Proc. QPL 2004, pp. 91–107

Communicating quantum processes

Simon J. Gay* Rajagopal Nagarajan†‡

Abstract

We define a language CQP (Communicating Quantum Processes) for modelling systems which combine quantum and classical communication and computation. CQP combines the communication primitives of the pi-calculus with primitives for measurement and transformation of quantum state; in particular, qubits can be transmitted from process to process along communication channels. CQP has a static type system which classifies channels, distinguishes between quantum and classical data, and controls the use of quantum state. We formally define the syntax, operational semantics and type system of CQP, prove that the semantics preserves typing, and prove that typing guarantees that each qubit is owned by a unique process within a system.

1 Introduction

Quantum cryptography is rapidly becoming a practical technology: secure communication based on quantum principles has recently been demonstrated in a scenario involving banking transactions in Vienna [15], systems are commercially available from IdQuantique, MagiQ Technologies and NEC, and plans have been reported to establish a nationwide quantum communication network in Singapore. Communication systems exploiting quantum cryptography offer security against any physically realizable attack, including those based on possible future developments in quantum computing. Secure quantum communication will undoubtedly become a fundamental part of the technological infrastructure of society.

However, secure quantum communication is not a solved problem. The existence of a perfect cryptographic technique does not in itself guarantee the security of a system which uses it. The protocols for creation, distribution and use of cryptographic keys and encrypted messages must be considered carefully. Even when protocols have been formally proved to be secure, it is notoriously difficult to achieve robust and reliable implementations of secure systems: security can be compromised by flaws at the implementation level or at the boundaries between systems. Computer scientists have developed an impressive armoury of techniques and tools for formal modelling, analysis and verification of classical security protocols and communication systems which use them [16]. These techniques have been remarkably successful both in establishing the security of new protocols

* Department of Computing Science,University of Glasgow,UK; simon@dcs.gla.ac.uk

† Department of Computer Science, University of Warwick,UK; biju@dcs.warwick.ac.uk

‡ Nagarajan is partially supported by EPSRC (GR/S34090) and the EU (Project SecoQC).

and in demonstrating flaws in protocols which had previously been believed to be secure. The state of the art in verification of quantum security systems is limited to mathematical proofs of correctness of particular protocols (for example, Mayers' analysis [11] of the Bennett-Brassard protocol (BB84) [3] for quantum key distribution) and there are not yet techniques or automated tools for the analysis of general systems. Furthermore, practical quantum communication systems will contain classical components: initially at least, communication will take place between classical computers; and protocols such as BB84 typically contain classical communication and computation as well as quantum cryptography. Therefore there is a need for formal verification of systems which combine quantum and classical components and protocols; moreover, formal models must be flexible enough to facilitate re-analysis of variations in design. A modelling language with a precisely-defined semantics is an essential first step towards formal analysis and verification. It is not sufficient to assume the security of quantum cryptography and incorporate it axiomatically into classical security analysis, because we also want to analyze the protocols which *construct* quantum cryptographic keys.

We define a language CQP (Communicating Quantum Processes) for modelling systems which combine classical and quantum communication and computation. CQP combines the communication primitives of the pi-calculus [12, 18] with the quantum information-processing primitives of Selinger's language QPL [19]. CQP is similar in some respects to the quantum process algebra proposed by Jorrand and Lalire [8] but introduces an important extension: quantum state (qubits) can be sent along communication channels and transferred from process to process. CQP has a static type system which classifies communication channels (at the very least, distinguishing between classical and quantum data) and controls the use of quantum state. If process P sends qubit q to process Q, then P must not access q subsequently, and this restriction can be enforced by static type-checking. The ability to send qubits along communication channels is necessary in order to model quantum cryptographic protocols such as BB84 and quantum communication protocols such as dense coding. We define an operational semantics for CQP in terms of non-deterministic transitions from processes to probability distributions over processes, and probabilistic transitions from probability distributions to processes. We prove that the invariants of the static type system (for example, unique ownership of quantum state) are preserved by the semantics.

Related Work

There has been a great deal of interest in quantum programming languages, resulting in a number of proposals in different styles, for example [5, 9, 13, 17, 19, 20]. Such languages can express arbitrary quantum state transformations and could be used to model quantum protocols in those terms. However, our view is that any model lacking an explicit treatment of communication is essentially incomplete for the analysis of protocols; certainly in the classical world, standard programming languages are not considered adequate frameworks in which to analyze or verify protocols. The closest work to our own is the *quantum process algebra* of Jorrand and Lalire [8], which also combines process-calculus-style communication with transformation and measurement of quantum state. Several aspects of our language CQP are similar to their process algebra; the distinctive features of CQP are

$$\begin{aligned}
&\mathit{Alice}(x:\mathsf{Qbit}, c:\hat{}[\mathsf{0..3}], z:\mathsf{Qbit}) = \\
&\qquad \{z, x \mathrel{*}= \mathsf{CNot}\} \,.\, \{z \mathrel{*}= \mathsf{H}\} \,.\, c![\mathsf{measure}\ z, x] \,.\, \mathbf{0} \\[1em]
&\mathit{Bob}(y:\mathsf{Qbit}, c:\hat{}[\mathsf{0..3}]) = c?[r:\mathsf{0..3}] \,.\, \{y \mathrel{*}= \sigma_r\} \,.\, \mathit{Use}(y) \\[1em]
&\mathit{System}(x:\mathsf{Qbit}, y:\mathsf{Qbit}, z:\mathsf{Qbit}) = \\
&\qquad (\mathsf{new}\ c:\hat{}[\mathsf{0..3}])(\mathit{Alice}(x, c, z) \mid \mathit{Bob}(y, c))
\end{aligned}$$

Figure 1: QUANTUM TELEPORTATION IN CQP

that it allows quantum state to be transmitted between processes, which is essential in order to model BB84 and other protocols, and that it has a static type system which classifies data and communication channels and enforces physical invariants such as non-duplication of quantum state. The work of Abramsky and Coecke [1] is also relevant. They define a category-theoretic semantic foundation for quantum protocols which supports reasoning about systems and exposes deep connections between quantum systems and programming language semantics, but they do not define a formal syntax in which to specify models. It will be interesting to investigate the relationship between CQP and the semantic structures which they propose.

2 Informal Examples

Before formally defining the syntax and semantics of CQP, we describe models of two communication protocols which exhibit a mixture of quantum and classical features. The first is quantum teleportation [4], which is a procedure for transmitting a quantum state via a non-quantum medium. This protocol is particularly important: not only is it a fundamental component of several more complex protocols, but it is likely to be a key enabling technology for the development of the *quantum repeaters* [6] which will be necessary in large-scale quantum communication networks. The second example, dense coding [2], is the converse of teleportation; a single quantum bit is used to transmit two classical bits of information.

