\mathscr{B}}(T, M_l)$.
First we need to know the relations between the M_k's and the C_i's and D_j's. If $S, T \in C_i$, then $S \in M_k$ implies $T \in M_k$, because $S\mathscr{B}_1 T \mathscr{B}_2 T$. As a consequence, $\forall i \in I,\ \forall k \in K,\ C_i \subseteq M_k$ or $C_i \cap M_k = \varnothing$. Since the equivalence classes of an equivalence relation form a partition of the space considered, $M_k = \bigcup_{i\in I_k} C_i$. Similarly, for $\mathscr{B}_2$ we have $M_k = \bigcup_{j\in J_k} D_j$. Moreover, for all S, T in M_k, there exists a path $V_1, \ldots, V_p$ such that $S\mathscr{B}_1 V_1 \mathscr{B}_2 V_2 \mathscr{B}_1 \ldots \mathscr{B}_2 V_p \mathscr{B}_1 T$.

$$M_k = \bigcup_{i\in I_k} C_i = \bigcup_{j\in J_k} D_j \qquad M_l = \bigcup_{i\in I_l} C_i = \bigcup_{j\in J_l} D_j$$

Now we compute $\mu_{\mathscr{B}}$ as a function of $\mu_{\mathscr{B}_1}$ (*cf.* the example at the end of this section). Let $S \in M_k$. There exists n such that $S \in C_n$. Now, $\mu_{\mathscr{B}}(S, M_l)$ is the probability to reach M_l from S without leaving $M_k \cup M_l$, in other words, it is the sum of the probabilities of reaching each C_j for $j \in I_l$ without leaving $\bigcup_{i\in I_k \cup I_l} C_i$. Since $\mathscr{B}_1$ is a bisimulation, the probability of reaching C_j from C_i is constant for all states in C_i, and this probability will be denoted $\mu_{\mathscr{B}_1}(C_i, C_j)$.
We have

$$\mu_{\mathscr{B}}(S, M_l) = \sum_{j\in I_l} \mu_{\mathscr{B}}(S, C_j) = \sum_{j\in I_l} \tilde{\mu}_{\mathscr{B}}(C_n, C_j)$$

with $\tilde{\mu}_{\mathscr{B}}$ defined by

$$\begin{aligned} \tilde{\mu}_{\mathscr{B}}(C_i, C_i) &= 1 \\ \tilde{\mu}_{\mathscr{B}}(C_i, C_j) &= \textstyle\sum_{m\in E_j} \mu_{\mathscr{B}_1}(C_i, C_m)\tilde{\mu}_{\mathscr{B}}(C_m, C_j) \end{aligned}$$

where $E_j = \{m \mid m \in I_k, \text{ and } C_m \triangleright C_j\}$.
This shows that the function $S \mapsto \mu_{\mathscr{B}}(S, M_l)$ is constant on C_i for all $i \in I_k$. Similarly, this function is also constant on D_j for all j in J_k.

https://doi.org/10.1017/S096012950600524X Published online by Cambridge University Press

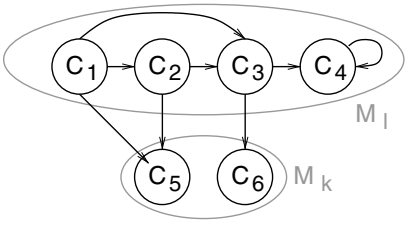


Figure 4. Example of equivalence classes

With these properties:

— If $S, T \in C_n$, then $\mu_{\mathscr{B}}(S, M_l) = \mu_{\mathscr{B}}(T, M_l)$.

— If $S, T \in D_m$, then $\mu_{\mathscr{B}}(S, M_l) = \mu_{\mathscr{B}}(T, M_l)$.

— Otherwise, there exists $\{V_1, \ldots, V_p\}$ such that $\forall i \in [\![1, p]\!], V_i \subseteq M_k$, and $S\mathscr{B}_1 V_1 \mathscr{B}_2 V_2 \mathscr{B}_1 \ldots \mathscr{B}_2 V_p \mathscr{B}_1 T$,

$$\begin{aligned} &\mu_{\mathscr{B}}(S, M_l) = \mu_{\mathscr{B}}(V_1, M_l), \\ &\mu_{\mathscr{B}}(V_1, M_l) = \mu_{\mathscr{B}}(V_2, M_l), \\ &\ldots \\ &\mu_{\mathscr{B}}(V_p, M_l) = \mu_{\mathscr{B}}(T, M_l). \end{aligned}$$

As a consequence, $\mu_{\mathscr{B}}(S, M_l) = \mu_{\mathscr{B}}(T, M_l)$. □

Example of computation of $\mu_{\mathscr{R}}$ as a function of $\mu_{\mathscr{R}_1}$ Figure 4 presents the equivalence classes on a process graph of the two relations $\mathscr{R}$, whose equivalence classes are M_l and M_k and $\mathscr{R}_1$ with the C_i's as equivalence classes. The arrows represent transitions of the process graph.

Let $S \in C_1$. Then,

$$\mu_{\mathscr{R}}(S, M_l) = \mu_{\mathscr{R}}(S, C_4) + \mu_{\mathscr{R}}(S, C_5)$$

$$\begin{aligned} \mu_{\mathscr{R}}(S, C_4) &= \tilde{\mu}_{\mathscr{R}}(C_1, C_4) \\ &= \mu_{\mathscr{R}_1}(C_1, C_4) + \mu_{\mathscr{R}_1}(C_1, C_2)\tilde{\mu}_{\mathscr{R}}(C_2, C_4) \\ &= \mu_{\mathscr{R}_1}(C_1, C_4) + \mu_{\mathscr{R}_1}(C_1, C_2)\mu_{\mathscr{R}_1}(C_2, C_4) \end{aligned}$$

$$\mu_{\mathscr{R}}(S, C_5) = \mu_{\mathscr{R}_1}(C_1, C_2)\mu_{\mathscr{R}_1}(C_2, C_5).$$

We then get

$$\mu_{\mathscr{R}}(S, M_l) = \mu_{\mathscr{R}_1}(C_1, C_4) + \mu_{\mathscr{R}_1}(C_1, C_2)(\mu_{\mathscr{R}_1}(C_2, C_4) + \mu_{\mathscr{R}_1}(C_2, C_5)).$$

4. Example: 'H ⇔ its measurement-based simulation?'

Quantum measurement is universal for quantum computation (Nielsen 2001). This means that every unitary transformation can be simulated using measurements only. We are interesting in proving with QPAlg that a unitary transformation and its measurement-based simulation behave in the same way. We consider the case of the Hadamard transformation H.

https://doi.org/10.1017/S096012950600524X Published online by Cambridge University Press

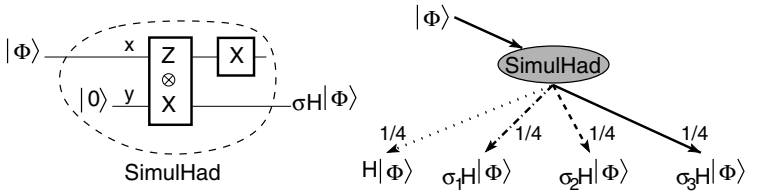


Figure 5. A step in the simulation of H

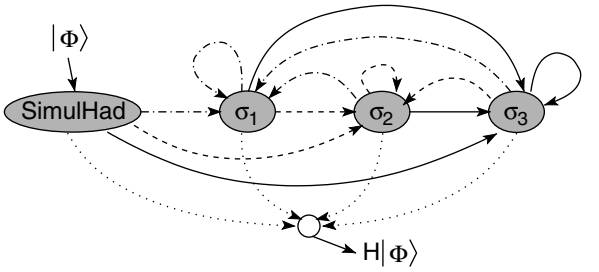


Figure 6. Full simulation of H

4.1. *Measurement-based simulation of H*

There exist several models of quantum computation via measurements only. We use the model based on state transfer defined by Perdrix in Perdrix (2004).

The gate network in Figure 5 describes a step in the simulation of H. This simulation needs an auxiliary qubit initialised in the state $|0\rangle$, which consists of two measurements: one on two qubits with observable $Z \otimes X$ and the other on one qubit with observable X. X and Z are Pauli observables. This step simulates H up to a Pauli operator σ. If $\sigma = I$, H has been simulated, otherwise, a correction must be applied to the result. This correction consists of simulating σ (Pauli operators are their proper inverses) in the same way as H has been simulated. The full simulation of H is given by the automaton in Figure 6. The states **SimulHad**, $\boldsymbol{\sigma_1}$, $\boldsymbol{\sigma_2}$ and $\boldsymbol{\sigma_3}$ represent a step in the simulation of H and of the Pauli operators.