2.1 Modelling Teleportation in CQP

Figure 1 shows a simple model of the quantum teleportation protocol. Alice and Bob each possess one qubit (x for Alice, y for Bob) of an *entangled pair* whose state is $\frac{1}{\sqrt{2}}(|0\rangle|0\rangle + |1\rangle|1\rangle)$. At this point we are assuming that appropriate qubits will be supplied to Alice and Bob as parameters of the system. Alice is also parameterized by a qubit z, whose state is to be teleported. She applies ($z, x \mathrel{*}= \mathsf{CNot}$) the *conditional not* transformation CNot to z and x and then applies ($z \mathrel{*}= \mathsf{H}$) the *Hadamard* transformation H to z, finally measuring z and x to yield a two-bit classical value which she sends ($c![\mathsf{measure}\ z, x]$) to Bob on the typed channel $c:\hat{}[\mathsf{0..3}]$ and then terminates ($\mathbf{0}$). Bob receives ($c?[r:\mathsf{0..3}]$) this value and

$$\mathit{Alice}'(s{:}\widehat{\ }[\mathsf{Qbit}], c{:}\widehat{\ }[\mathsf{0..3}], z{:}\mathsf{Qbit}) = s?[x{:}\mathsf{Qbit}]\,.\,\mathit{Alice}(x,c,a)$$

$$\mathit{Bob}'(t{:}\widehat{\ }[\mathsf{Qbit}], c{:}\widehat{\ }[\mathsf{0..3}]) = t?[y{:}\mathsf{Qbit}]\,.\,\mathit{Bob}(y,c)$$

$$\begin{aligned}&\mathit{Source}(s{:}\widehat{\ }[\mathsf{Qbit}], t{:}\widehat{\ }[\mathsf{Qbit}]) =\\ &\qquad(\mathsf{new}\ x{:}\mathsf{Qbit}, y{:}\mathsf{Qbit})(\{x \mathrel{*}= \mathsf{H}\}\,.\,\{x, y \mathrel{*}= \mathsf{CNot}\}\,.\,s![x]\,.\,t![y]\,.\,\mathbf{0})\end{aligned}$$

$$\begin{aligned}&\mathit{System}'(z{:}\mathsf{Qbit}) =\\ &\qquad(\mathsf{new}\ c{:}\widehat{\ }[\mathsf{0..3}], s{:}\widehat{\ }[\mathsf{Qbit}], t{:}\widehat{\ }[\mathsf{Qbit}])\\ &\qquad\quad(\mathit{Alice}'(s,c,z) \mid \mathit{Bob}'(t,c) \mid \mathit{Source}(s,t))\end{aligned}$$

Figure 2: Quantum teleportation with an EPR source

uses it to select a *Pauli* transformation $\sigma_0 \ldots \sigma_3$ to apply ($y \mathrel{*}= \sigma_r$) to y. The result is that Bob's qubit y takes on the state of z, without any quantum state having been transmitted from Alice to Bob. Bob may then use y in his continuation process $\mathit{Use}(y)$.

The definition of the complete system shows the use of new for declaration of channels, and parallel composition ($\mathit{Alice}(\ldots) \mid \mathit{Bob}(\ldots)$). Most of the syntax of CQP is based on pi-calculus. Quantum state transformations such as $x, y \mathrel{*}= \mathsf{CNot}$ are described in a syntax similar to that of Selinger's QPL [19], and converted into actions by means of $\{\ldots\}$. Measurements such as $\mathsf{measure}\ z, x$ also use the syntax of QPL and are treated as expressions, executed for their value as well as for side-effects on quantum state. Alice and Bob are parameterized by their parts of the entangled pair and by the channels a and c. This is similar to Jorrand and Lalire's [8] model with added type information, but now we can go further by introducing what is known in the physics literature as an *EPR source* (computer scientists might regard it as an *entanglement server*). This process constructs the entangled pair (by using the Hadamard and controlled not transformations) and sends its components to Alice and Bob on the typed channels $s, t{:}\widehat{\ }[\mathsf{Qbit}]$ (this syntax is found in several presentations of typed pi-calculus, for example [14]). Figure 2 shows the revised model.

It is desirable to ensure that each qubit in the system is owned by a unique process, even though qubits may be transmitted along channels. This corresponds to the reality that a qubit is a physical object and cannot simply be duplicated in the manner of a classical data value. We guarantee this by introducing techniques of linear typing [7], which have already been applied to the pi-calculus [10], into our type system. If Source had a non-trivial continuation C, instead of $\mathbf{0}$, then the qubits x and y would be removed from the type environment before checking C. This point will be discussed further in Section 4.

2.2 Modelling Dense Coding in CQP

The dense coding protocol achieves transmission of two classical bits by transmitting a single qubit, thus increasing the information-carrying capacity of a channel. Our model is shown in Figure 3.

$$
\begin{array}{l}
\mathit{Alice}(x\,{:}\,\mathsf{Qbit}, q\,{:}\,\hat{}[\mathsf{Qbit}], n\,{:}\,\mathsf{0..3}) = \{x \mathrel{*}= \sigma_n\}\,.\,q![x]\,.\,\mathbf{0}\\[1em]
\mathit{Bob}(y\,{:}\,\mathsf{Qbit}, q\,{:}\,\hat{}[\mathsf{Qbit}]) =\\
\qquad\quad q?[x\,{:}\,\mathsf{Qbit}]\,.\,\{x, y \mathrel{*}= \mathsf{CNot}\}\,.\,\{x \mathrel{*}= \mathsf{H}\}\,.\,\mathit{Use}(\mathsf{measure}\ x, y)\\[1em]
\mathit{System}(x\,{:}\,\mathsf{Qbit}, y\,{:}\,\mathsf{Qbit}, n\,{:}\,\mathsf{0..3}) =\\
\qquad\quad (\mathsf{new}\ q\,{:}\,\hat{}[\mathsf{Qbit}])(\mathit{Alice}(x, q, n) \mid \mathit{Bob}(y, q))
\end{array}
$$

Figure 3: DENSE CODING IN CQP

$$
\begin{array}{rcl}
T & ::= & \mathsf{Int} \mid \mathsf{Unit} \mid \mathsf{Qbit} \mid \hat{}[\widetilde{T}] \mid \mathsf{Op}(1) \mid \mathsf{Op}(2) \mid \ldots\\
v & ::= & x \mid \mathsf{0} \mid \mathsf{1} \mid \ldots \mid \mathsf{unit} \mid \mathsf{H} \mid \ldots\\
e & ::= & v \mid \mathsf{measure}\ \widetilde{e} \mid \widetilde{e} \mathrel{*}= e \mid e{+}e\\
P & ::= & \mathbf{0} \mid (P \mid P) \mid e?[\widetilde{x}\,{:}\,\widetilde{T}]\,.\,P \mid e![\widetilde{e}]\,.\,P \mid \{e\}\,.\,P \mid\\
 & & (\mathsf{new}\ x\,{:}\,T)P \mid (\mathsf{qbit}\ x)P
\end{array}
$$

Figure 4: SYNTAX OF CQP

Alice and Bob share an entangled pair x, y of qubits and Alice wishes to send a two-bit classical value, interpreted as an integer n in the range 0–3, to Bob. Alice applies a Pauli transformation σ_n to x and then sends the transformed x to Bob on a quantum channel. Bob applies the controlled not transformation to x and y and then applies the Hadamard transformation to x. These transformations correspond to a change of basis: it would be desirable to specify changes of basis, and measurements with respect to different bases, more abstractly; this is a challenge for a general theory of quantum data. Finally Bob measures x and y to obtain a two-bit classical value which is the same as n.

The complete system is parameterized on the entangled qubits x and y; again we can make it self-contained by introducing an EPR source.

3 Syntax and Operational Semantics

We now formally define the syntax and operational semantics of the core of CQP, excluding named process definitions and recursion; these features can easily be added. An example at the end of this section illustrates the execution of the teleportation protocol (Figure 2) according to the operational semantics.

3.1 Syntax

The syntax of CQP is defined by the grammar in Figure 4. Types T consist of data types such as Int and Unit (others can easily be added), the type Qbit of qubits, channel types $\hat{}[T_1, \ldots, T_n]$ (specifying that each message is an n-tuple with component types

$$\begin{array}{rcl}
v & ::= & \ldots \mid q \mid c \\
E & ::= & [\,] \mid \mathsf{measure}\ E, \widetilde{e} \mid \mathsf{measure}\ v, E, \widetilde{e} \mid \ \ldots\ \mid \mathsf{measure}\ \widetilde{v}, E \mid \\
 & & E, \widetilde{e} \mathrel{*}= e \mid v, E, \widetilde{e} \mathrel{*}= e \mid \ \ldots\ \mid \widetilde{v} \mathrel{*}= E \mid E{+}e \mid v{+}E \\
F & ::= & [\,]?[\widetilde{x} : \widetilde{T}] \,.\, P \mid [\,]![\widetilde{e}] \,.\, P \mid v![[\,], \widetilde{e}] \,.\, P \mid v![v, [\,], \widetilde{e}] \,.\, P \mid \ \ldots\ \mid \\
 & & v![\widetilde{v}, [\,]] \,.\, P \mid \{[\,]\} \,.\, P
\end{array}$$

Figure 5: INTERNAL SYNTAX OF CQP

$T_1, \ldots, T_n$) and operator types $\mathsf{Op}(n)$ (the type of a unitary operator on n qubits). We use the notation $\widetilde{T} = T_1, \ldots, T_n$ and $\widetilde{e} = e_1, \ldots, e_n$. Values v consist of variables (x, y, z etc.), literal values of data types ($0, 1, \ldots$ and unit) and unitary operators such as the Hadamard operator H. Expressions e consist of values, measurements $\mathsf{measure}\ e_1, \ldots, e_n$, applications of unitary operators $e_1, \ldots, e_n \mathrel{*}= e$, and expressions involving data operators such as $e + e'$ (others can easily be added). Note that although the syntax refers to measurements and transformation of expressions e, the type system will require these expressions to refer to qubits. Processes P consist of the null (terminated) process $\mathbf{0}$, parallel compositions $P \mid Q$, inputs $e?[\widetilde{x} : \widetilde{T}] \,.\, P$ (notation: $\widetilde{x} : \widetilde{T} = x_1 : T_1, \ldots, x_n : T_n$, declaring the types of all the input-bound variables), outputs $e![\widetilde{e}] \,.\, P$, actions $\{e\} \,.\, P$ (typically e will be an application of a unitary operator), channel declarations $(\mathsf{new}\ x : T)P$ and qubit declarations $(\mathsf{qbit}\ x)P$. In inputs and outputs, the expression e will be constrained by the type system to refer to a channel.

The grammar in Figure 5 defines the *internal* syntax of CQP, which is needed in order to define the operational semantics. Values are extended by two new forms: qubit names q, and channel names c. Evaluation contexts $E[\,]$ (for expressions) and $F[\,]$ (for processes) are used in the definition of the operational semantics, in the style of Wright and Felleisen [21]. The structure of $E[\,]$ is used to define call-by-value evaluation of expressions; the hole $[\,]$ specifies the first part of the expression to be evaluated. The structure of $F[\,]$ is used to define reductions of processes, specifying which expressions within a process must be evaluated.

Given a process P we define its free variables $fv(P)$, free qubit names $fq(P)$ and free channel names $fc(P)$ in the usual way; the binders (of x or $\widetilde{x}$) are $y?[\widetilde{x} : \widetilde{T}]$, $(\mathsf{qbit}\ x)$ and $(\mathsf{new}\ x : T)$.

3.2 Operational Semantics

The operational semantics of CQP is defined by reductions (small-step evaluations of expressions, or inter-process communications) alternating with probabilistic transitions. The general form of a reduction is $t \longrightarrow \boxplus_i\, p_i \bullet t_i$ where t and the t_i are configurations consisting of expressions or processes with state information. The notation $\boxplus_i\, p_i \bullet t_i$ denotes a probability distribution over configurations, in which $\Sigma_i p_i = 1$; we may also write this distribution as $p_1 \bullet t_1 \boxplus \cdots \boxplus p_n \bullet t_n$. If the probability distribution contains a single configuration (with probability 1) then we simply write $t \longrightarrow t'$. Probability distributions reduce probabilistically to single configurations: $\boxplus_i\, p_i \bullet t_i \xrightarrow{p_i} t_i$ (with probability p_i, the

$$(\sigma;\phi;u{+}v) \longrightarrow_{\mathsf{v}} (\sigma;\phi;w) \quad \text{if } u \text{ and } v \text{ are integer literals and } u+v=w \qquad \text{(R-PLUS)}$$

$$\begin{array}{l}(q_0,\ldots,q_n = \alpha_0|\psi_0\rangle + \cdots + \alpha_{2^n-1}|\psi_{2^n-1}\rangle;\phi;\mathsf{measure}\ q_0,\ldots,q_{r-1}) \longrightarrow_{\mathsf{v}} \\ \qquad \boxplus_{0\leqslant m<2^r} p_m \bullet (q_0,\ldots,q_n = \frac{\alpha_{l_m}}{p_m}|\psi_{l_m}\rangle + \cdots + \frac{\alpha_{u_m}}{p_m}|\psi_{u_m}\rangle;\phi;m) \\ \text{where } l_m = 2^{n-r}m,\ u_m = 2^{n-r}(m+1)-1,\ p_m = |\alpha_{l_m}|^2 + \cdots + |\alpha_{u_m}|^2 \end{array} \qquad \text{(R-MEASURE)}$$

$$\begin{array}{l}(q_0,\ldots,q_n = |\psi\rangle;\phi;q_0,\ldots,q_{r-1} \mathrel{*}= U) \longrightarrow_{\mathsf{v}} \\ \qquad (q_0,\ldots,q_{n-1} = (U \otimes I_{n-r})|\psi\rangle;\phi;\mathsf{unit}) \end{array} \qquad \text{(R-TRANS)}$$

$$\begin{array}{l}(q_0,\ldots,q_{n-1} = |\psi\rangle;\phi;e) \longrightarrow_{\mathsf{v}} (q_{\pi(0)},\ldots,q_{\pi(n-1)} = \Pi|\psi\rangle;\phi;e) \\ \text{where } \pi \text{ is a permutation and } \Pi \text{ is the corresponding unitary operator} \end{array} \qquad \text{(R-PERM)}$$

$$\frac{(\sigma;\phi;e) \longrightarrow_{\mathsf{v}} \boxplus_i\, p_i \bullet (\sigma_i;\phi_i;e_i)}{(\sigma;\phi;E[e]) \longrightarrow_{\mathsf{e}} \boxplus_i\, p_i \bullet (\sigma_i;\phi_i;E[e_i])} \qquad \text{(R-CONTEXT)}$$

Figure 6: REDUCTION RULES FOR EXPRESSION CONFIGURATIONS

distribution $\boxplus_i\, p_i \bullet t_i$ reduces to t_i).

The semantics of expressions is defined by the reduction relations $\longrightarrow_{\mathsf{v}}$ and $\longrightarrow_{\mathsf{e}}$ (Figure 6), both on configurations of the form $(\sigma;\phi;e)$. If n qubits have been declared then σ has the form $q_0,\ldots,q_{n-1} = |\psi\rangle$ where $|\psi\rangle = \alpha_0|\psi_0\rangle + \cdots + \alpha_{2^n-1}|\psi_{2^n-1}\rangle$ is an element of the 2^n-dimensional vector space with basis $|\psi_0\rangle = |0\ldots0\rangle,\ldots,|\psi_{2^n-1}\rangle = |1\ldots1\rangle$. The remaining part of the configuration, ϕ, is a list of channel names. Reductions $\longrightarrow_{\mathsf{v}}$ are basic steps of evaluation, defined by the rules R-PLUS (and similar rules for any other data operators), R-MEASURE and R-TRANS. Rule R-PERM allows qubits in the state to be permuted, compensating for the way that R-MEASURE and R-TRANS operate on qubits listed first in the state. Reductions $\longrightarrow_{\mathsf{e}}$ extend execution to evaluation contexts $E[\,]$, as defined by rule R-CONTEXT. Note that the probability distribution remains at the top level.