4.2. *Modelling with QPAlg*

Unitary transformation of Hadamard. The process **Had** applies the Hadamard transformation on a qubit received from gate g and returns this qubit through gate h:

$$\mathbf{Had} \stackrel{\text{def}}{=} [\ x : \texttt{Qubit}\ .\ g\ ?x\ .H[x].h\ !x\ .nil\ ].$$

Simulation of Hadamard As with the process **Had**, the process **SimulHad** (Program 2) begins by receiving a qubit x from gate g. Then the Hadamard transformation is simulated in the way described by the automaton shown in Figure 6. At the end, the result is in y (and not x) that is sent through gate h. $\boldsymbol{\sigma}$ is a process simulating a Pauli transformation, the Pauli transformation simulated is specified by the parameter i in $\sigma[i, y]$. The process $\boldsymbol{\sigma}$ is not given, it is organised in a way similar to **SimulHad**.

https://doi.org/10.1017/S096012950600524X Published online by Cambridge University Press

Program 2 Process for the simulation of H

SimulHad $\stackrel{\text{def}}{=}$
[y : Qubit, i : Nat .
  [x : Qubit, m, n : Nat .
    $g\ ?x$. // initialisation of x
    $(p\ ?y\ .end \parallel p\ !0\ .end)\backslash\{p\}$; // initialisation of y to $|0\rangle$
    $(q_1\ !Z \otimes X\ [x, y].q_2\ !X[x]\ .end \parallel q_1\ ?m\ .q_2\ ?n\ .end)\backslash\{q_1, q_2\}$;
    (**[**
      $(m = 1 \wedge n = 1) \rightarrow r\ !0\ .end$,
      $(m = 1 \wedge n = -1) \rightarrow r\ !1\ .end$, // initialisation of i
      $(m = -1 \wedge n = 1) \rightarrow r\ !3\ .end$, // depending on
      $(m = -1 \wedge n = -1) \rightarrow r\ !2\ .end$ // measurement results
    ] $\parallel r\ ?i\ .end)\backslash\{r\}$
  ] ;$\boldsymbol{\sigma}[i, y]$; // correction
  $h\ !y\ .nil$ // result sending
]

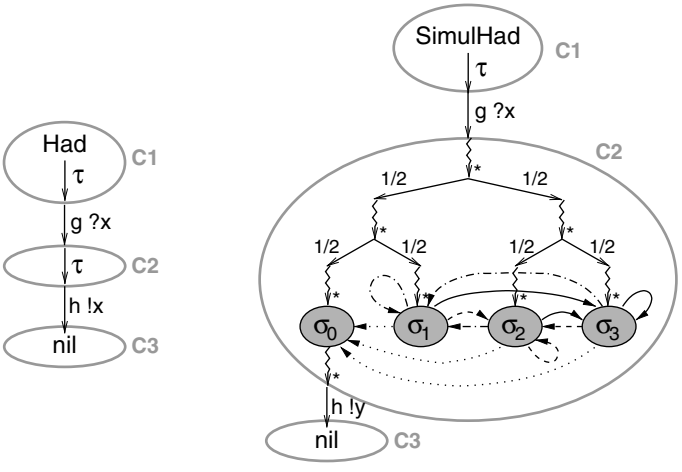


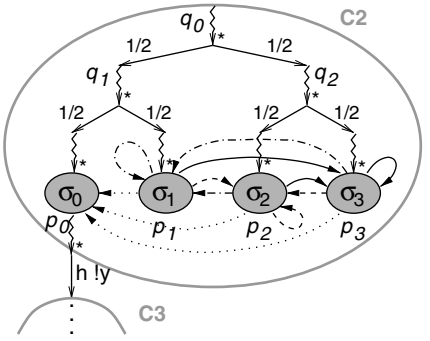
Figure 7. Process graphs of **Had** and **SimulHad** in an empty context

The operational semantics associates with the processes **Had** and **SimulHad** (Program 2) in empty contexts the process graphs described in Figure 7.

4.3. *Bisimilarity*

To prove that the application of the Hadamard transformation and its measurement-based simulation have the same behaviour, we prove that their modellings in QPAlg are bisimilar, that is, that there exists a bisimulation between **Had** and **SimulHad** in empty contexts.

https://doi.org/10.1017/S096012950600524X Published online by Cambridge University Press



$$p_0 = 1$$
$$\forall i \in \{1,2,3\}\ p_i = \tfrac{1}{4}p_0 + \tfrac{1}{4}p_1 + \tfrac{1}{4}p_2 + \tfrac{1}{4}p_3$$
$$q_1 = \tfrac{1}{2}p_0 + \tfrac{1}{2}p_1$$
$$q_2 = \tfrac{1}{2}p_2 + \tfrac{1}{2}p_3$$
$$q_0 = \tfrac{1}{2}q_1 + \tfrac{1}{2}q_2$$

Figure 8. Computation of the probabilities.

The process graphs of **Had** and **SimulHad** in empty contexts are given in Figure 7. C_1, C_2, C_3 and C_4 represents equivalence classes of an equivalence relation $\mathscr{R}$ on process states. This relation is a bisimulation, the main points of the proof are:

— From each process state in C_1, the transition $g\ ?x$ can be performed, possibly after several silent transitions.

— From each process state in C_2, the transition $h\ !x$ or $h\ !y$ can be performed, possibly after several silent transitions.

— When $h\ !x$ and $h\ !y$ are performed, the states of x and y are the same.

— The probability of reaching C_3 from each state in C_2 must be the same, which, for any state S in C_2, corresponds to $\mu_{\mathscr{R}}(S, C_3) = 1$.

For the last point, the computation of the function $\mu_{\mathscr{R}}$ on each state in C_2 leads to the system of equations in Figure 8. We obtain $\forall i \in \{0, \ldots, 3\}$, $p_i = 1$, and then $q_0 = q_1 = q_2 = 1$. From any state in the class C_2, the probability of reaching a state in the class C_3 is 1.

To conclude, there exists a bisimulation relation between the modelling of the application of Hadamard and the modelling of its measurement-based simulation.

5. Is $\leftrightarrows$ a congruence?

It is natural to ask whether the bisimilarity defined in this section is a congruence. In fact, this question does not make much sense since this relation is defined on process states (process and context), and there are no composition operators on process states.

There are two possibilities to make it a congruence, either the operators on processes are extended to operators on process states, or a congruence on processes is defined from the bisimilarity relation on process states.

We explore the second possibility. We define a relation $\sim$ on processes as follows: if P and Q are two processes, then $P \sim Q \Leftrightarrow \forall C, P/C \leftrightarrows Q/C$.

https://doi.org/10.1017/S096012950600524X Published online by Cambridge University Press

Nonetheless, $\sim$ is not a congruence, as shown by the following two examples:

$$H[x].g\ !x\ .nil \sim g\ !x\ .nil$$
$$\text{but } H[x].g\ !x\ .nil + h\ !2\ .nil \not\sim g\ !x\ .nil + h\ !2\ .nil$$

$$H[x].g\ !x\ .nil \sim g\ !x\ .nil \oplus_{0.3} g\ !x\ .nil$$
$$\text{but } H[x].g\ !x\ .nil + h\ !2\ .nil \not\sim (g\ !x\ .nil \oplus_{0.3} g\ !x\ .nil) + h\ !2\ .nil\,.$$

To overcome these problems, we define probabilistic rooted branching bisimulation and probabilistic rooted branching bisimilarity.

Definition 4. Let $\mathscr{B}$ be an equivalence relation. $\mathscr{B}$ is a **probabilistic rooted branching bisimulation** if and only if:

— If $S\mathscr{B}T$ and $S \xrightarrow{a} S'$ ($a = g\ !v$ or $g\ ?v$ or τ), then $T \xrightarrow{a} T'$ with $S' \leftrightarrows T'$.
— If $S\mathscr{B}T$ and $S \xrightarrow{g\ !x} S'$, then $T \xrightarrow{g\ !y} T'$ with $\rho_x^S = \rho_y^T$ and $S' \leftrightarrows T'$.
— If $S\mathscr{B}T$ and $S \xrightarrow{g\ ?x} S'$, then $T \xrightarrow{g\ ?y} T'$ with $\rho_x^{S'} = \rho_y^{T'}$ and $S' \leftrightarrows T'$.
— If $S\mathscr{B}T$ and $S \longrightarrow_p S'$, then $T \longrightarrow_q T'$ with $S'\mathscr{B}T'$.
— If $S\mathscr{B}T$ then $\mu_{\leftrightarrows}(S, M) = \mu_{\leftrightarrows}(T, M)$, $\forall M \in \mathcal{S}/\leftrightarrows$.