Figure 7 defines the reduction relation $\longrightarrow$ on configurations of the form $(\sigma;\phi;P)$. Rule R-EXPR lifts reductions of expressions to $F[\,]$ contexts, again keeping probability distributions at the top level. Rule R-COM defines communication in the style of pi-calculus, making use of substitution, which is defined in Figure 8 (we assume that bound identifiers are renamed to avoid capture). Rule R-ACT trivially removes actions; in general the reduction of the action expression to v will have involved side-effects such as measurement or transformation of quantum state. Rules R-NEW and R-QBIT create new channels and qubits, updating the state information in the configuration. Note that this treatment of channel creation is different from standard presentations of the pi-calculus; we treat both qubits and channels as elements of a global store. Rule R-PAR allows reduction to take place in parallel contexts, again lifting the probability distribution to the top level, and rule R-CONG allows the use of a structural congruence relation as in the pi-calculus. Structural congruence is the smallest congruence relation (closed under the

$$\frac{(\sigma;\phi;e) \longrightarrow_{\mathsf{e}} \boxplus_i\, p_i \bullet (\sigma_i;\phi_i;e_i)}{(\sigma;\phi;F[e]) \longrightarrow \boxplus_i\, p_i \bullet (\sigma_i;\phi_i;F[e_i])} \qquad \text{(R-EXPR)}$$

$$(\sigma;\phi;c![\widetilde{v}]\,.\,P \mid c?[\widetilde{x}{:}\widetilde{T}]\,.\,Q) \longrightarrow (\sigma;\phi;P \mid Q\{\widetilde{v}/\widetilde{x}\}) \quad \text{if } |\widetilde{v}| = |\widetilde{x}| \qquad \text{(R-COM)}$$

$$(\sigma;\phi;\{v\}\,.\,P) \longrightarrow (\sigma;\phi;P) \qquad \text{(R-ACT)}$$

$$(\sigma;\phi;(\mathsf{new}\ x{:}T)P) \longrightarrow (\sigma;\phi,c;P\{c/x\}) \quad \text{where } c \text{ is fresh} \qquad \text{(R-NEW)}$$

$$(q_0,\ldots,q_n = |\psi\rangle;\phi;(\mathsf{qbit}\ x)P) \longrightarrow (q_0,\ldots,q_n,q = |\psi\rangle \otimes |0\rangle;\phi;P\{q/x\}) \\ \text{where } q \text{ is fresh} \qquad \text{(R-QBIT)}$$

$$\frac{(\sigma;\phi;P) \longrightarrow \boxplus_i\, p_i \bullet (\sigma_i;\phi_i;P_i)}{(\sigma;\phi;P \mid Q) \longrightarrow \boxplus_i\, p_i \bullet (\sigma_i;\phi_i;P_i \mid Q)} \qquad \text{(R-PAR)}$$

$$\frac{P' \equiv P \quad (\sigma;\phi;P) \longrightarrow \boxplus_i\, p_i \bullet (\sigma_i;\phi_i;P_i) \quad \forall i.(P_i \equiv P_i')}{(\sigma;\phi;P') \longrightarrow \boxplus_i\, p_i \bullet (\sigma_i;\phi_i;P_i')} \qquad \text{(R-CONG)}$$

$$\boxplus_i\, p_i \bullet (\sigma_i;\phi_i;P_i) \xrightarrow{p_i} (\sigma_i;\phi_i;P_i) \qquad \text{(R-PROB)}$$

Figure 7: REDUCTION RULES FOR PROCESS CONFIGURATIONS

$$\begin{array}{rcll}
v\{\widetilde{v}/\widetilde{x}\} & = & v & \text{if } v \text{ is not a variable} \\
x\{\widetilde{v}/\widetilde{x}\} & = & v_i & \text{if } x = x_i \\
x\{\widetilde{v}/\widetilde{x}\} & = & x & \text{if } x \notin \widetilde{x} \\
(\mathsf{measure}\ \widetilde{e})\{\widetilde{v}/\widetilde{x}\} & = & \mathsf{measure}\ \widetilde{e}\{\widetilde{v}/\widetilde{x}\} & \\
(\widetilde{e} \mathrel{*}= e)\{\widetilde{v}/\widetilde{x}\} & = & \widetilde{e}\{\widetilde{v}/\widetilde{x}\} \mathrel{*}= e\{\widetilde{v}/\widetilde{x}\} & \\
(e + e')\{\widetilde{v}/\widetilde{x}\} & = & e\{\widetilde{v}/\widetilde{x}\} + e'\{\widetilde{v}/\widetilde{x}\} & \text{etc.} \\
\mathbf{0}\{\widetilde{v}/\widetilde{x}\} & = & \mathbf{0} & \\
(P \mid Q)\{\widetilde{v}/\widetilde{x}\} & = & P\{\widetilde{v}/\widetilde{x}\} \mid Q\{\widetilde{v}/\widetilde{x}\} & \\
(e?[\widetilde{y}{:}\widetilde{T}]\,.\,P)\{\widetilde{v}/\widetilde{x}\} & = & (e\{\widetilde{v}/\widetilde{x}\})?[\widetilde{y}{:}\widetilde{T}]\,.\,P\{\widetilde{v}/\widetilde{x}\} & \\
(e![\widetilde{e}]\,.\,P)\{\widetilde{v}/\widetilde{x}\} & = & (e\{\widetilde{v}/\widetilde{x}\})![\widetilde{e}\{\widetilde{v}/\widetilde{x}\}]\,.\,P\{\widetilde{v}/\widetilde{x}\} & \\
(\{e\}\,.\,P)\{\widetilde{v}/\widetilde{x}\} & = & \{e\{\widetilde{v}/\widetilde{x}\}\}\,.\,P\{\widetilde{v}/\widetilde{x}\} & \\
((\mathsf{new}\ y{:}T)P)\{\widetilde{v}/\widetilde{x}\} & = & (\mathsf{new}\ y{:}T)(P\{\widetilde{v}/\widetilde{x}\}) & \\
((\mathsf{qbit}\ y)P)\{\widetilde{v}/\widetilde{x}\} & = & (\mathsf{qbit}\ y)(P\{\widetilde{v}/\widetilde{x}\}) &
\end{array}$$

Figure 8: SUBSTITUTION

$$P \mid \mathbf{0} \equiv P \qquad \text{(S-NIL)}$$

$$P \mid Q \equiv Q \mid P \qquad \text{(S-COMM)}$$

$$P \mid (Q \mid R) \equiv (P \mid Q) \mid R \qquad \text{(S-ASSOC)}$$

Figure 9: STRUCTURAL CONGRUENCE

process constructions) containing α-equivalence and closed under the rules in Figure 9.

3.3 Example: Execution of Teleportation

Figure 10 shows the execution of the teleportation protocol from Figure 1, as far as the measurement ($\mathsf{measure}\ z, x$) which introduces a probability distribution over configurations. The initial state $x, y, z = \frac{1}{\sqrt{2}}|000\rangle + \frac{1}{\sqrt{2}}|110\rangle$ provides the entangled pair of qubits which are needed as parameters for $System$, and the qubit z (in this case $|0\rangle$) which Alice teleports.