Definition 5. Two process states S and T are **probabilistic rooted branching bisimilar**, denoted $S \leftrightarrows_r T$, if and only if there is a probabilistic branching bisimulation $\mathscr{B}$ such that $S\mathscr{B}T$.

Proposition 2. $\leftrightarrows_r$ is an equivalence relation.

Proof. The proof follows from the fact that $\leftrightarrows$ is an equivalence relation. □

Note. For all S, T in $\mathcal{S}$, we have $S \leftrightarrows_r T$ implies $S \leftrightarrows T$.

Definition 6. If P and Q are two processes,

$$P \sim Q \Leftrightarrow \forall C, P/C \leftrightarrows_r Q/C\,.$$

Proposition 3. $\sim$ is an equivalence relation and is preserved by variable declaration, action prefix, non-deterministic choice, probabilistic choice, conditional choice and restriction.

Proof. Since $\leftrightarrows_r$ is an equivalence relation, it is easy to see that $\sim$ is also an equivalence relation. Now we prove that $\sim$ is preserved by all operators on processes. Let P, Q be processes such that $P \sim Q$, that is to say, $\forall C, P/C \leftrightarrows_r Q/C$.

Action prefix. Let a be an action. We need to know whether $a.P/C$ and $a.Q/C$ are probabilistically rooted branching bisimilar for all contexts C.
If C is probabilistically stable, $a.P/C \xrightarrow{a} P/C'$ and $a.Q/C \xrightarrow{a} Q/C'$. So $P/C' \leftrightarrows_r Q/C'$ implies $P/C' \leftrightarrows Q/C'$, and $\mathscr{B} = \{(a.P/C, a.Q/C)\}$ is a probabilistic rooted branching bisimulation.
If $C = \boxplus_{p_i} C_i$, then for all i, $a.P/C \longrightarrow_{p_i} a.P/C_i$ and $a.Q/C \longrightarrow_{p_i} a.Q/C_i$. From the previous case, $\forall\ i$: $a.P/C_i \leftrightarrows_r a.Q/C_i$, and we deduce $a.P/C \leftrightarrows_r a.Q/C$.

https://doi.org/10.1017/S096012950600524X Published online by Cambridge University Press

Nondeterminism. Let R be a process. We need to know whether $P + R/C$ and $Q + R/C$ are probabilistic rooted branching bisimilar for all contexts C.
First consider the case in which C is probabilistically stable. If $P + R/C \to P'/C_P$, then, as $P/C \leftrightarrows_r Q/C$, there exists Q'/C_Q such that $Q + R/C \to Q'/C_Q$ and $P'/C_p \leftrightarrows Q'/C_Q$. Otherwise, $P + R/C \to R'/C''$, and we have $Q + R/C \to R'/C''$ and $R'/C'' \leftrightarrows R'/C''$.
If C is not probabilistically stable, we reduce the problem to the previous point after a probabilistic transition.

The other points are similar to those developed. □

We conclude that $\sim$ is preserved by all operators except $\|$, as shown by the following example: $(g\ !2.h\ !3.nil \| g\ ?x.h\ ?x.nil) \setminus \{g,h\} \sim (h\ !3.nil \| h\ \ ?x.nil) \setminus \{h\}$, nonetheless, $(g\ !2.h\ !3.nil \| g\ ?x.h\ ?x.nil) \setminus \{g,h\} \| k\ !x.nil \not\sim (h\ !3.nil \| h\ ?x.nil) \setminus \{h\} \| k\ !x.nil$. The left process can send 2 or 3 through gate k, while the right process can only send 3.

This problem could be overcome by restricting processes in parallel not to use the same variable names. This is done in CQP (Nagarajan and Gay 2004) and can be justified by the fact that variables cannot be at two places at the same time. However, because of entanglement in the quantum state, this does not solve the whole problem:

$$M_{std,1}[x].nil \ \sim\ I[x].(0.nil \bigoplus_{0.5} 1.nil)$$
$$M_{std,1}[x].nil \| g\ !y.nil \ \not\sim\ (I[x].(0.nil \bigoplus_{0.5} 1.nil)) \| g\ !y.nil.$$

In a configuration where the state of x and y is $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ (EPR state), the left process can send through gate g the qubit y in the mixed state $\{0.5 : |0\rangle, 0.5 : |1\rangle\}$ if $M_{std,1}[x]$ has not been applied, or in the states $|0\rangle$ or $|1\rangle$ if the measurement has been performed, while the right process can only send y in the state $\{0.5 : |0\rangle, 0.5 : |1\rangle\}$.