Once the probability distribution is reached, the next step is a probabilistic transition: to one of the four configurations, with equal probability. Reductions then continue, in all cases resulting in $Use(y)$ where the qubits are in a pure state with $y = |0\rangle$.

4 Type System

The typing rules defined in Figure 11 apply to the syntax defined in Figure 4. Environments Γ are mappings from variables to types in the usual way. Typing judgements are of two kinds. $\Gamma \vdash e : T$ means that expression e has type T in environment Γ. $\Gamma \vdash P$ means that process P is well-typed in environment Γ. The rules for expressions are straightforward; note that in rule T-TRANS, $x_1, \ldots, x_n$ must be distinct variables of type Qbit.

In rule T-PAR the operation $+$ on environments (Definition 1) is the key to ensuring that each qubit is controlled by a unique part of a system. An implicit hypothesis of T-PAR is that $\Gamma_1 + \Gamma_2$ must be defined. This is very similar to the linear type system for the pi-calculus, defined by Kobayashi *et al.* [10].

Definition 1 (Addition of Environments)
The partial operation of adding a typed variable to an environment, $\Gamma + x : T$*, is defined by*

$$\begin{array}{lcll} \Gamma + x{:}T & = & \Gamma, x{:}T & \textit{if } x \notin dom(\Gamma) \\ \Gamma + x{:}T & = & \Gamma & \textit{if } T \neq \mathsf{Qbit} \textit{ and } x{:}T \in \Gamma \\ \Gamma + x{:}T & = & \multicolumn{2}{l}{\textit{undefined, otherwise}} \end{array}$$

This operation is extended inductively to a partial operation $\Gamma + \Delta$ *on environments.*

Rule T-OUT allows output of classical values and qubits to be combined, but the qubits must be distinct variables and they cannot be used by the continuation of the outputting process (note the hypothesis $\Gamma \vdash P$). The remaining rules are straightforward.

According to the operational semantics, execution of (qbit) and (new) declarations introduces qubit names and channel names. In order to be able to use the type system to prove results about the behaviour of executing processes, we introduce the internal type system (Figure 12). This uses judgements $\Gamma; \Sigma; \Phi \vdash e : T$ and $\Gamma; \Sigma; \Phi \vdash P$ where Σ is a set of qubit names and Φ is a mapping from channel names to channel types. Most of the typing rules are straightforward extensions of the corresponding rules in Figure 11. Because references to qubits may now be either variables or explicit qubit names, the rules represent them by general expressions e and impose conditions that e is either a variable or a qubit name. This is seen in rules T-TRANS and T-OUT. Note that in T-PAR, the operation $\Sigma_1 + \Sigma_2$ is disjoint union and an implicit hypothesis is that Σ_1 and Σ_2 are disjoint.

$$(x, y, z = \tfrac{1}{\sqrt{2}}|000\rangle + \tfrac{1}{\sqrt{2}}|110\rangle; ; \mathit{System}(x, y, z))$$

$$\downarrow$$

$$\begin{gathered} x, y, z = \tfrac{1}{\sqrt{2}}|000\rangle + \tfrac{1}{\sqrt{2}}|110\rangle\ ;\ ; \\ (\mathsf{new}\ c{:}\widehat{\ }[\mathsf{0..3}])(\mathit{Alice}(x, c, z) \mid \mathit{Bob}(y, c)) \end{gathered}$$

$$\downarrow$$

$$\begin{gathered} x, y, z = \tfrac{1}{\sqrt{2}}|000\rangle + \tfrac{1}{\sqrt{2}}|110\rangle\ ;\ c\ ; \\ \mathit{Alice}(x, c, z) \mid \mathit{Bob}(y, c) \end{gathered}$$

$$\downarrow$$

$$\begin{gathered} x, y, z = \tfrac{1}{\sqrt{2}}|000\rangle + \tfrac{1}{\sqrt{2}}|110\rangle\ ;\ c\ ; \\ \{z, x \mathrel{*}= \mathsf{CNot}\} \,.\, \{z \mathrel{*}= \mathsf{H}\} \,.\, c![\mathsf{measure}\ z, x] \,.\, \mathbf{0} \\ \mid c?[r : \mathsf{0..3}] \,.\, \{y \mathrel{*}= \sigma_r\} \,.\, \mathit{Use}(y) \end{gathered}$$

$$\downarrow$$

$$\begin{gathered} x, y, z = \tfrac{1}{\sqrt{2}}|000\rangle + \tfrac{1}{\sqrt{2}}|110\rangle\ ;\ c\ ; \\ \{z, x \mathrel{*}= \mathsf{CNot}\} \,.\, \{z \mathrel{*}= \mathsf{H}\} \,.\, c![\mathsf{measure}\ z, x] \,.\, \mathbf{0} \\ \mid c?[r : \mathsf{0..3}] \,.\, \{y \mathrel{*}= \sigma_r\} \,.\, \mathit{Use}(y) \end{gathered}$$

$$\downarrow$$

$$\begin{gathered} z, x, y = \tfrac{1}{\sqrt{2}}|000\rangle + \tfrac{1}{\sqrt{2}}|011\rangle\ ;\ c\ ; \\ \{z, x \mathrel{*}= \mathsf{CNot}\} \,.\, \{z \mathrel{*}= \mathsf{H}\} \,.\, c![\mathsf{measure}\ z, x] \,.\, \mathbf{0} \\ \mid c?[r : \mathsf{0..3}] \,.\, \{y \mathrel{*}= \sigma_r\} \,.\, \mathit{Use}(y) \end{gathered}$$

$$\downarrow$$

$$\begin{gathered} z, x, y = \tfrac{1}{\sqrt{2}}|000\rangle + \tfrac{1}{\sqrt{2}}|011\rangle\ ;\ c\ ; \\ \{z \mathrel{*}= \mathsf{H}\} \,.\, c![\mathsf{measure}\ z, x] \,.\, \mathbf{0} \mid c?[r : \mathsf{0..3}] \,.\, \{y \mathrel{*}= \sigma_r\} \,.\, \mathit{Use}(y) \end{gathered}$$

$$\downarrow$$

$$\begin{gathered} z, x, y = \tfrac{1}{2}|000\rangle + \tfrac{1}{2}|011\rangle + \tfrac{1}{2}|100\rangle + \tfrac{1}{2}|111\rangle\ ;\ c\ ; \\ c![\mathsf{measure}\ z, x] \,.\, \mathbf{0} \mid c?[r : \mathsf{0..3}] \,.\, \{y \mathrel{*}= \sigma_r\} \,.\, \mathit{Use}(y) \end{gathered}$$

$$\downarrow$$

$$\begin{gathered} \tfrac{1}{4} \bullet ((z, x, y = |000\rangle; a, c; c![0] \,.\, \mathbf{0} \mid c?[r : \mathsf{0..3}] \,.\, \{y \mathrel{*}= \sigma_r\} \,.\, \mathit{Use}(y))) \\ \boxplus \tfrac{1}{4} \bullet ((z, x, y = |011\rangle; a, c; c![1] \,.\, \mathbf{0} \mid c?[r : \mathsf{0..3}] \,.\, \{y \mathrel{*}= \sigma_r\} \,.\, \mathit{Use}(y))) \\ \boxplus \tfrac{1}{4} \bullet ((z, x, y = |100\rangle; a, c; c![2] \,.\, \mathbf{0} \mid c?[r : \mathsf{0..3}] \,.\, \{y \mathrel{*}= \sigma_r\} \,.\, \mathit{Use}(y))) \\ \boxplus \tfrac{1}{4} \bullet ((z, x, y = |111\rangle; a, c; c![3] \,.\, \mathbf{0} \mid c?[r : \mathsf{0..3}] \,.\, \{y \mathrel{*}= \sigma_r\} \,.\, \mathit{Use}(y))) \end{gathered}$$