6. Conclusion

This paper has presented a process algebra for quantum programming that can describe both classical and quantum programming, and their cooperation. This language has an operational semantics, one of its peculiarities being the introduction of probabilistic transitions, due to quantum measurement and to the operator of probabilistic choice.

Then we defined a semantical equivalence relation on process states. This equivalence is a bisimulation that identifies processes associated with process graphs having the same branching structure. From this bisimulation, an equivalence relation on processes has been defined. This relation is preserved by all the operators of the process algebra except parallel composition. This is the first step toward the verification of quantum cryptographic protocols.

Several extensions are possible. As already mentioned, we could define a congruence on process states from the bisimulation defined here by extending the operators on processes to operators on process states. Another possible extension is the definition of a type system to give a static verification of such properties as the no-cloning theorem (quantum variables cannot be copied).

https://doi.org/10.1017/S096012950600524X Published online by Cambridge University Press

Appendix A. The quantum process algebra

A.1. *Syntax*

Expressions

qexp ::= *qvar* ⊗ *qexp* | *qvar*

nexp ::= *nfact* + *nexp* | *nfact* − *nexp* | *nfact*
nfact ::= *nterm* × *nfact* | *nterm* ÷ *nfact* | *nterm*
nterm ::= *nvar* | *nval* | *transf_admissible* **[** *qexp* **]**
| **(** nexp **)** | −nterm

Actions

communication ::= *gate* **!** *variable* | *gate* **!** *nexp*
| *gate* **?** *variable*
action ::= *communication* | *nexp*

Processes

type ::= **Nat** | **Qubit**
variable_decl ::= *variable* **:** *type* {**,** *variable* **:** *type* }*

process ::= ***nil***
| *name* {**[** *variable_list* **]**}
| *param_decl* • *process* **[** *variable_list* **]**
| **[** *variable_decl* **.** *process* **]**
| *action* **.** *process*
| *process* ‖ *process*
| *process* + *process*
| **[** *condition* → *process* {**,** *condition* → *process* }* **]**
| *process* $\{\oplus_{probability}\ process\}^{+}$
| *process* \{ *gate_list* }

param_decl ::= **[** *variable_decl* **]**

process_decl ::= *name* $\overset{\text{def}}{=}$ *process* | *nom* $\overset{\text{def}}{=}$ *param_decl* • *process*

A.2. *Main inference rules of the semantics*

The semantics is specified with inference rules, which give the evolution of the states of processes. There are five kinds of transitions:

— Transitions for evaluating expressions: $\longrightarrow_v$ and $\longrightarrow_e$
— Action transitions: $\xrightarrow{\alpha}$ where α is $g\ !x$ or $g\ ?x$
— Silent transitions: $\xrightarrow{\tau}$, for internal transitions
— Probabilistic transitions: $\longrightarrow_p$, where p is a probability.

https://doi.org/10.1017/S096012950600524X Published online by Cambridge University Press

In the following, P, Q, P', Q', P_i and P_i' are processes, C, C' and C_i are contexts, α is an action, g is a communication gate, v is a value, x is a variable, and c_j is a condition.

Expressions

$$\frac{}{x/ < s, q = \rho, f > \longrightarrow_v f(x)/ < s, q = \rho, f >} \qquad x \in \mathrm{dom}(f) \tag{1}$$

$$\frac{}{A[x_1 \otimes \ldots \otimes x_n]/C \longrightarrow_v y/ \boxplus_{p_i} C_i} \tag{2}$$

with:

— $A = \{A_{\tau_1}, \ldots, A_{\tau_n}\}$, admissible transformation
— y, fresh variable name
— $C =< e.s, q = \rho, f >$
— $C_i =< (\{y : \mathtt{Nat}\} \cup e).s, q = \rho_i, f \cup \{y \mapsto \tau_i\} >$
— $x_1, \ldots, x_n \in q$
— $p_i = Tr(\mathscr{T}_{A_{\tau_i}}(\rho))$ and $\rho_i = \frac{1}{p_i}\mathscr{T}_{A_{\tau_i}}(\rho)$.