Figure 10: EXECUTION OF THE TELEPORTATION PROTOCOL

$$\Gamma \vdash v : \mathsf{Int} \quad \text{if } v \text{ is an integer literal} \qquad \text{(T-IntLit)}$$

$$\Gamma \vdash \mathsf{unit} : \mathsf{Unit} \qquad \text{(T-Unit)}$$

$$\Gamma \vdash \mathsf{H} : \mathsf{Op}(2) \quad \text{etc.} \qquad \text{(T-Op)}$$

$$\Gamma, x\!:\!T \vdash x : T \qquad \text{(T-Var)}$$

$$\frac{\Gamma \vdash e : \mathsf{Int} \quad \Gamma \vdash e' : \mathsf{Int}}{\Gamma \vdash e{+}e' : \mathsf{Int}} \qquad \text{(T-Plus)}$$

$$\frac{\Gamma \vdash \widetilde{e} : \widetilde{\mathsf{Qbit}}}{\Gamma \vdash \mathsf{measure}\ \widetilde{e} : \mathsf{Int}} \qquad \text{(T-Measure)}$$

$$\frac{\forall i.(x_i\!:\!\mathsf{Qbit} \in \Gamma) \quad x_1 \ldots x_n \text{ distinct} \quad \Gamma \vdash U : \mathsf{Op}(n)}{\Gamma \vdash x_1, \ldots, x_n \mathrel{*}= U : \mathsf{Unit}} \qquad \text{(T-Trans)}$$

$$\Gamma \vdash \mathbf{0} \qquad \text{(T-Nil)}$$

$$\frac{\Gamma_1 \vdash P \quad \Gamma_2 \vdash Q}{\Gamma_1 + \Gamma_2 \vdash P \mid Q} \qquad \text{(T-Par)}$$

$$\frac{\Gamma \vdash x : \hat{}[T_1, \ldots, T_n] \quad \Gamma, y_1\!:\!T_1, \ldots, y_n\!:\!T_n \vdash P}{\Gamma \vdash x?[y_1\!:\!T_1, \ldots, y_n\!:\!T_n] \,.\, P} \qquad \text{(T-In)}$$

$$\frac{\begin{array}{cc} \Gamma \vdash x : \hat{}[T_1, \ldots, T_m, \mathsf{Qbit}, \ldots, \mathsf{Qbit}] & \forall i.(T_i \neq \mathsf{Qbit}) \\ \forall i.(\Gamma \vdash e_i : T_i) \quad y_1, \ldots, y_n \text{ distinct} & \Gamma \vdash P \end{array}}{\Gamma, y_1 : \mathsf{Qbit} \ldots, y_n\!:\!\mathsf{Qbit} \vdash x![e_1, \ldots, e_m, y_1, \ldots, y_n] \,.\, P} \qquad \text{(T-Out)}$$

$$\frac{\Gamma \vdash e : T \quad \Gamma \vdash P}{\Gamma \vdash \{e\} \,.\, P} \qquad \text{(T-Action)}$$

$$\frac{\Gamma, x\!:\!\hat{}[T_1, \ldots, T_n] \vdash P}{\Gamma \vdash (\mathsf{new}\ x\!:\!\hat{}[T_1, \ldots, T_n])P} \qquad \text{(T-New)}$$

$$\frac{\Gamma, x\!:\!\mathsf{Qbit} \vdash P}{\Gamma \vdash (\mathsf{qbit}\ x)P} \qquad \text{(T-Qbit)}$$

Figure 11: Typing rules

$$\Gamma;\Sigma;\Phi \vdash v : \mathsf{Int} \quad \text{if } v \text{ is an integer literal} \qquad \text{(T-IntLit)}$$

$$\Gamma;\Sigma;\Phi \vdash \mathsf{unit} : \mathsf{Unit} \qquad \text{(T-Unit)}$$

$$\Gamma;\Sigma;\Phi \vdash \mathsf{H} : \mathsf{Op}(2) \quad \text{etc.} \qquad \text{(T-Op)}$$

$$\Gamma, x\!:\!T;\Sigma;\Phi \vdash x : T \qquad \text{(T-Var)}$$

$$\Gamma;\Sigma, q;\Phi \vdash q : \mathsf{Qbit} \qquad \text{(T-IdQ)}$$

$$\Gamma;\Sigma;\Phi, c\!:\!T \vdash c : T \qquad \text{(T-IdC)}$$

$$\frac{\Gamma;\Sigma;\Phi \vdash e : \mathsf{Int} \quad \Gamma;\Sigma;\Phi \vdash e' : \mathsf{Int}}{\Gamma;\Sigma;\Phi \vdash e+e' : \mathsf{Int}} \qquad \text{(T-Plus)}$$

$$\frac{\Gamma;\Sigma;\Phi \vdash \widetilde{e} : \widetilde{\mathsf{Qbit}}}{\Gamma;\Sigma;\Phi \vdash \mathsf{measure}\ \widetilde{e} : \mathsf{Int}} \qquad \text{(T-Measure)}$$

$$\frac{\begin{array}{c}\forall i.(\Gamma;\Sigma;\Phi \vdash e_i : \mathsf{Qbit}) \quad \Gamma;\Sigma;\Phi \vdash U : \mathsf{Op}(n) \\ \text{each } e_i \text{ is either } x_i \text{ or } q_i\text{, all distinct}\end{array}}{\Gamma;\Sigma;\Phi \vdash e_1,\ldots,e_n \mathrel{*}= U : \mathsf{Unit}} \qquad \text{(T-Trans)}$$

$$\Gamma;\Sigma;\Phi \vdash \mathbf{0} \qquad \text{(T-Nil)}$$

$$\frac{\Gamma_1;\Sigma_1;\Phi \vdash P \quad \Gamma_2;\Sigma_2;\Phi \vdash Q}{\Gamma_1+\Gamma_2;\Sigma_1+\Sigma_2;\Phi \vdash P \mid Q} \qquad \text{(T-Par)}$$

$$\frac{\Gamma;\Sigma;\Phi \vdash e : \widehat{\ }[T_1,\ldots,T_n] \quad \Gamma, y_1\!:\!T_1,\ldots,y_n\!:\!T_n;\Sigma;\Phi \vdash P}{\Gamma;\Sigma;\Phi \vdash e?[y_1\!:\!T_1,\ldots,y_n\!:\!T_n]\,.\,P} \qquad \text{(T-In)}$$

$$\frac{\begin{array}{c}\Gamma;\Sigma;\Phi \vdash e : \widehat{\ }[\widetilde{T},\widetilde{\mathsf{Qbit}}] \quad \forall i.(T_i \neq \mathsf{Qbit}) \quad \forall i.(\Gamma;\Sigma;\Phi \vdash e_i : T_i) \\ \forall i.(\Gamma;\Sigma;\Phi \vdash f_i : \mathsf{Qbit}) \quad \Gamma;\Sigma;\Phi \vdash P \\ \widetilde{f} \text{ consists of distinct variables } \widetilde{f_x} \text{ and distinct qubit names } \widetilde{f_q}\end{array}}{\Gamma, \widetilde{f_x}\!:\!\widetilde{\mathsf{Qbit}};\Sigma, \widetilde{f_q}\!:\!\widetilde{\mathsf{Qbit}};\Phi \vdash e![e_1,\ldots,e_m,f_1,\ldots,f_n]\,.\,P} \qquad \text{(T-Out)}$$