Evaluation contexts $E[\,]$ is an evaluation context of an expression, it is an expression in which a sub-expression has been replaced by $[\,]$. Similarly, $F[\,]$ is an evaluation context of a process.

$$\frac{e/C \longrightarrow_v e'/C'}{E[e]/C \longrightarrow_e E[e']/C'} \tag{3}$$

$$\frac{e/C \longrightarrow_e e'/C'}{F[e]/C \xrightarrow{\tau} F[e']/C'} \tag{4}$$

Action Prefix

Communication

$$\frac{}{g\ !v\ .P/C \xrightarrow{g\ !v} P/C} \qquad v \in \mathbb{N} \tag{5}$$

$$\frac{}{g\ !x\ .P/C \xrightarrow{g\ !x} P/C} \qquad C =< s, q = \rho, f > \text{ and } x \in q \tag{6}$$

For all $v \in \mathbb{N}$:

$$\frac{}{g\ ?x\ .P/C \xrightarrow{g\ ?v} P/C'} \tag{7}$$

with $C =< s, q = \rho, f >$, $C' =< s, q = \rho, f \cup \{x \mapsto v\} >$ and $(x, \mathtt{Nat}) \in s$

For any density matrix v of dimension 2,

$$\frac{}{g\ ?x\ .P/C \xrightarrow{g\ ?x} P/C'} \tag{8}$$

https://doi.org/10.1017/S096012950600524X Published online by Cambridge University Press

with

— $C =< s, q = \rho, f >$, $C' =< s, x.q = v \otimes \rho, f >$
— $(x, \texttt{Qubit}) \in s$ and $x \notin q$

Expression

$$\frac{}{v.P/C \xrightarrow{\tau} P/C} \qquad v \in \mathbb{N} \tag{9}$$

Probabilities

$$\frac{}{P/\boxplus_{p_j} C_j \longrightarrow_{p_i} P/C_i} \qquad \sum_j p_j = 1 \tag{10}$$

$$\frac{}{\bigoplus_{p_j} P_j/C \longrightarrow_{p_i} P_i/C} \qquad \sum_j p_j = 1 \tag{11}$$

Nondeterministic choice

$$\frac{P/C \quad P'/C'}{P + Q/C \quad P'/C'} \qquad P\downarrow, Q\downarrow \tag{12}$$

where represents any transition.

$$\frac{P/C \longrightarrow_p P'/C'}{P + Q/C \longrightarrow_p P' + Q/C'} \qquad Q\downarrow \tag{13}$$

$$\frac{P/C \longrightarrow_p P'/C' \qquad Q/C \longrightarrow_q Q'/C''}{P + Q/C \longrightarrow_{pq} P' + Q'/C} \tag{14}$$

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

— $C =< (s_P \parallel s_Q).s, q = \rho, f >$
— $C_P =< s_P|s, q = \rho, f >$
— $C_Q =< s_Q|s, q = \rho, f >$.

In the definition of C, the operator $\parallel$ permits us to build a cactus stack (see Section 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, while the names in s_P (respectively, s_Q) correspond to variables declared in P (respectively, Q).

$$\frac{P/C_P \xrightarrow{\alpha} P'/C'_P}{P \parallel Q/C \xrightarrow{\alpha} P' \parallel Q/C'} \qquad P\downarrow, Q\downarrow \tag{15}$$

where

— If $C'_P =< s', q' = \rho', f' >$, then $C' =< (s'_P \parallel s_Q).s, q' = \rho', f' >$ with s'_P such that $s' = s'_P|s$ (P can neither add to nor remove variables from s)
— If $C'_P = \boxplus_{p_i} < s'_i, q'_i = \rho'_i, f'_i >$, then $C' = \boxplus_{p_i} < (s_{P'_i} \parallel s_Q).s, q'_i = \rho'_i, f'_i >$ with $s_{P'_i}$ such that $s'_i = s_{P'_i}|s$

https://doi.org/10.1017/S096012950600524X Published online by Cambridge University Press

$$\frac{P/C_P \xrightarrow{g\ !v} P'/C'_P \qquad Q/C_Q \xrightarrow{g\ ?v} Q'/C'_Q}{P \parallel Q/C \xrightarrow{\tau} P' \parallel Q'/C'} \qquad P\downarrow, Q\downarrow \tag{16}$$

where $v \in I\!N$, $C'_Q =< s', q' = \rho', f' >$, and $C' =< (s_P \parallel s_Q).s, q = \rho, f' >$

$$\frac{P/C_P \xrightarrow{g\ !v} P'/C'_P \qquad Q/C_Q \xrightarrow{g\ ?y} Q'/C'_Q}{P \parallel Q/C \xrightarrow{\tau} P' \parallel Q'/C'} \qquad P\downarrow, Q\downarrow \tag{17}$$

where

— $(x, \texttt{Qubit}) \in s \cup s_Q$, $x \notin q$, $v \in \{0, 1\}$
— $C' =< (s_P \parallel s_Q).s, x.q = |v\rangle\langle v| \otimes \rho, f >$

$$\frac{P/C_P \xrightarrow{g\ !x} P'/C'_P \qquad Q/C_Q \xrightarrow{g\ ?y} Q'/C'_Q}{P \parallel Q/C \xrightarrow{\tau} P' \parallel Q'/C'} \qquad P\downarrow, Q\downarrow \tag{18}$$

where

— $(x, \texttt{Qubit}) \in s \cup s_P$, $x \in q$
— $(y, \texttt{Qubit}) \in s \cup s_Q$, $y \notin q$
— $C' =< ((s_P \parallel s_Q).s)\backslash\{x\}, q[x \leftarrow y] = \rho, f >$

$$\frac{P/C_P \longrightarrow_p P'/C'_P}{P \parallel Q/C \longrightarrow_p P' \parallel Q/C} \qquad Q\downarrow \tag{19}$$

$$\frac{P/C_P \longrightarrow_p P'/C'_P \qquad Q/C_Q \longrightarrow_q Q'/C'_Q}{P \parallel Q/C \longrightarrow_{pq} P' \parallel Q'/C}. \tag{20}$$

References

Andova, S. (1999) Process algebra with probabilistic choice. *Springer-Verlag Lecture Notes in Computer Science* **1601** 111–129.

Arrighi, P. and Dowek, G. (2004) Operational semantics for formal tensorial calculus. In: Selinger, P. (ed.) *Proceedings of the 2nd International Workshop on Quantum Programming Languages* 21–38.

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

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

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

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

Fokkink, W. (2000) *Introduction to Process Algebra*, Springer.

Lalire, M. and Jorrand, P. (2004) A process algebraic approach to concurrent and distributed quantum computation: operational semantics. In: Selinger, P. (ed.) *Proceedings of the 2nd International Workshop on Quantum Programming Languages* 109–126.

https://doi.org/10.1017/S096012950600524X Published online by Cambridge University Press

Milner, R. (1989) *Communication and Concurrency*, Prentice-Hall.
Nagarajan, R. and Gay, S. (2004) Communicating quantum processes. In: Selinger, P. (ed.) *Proceedings of the 2nd International Workshop on Quantum Programming Languages* 91–107.
Nielsen, M. (2001) Universal quantum computation using only projective measurement, quantum memory and preparation of the 0 state. Los Alamos e-print arXiv.
Nielsen, M. A. and Chuang, I. L. (2000) *Quantum Computation and Quantum Information*, Cambridge University Press.
Ömer, B. (2000) Quantum programming in QCL. Master's thesis, Institute of Information Systems, Technical University of Vienna.
Perdrix, S. (2004) State transfer instead of teleportation in measurement-based quantum computation. Los Alamos e-print arXiv.
Roscoe, A. W. (1998) *The theory and practice of concurrency*, Prentice-Hall.
Selinger, P. (2004) Toward a quantum programming language. *Mathematical Structures in Computer Science* **14** (4) 525–586.
van Glabbeek, R. (1993) The linear time – branching time spectrum ii: the semantics of sequential systems with silent moves. In: Best, E. (ed.) Proceedings of the 4th Conference on Concurrency Theory (CONCUR'93), Hildesheim. *Springer-Verlag Lecture Notes in Computer Science* **715**. 66–81.
Van Tonder, A. (2003) A lambda calculus for quantum computation. Los Alamos e-print arXiv.
Zuliani, P. (2001) *Quantum Programming*, Ph.D. thesis, St Cross College, University of Oxford.

https://doi.org/10.1017/S096012950600524X Published online by Cambridge University Press