$$\frac{\Gamma;\Sigma;\Phi \vdash e : T \quad \Gamma;\Sigma;\Phi \vdash P}{\Gamma;\Sigma;\Phi \vdash \{e\}\,.\,P} \qquad \text{(T-Action)}$$

$$\frac{\Gamma, x\!:\!\widehat{\ }[T_1,\ldots,T_n];\Sigma;\Phi \vdash P}{\Gamma;\Sigma;\Phi \vdash (\mathsf{new}\ x\!:\!\widehat{\ }[T_1,\ldots,T_n])P} \qquad \text{(T-New)}$$

$$\frac{\Gamma, x\!:\!\mathsf{Qbit};\Sigma;\Phi \vdash P}{\Gamma;\Sigma;\Phi \vdash (\mathsf{qbit}\ x)P} \qquad \text{(T-Qbit)}$$

Figure 12: Internal typing rules

5 Soundness of the Type System

We prove a series of standard lemmas, following the approach of Wright and Felleisen [21], leading to a proof that typing is preserved by execution of processes (Theorem 1). We then prove that in a typable process, each qubit is used by at most one of any parallel collection of sub-processes (Theorem 2); because of type preservation, this property holds at every step of the execution of a typable process. This reflects the physical reality of the protocols which we want to model.

Lemma 1 (Typability of Subterms in E)
If $\mathcal{D}$ is a typing derivation concluding $\Gamma; \Sigma; \Phi \vdash E[e] : T$ then there exists U such that $\mathcal{D}$ has a subderivation $\mathcal{D}'$ concluding $\Gamma; \Sigma; \Phi \vdash e : U$ and the position of $\mathcal{D}'$ in $\mathcal{D}$ corresponds to the position of the hole in $E[\,]$.

Proof: By induction on the structure of $E[\,]$. □

Lemma 2 (Replacement in E)
If

1. *$\mathcal{D}$ is a typing derivation concluding $\Gamma; \Sigma; \Phi \vdash E[e] : T$*
2. *$\mathcal{D}'$ is a subderivation of $\mathcal{D}$ concluding $\Gamma; \Sigma; \Phi \vdash e : U$*
3. *$\mathcal{D}'$ occurs in $\mathcal{D}$ in a position corresponding to the hole in $E[\,]$*
4. *$\Gamma; \Sigma; \Phi \vdash e' : U$*

then $\Gamma; \Sigma; \Phi \vdash E[e'] : T$.

Proof: Replace $\mathcal{D}'$ in $\mathcal{D}$ by a derivation of $\Gamma; \Sigma; \Phi \vdash e' : U$. □

Lemma 3 (Type Preservation for $\longrightarrow_{\mathsf{v}}$)
If $\Gamma; \Sigma; \Phi \vdash e : T$ and $(\sigma; \phi; e) \longrightarrow_{\mathsf{v}} \boxplus_i p_i \bullet (\sigma_i; \phi_i; e_i)$ and $\Sigma = dom(\sigma)$ and $\phi = dom(\Phi)$ then $\forall i.(\sigma_i = \sigma)$ and $\forall i.(\phi_i = \phi)$ and $\forall i.(\Gamma; \Sigma; \Phi \vdash e_i : T)$.

Proof: Straightforward from the definition of $\longrightarrow_{\mathsf{v}}$ by examining each case. □

Lemma 4 (Type Preservation for $\longrightarrow_{\mathsf{e}}$)
If $\Gamma; \Sigma; \Phi \vdash e : T$ and $(\sigma; \phi; e) \longrightarrow_{\mathsf{e}} \boxplus_i p_i \bullet (\sigma_i; \phi_i; e_i)$ and $\Sigma = dom(\sigma)$ and $\phi = dom(\Phi)$ then $\forall i.(\sigma_i = \sigma)$ and $\forall i.(\phi_i = \phi)$ and $\forall i.(\Gamma; \Sigma; \Phi \vdash e_i : T)$.

Proof: $(\sigma; \phi; e) \longrightarrow_{\mathsf{e}} \boxplus_i p_i \bullet (\sigma_i; \phi_i; e_i)$ is derived by R-CONTEXT, so for some $E[\,]$ we have $e = E[f]$ and $\forall i.(e_i = E[f_i])$ and $(\sigma; \phi; f) \longrightarrow_{\mathsf{v}} \boxplus_i p_i \bullet (\sigma_i; \phi_i; f_i)$. From $\Gamma; \Sigma; \Phi \vdash E[f] : T$, Lemma 1 gives $\Gamma; \Sigma; \Phi \vdash f : U$ for some U, Lemma 3 gives $\forall i.(\Gamma; \Sigma; \Phi \vdash f - i : U)$ and $\forall i.(\sigma_i = \sigma)$ and $\forall i.(\phi_i = \phi)$, and Lemma 2 gives $\forall i.(\Gamma; \Sigma; \Phi \vdash E[f_i] : T)$. □

Lemma 5 (Typability of Subterms in F)
If $\mathcal{D}$ is a typing derivation concluding $\Gamma; \Sigma; \Phi \vdash F[e]$ then there exists T such that $\mathcal{D}$ has a subderivation $\mathcal{D}'$ concluding $\Gamma; \Sigma; \Phi \vdash e : T$ and the position of $\mathcal{D}'$ in $\mathcal{D}$ corresponds to the position of the hole in $F[\,]$.

Proof: By case-analysis on the structure of $F[\,]$. □

Lemma 6 (Replacement in F)
If

1. *$\mathcal{D}$ is a typing derivation concluding $\Gamma; \Sigma; \Phi \vdash F[e]$*
2. *$\mathcal{D}'$ is a subderivation of $\mathcal{D}$ concluding $\Gamma; \Sigma; \Phi \vdash e : T$*
3. *$\mathcal{D}'$ occurs in $\mathcal{D}$ in a position corresponding to the hole in $F[\,]$*
4. *$\Gamma; \Sigma; \Phi \vdash e' : T$*

then $\Gamma; \Sigma; \Phi \vdash E[e']$.

Proof: Replace $\mathcal{D}'$ in $\mathcal{D}$ by a derivation of $\Gamma; \Sigma; \Phi \vdash e' : T$. □

Lemma 7 (Weakening for Expressions)
If $\Gamma; \Sigma; \Phi \vdash e : T$ and $\Gamma \subseteq \Gamma'$ and $\Sigma \subseteq \Sigma'$ and $\Phi \subseteq \Phi'$ then $\Gamma'; \Sigma'; \Phi' \vdash e : T$.

Proof: Straightforward induction on the derivation of $\Gamma; \Sigma; \Phi \vdash e : T$. □

Lemma 8
If $\Gamma; \Sigma; \Phi \vdash e : T$ then $fv(e) \subseteq dom(\Gamma)$ and $fq(e) \subseteq \Sigma$ and $fc(e) \subseteq dom(\Phi)$.

Proof: By induction on the derivation of $\Gamma; \Sigma; \Phi \vdash e : T$. □

Lemma 9
If $\Gamma; \Sigma; \Phi \vdash P$ then $fv(P) \subseteq dom(\Gamma)$ and $fq(P) \subseteq \Sigma$ and $fc(P) \subseteq dom(\Phi)$.

Proof: By induction on the derivation of $\Gamma; \Sigma; \Phi \vdash P$. □

Lemma 10 (Substitution in Expressions)
Assume that $\Gamma, \widetilde{x}\!:\!\widetilde{T}; \Sigma; \Phi \vdash e : T$ and let $\widetilde{v}$ be values such that, for each i:

1. *if $T_i = \mathsf{Qbit}$ then v_i is either a variable or a qubit name*
2. *if $T_i = \mathsf{Qbit}$ and $v_i = y_i$ (a variable) then $y_i \notin \Gamma, \widetilde{x}\!:\!\widetilde{T}$*
3. *if $T_i = \mathsf{Qbit}$ and $v_i = q_i$ (a qubit name) then $q_i \notin \Sigma$*
4. *if $T_i \neq \mathsf{Qbit}$ then $\Gamma; \Sigma; \Phi \vdash v_i : T_i$.*

Let $\widetilde{y}$ be the variables of type Qbit from $\widetilde{v}$ (corresponding to condition (2)) and assume that they are distinct; let $\widetilde{q}$ be the qubit names from $\widetilde{v}$ (corresponding to condition (3)) and assume that they are distinct. Then $\Gamma, \widetilde{y}\!:\!\widetilde{\mathsf{Qbit}}; \Sigma, \widetilde{q}; \Phi \vdash e\{\widetilde{v}/\widetilde{x}\} : T$.

Proof: By induction on the derivation of $\Gamma, \widetilde{x}\!:\!\widetilde{T}; \Sigma; \Phi \vdash e : T$. □

Lemma 11 (Substitution in Processes)
Assume that $\Gamma, \widetilde{x}\!:\!\widetilde{T}; \Sigma; \Phi \vdash P$ and let $\widetilde{v}$ be values such that, for each i:

1. *if* $T_i = \mathsf{Qbit}$ *then* v_i *is either a variable or a qubit name*
2. *if* $T_i = \mathsf{Qbit}$ *and* $v_i = y_i$ *(a variable) then* $y_i \notin \Gamma, \widetilde{x}\!:\!\widetilde{T}$
3. *if* $T_i = \mathsf{Qbit}$ *and* $v_i = q_i$ *(a qubit name) then* $q_i \notin \Sigma$
4. *if* $T_i \neq \mathsf{Qbit}$ *then* $\Gamma; \Sigma; \Phi \vdash v_i : T_i$.

Let $\widetilde{y}$ *be the variables of type* Qbit *from* $\widetilde{v}$ *(corresponding to condition (2)) and assume that they are distinct; let* $\widetilde{q}$ *be the qubit names from* $\widetilde{v}$ *(corresponding to condition (3)) and assume that they are distinct. Then* $\Gamma, \widetilde{y}\!:\!\widetilde{\mathsf{Qbit}}; \Sigma, \widetilde{q}; \Phi \vdash P\{\widetilde{v}/\widetilde{x}\}$.

Proof: By induction on the derivation of $\Gamma, \widetilde{x}\!:\!\widetilde{T}; \Sigma; \Phi \vdash P$. The key cases are T-PAR and T-OUT.

For T-PAR the final step in the typing derivation has the form

$$\frac{\Gamma_1; \Sigma_1; \Phi \vdash P \quad \Gamma_2; \Sigma_2; \Phi \vdash Q}{\Gamma, \widetilde{x}\!:\!\widetilde{T}; \Sigma; \Phi \vdash P \mid Q}$$

where $\Gamma_1 + \Gamma_2 = \Gamma, \widetilde{x}\!:\!\widetilde{T}$ and $\Sigma_1 + \Sigma_2 = \Sigma$. Each variable of type Qbit in $\Gamma, \widetilde{x}\!:\!\widetilde{T}$ is in exactly one of Γ_1 and Γ_2. Because the free variables of P and Q are contained in Γ_1 and Γ_2 respectively, substitution into $P \mid Q$ splits into disjoint substitutions into P and Q. The induction hypothesis gives typings for $P\{\widetilde{v}/\widetilde{x}\}$ and $Q\{\widetilde{v}/\widetilde{x}\}$, which combine (by T-PAR) to give $\Gamma, \widetilde{y}\!:\!\widetilde{\mathsf{Qbit}}; \Sigma, \widetilde{q}; \Phi \vdash P \mid Q\{\widetilde{v}/\widetilde{x}\}$. □

Lemma 12 (Structural Congruence Preserves Typing)
If $\Gamma; \Sigma; \Phi \vdash P$ *and* $P \equiv Q$ *then* $\Gamma; \Sigma; \Phi \vdash Q$.

Proof: By induction on the derivation of $P \equiv Q$. □

Lemma 13 (Embedding in the Internal Type System)
If $\Gamma \vdash e : T$ *then* $\Gamma; \emptyset; \emptyset \vdash e : T$. *If* $\Gamma \vdash P$ *then* $\Gamma; \emptyset; \emptyset \vdash P$.

Theorem 1 (Type Preservation for $\longrightarrow$)
If $\Gamma; \Sigma; \Phi \vdash P$ *and* $(\sigma; \phi; P) \longrightarrow \boxplus_i p_i \bullet (\sigma_i; \phi_i; P_i)$ *and* $\Sigma = dom(\sigma)$ *and* $\phi = dom(\Phi)$ *then* $\forall i.(\sigma_i = \sigma)$ *and* $\forall i.(\phi_i = \phi)$ *and* $\forall i.(\Gamma; \Sigma; \Phi \vdash P_i)$.

Proof: By induction on the derivation of $(\sigma; \phi; P) \longrightarrow \boxplus_i p_i \bullet (\sigma_i; \phi_i; P_i)$, in each case examining the final steps in the derivation of $\Gamma; \Sigma; \Phi \vdash P$. □

Theorem 2 (Unique Ownership of Qubits)
If $\Gamma; \Sigma; \Phi \vdash P \mid Q$ *then* $fq(P) \cap fq(Q) = \emptyset$.

Proof: The final step in the derivation of $\Gamma; \Sigma; \Phi \vdash P \mid Q$ has the form

$$\frac{\Gamma_1; \Sigma_1; \Phi \vdash P \quad \Gamma_2; \Sigma_2; \Phi \vdash Q}{\Gamma; \Sigma; \Phi \vdash P \mid Q}$$

where $\Gamma = \Gamma_1 + \Gamma_2$ and $\Sigma = \Sigma_1 + \Sigma_2$. By Lemma 9, $fq(P) \subseteq \Sigma_1$ and $fq(Q) \subseteq \Sigma_2$. The implicit hypothesis of the typing rule T-PAR is that $\Sigma_1 + \Sigma_2$ is defined, meaning that $\Sigma_1 \cap \Sigma_2 = \emptyset$. Hence $fq(P) \cap fq(Q) = \emptyset$. □

6 Conclusions and Future Work

We have defined a language, CQP, for modelling systems which combine quantum and classical communication and computation; in particular, qubits can be sent from process to process on communication channels. CQP has a formal operational semantics, and a static type system which guarantees that sending a qubit corresponds to a physical transfer of ownership.

The syntax and semantics of CQP are based on a combination of the pi-calculus and an expression language which includes measurement and transformation of quantum state. We have presented the syntax and semantics in a style which makes it easy to enrich the language. Purely classical features such as functions and assignable variables should be straightforward to add. Extensions which combine quantum data with enhanced classical control structures require more care; we have not yet considered functions of quantum data.

We intend to use CQP as the basis for analysis of quantum protocols using a variety of techniques including model-checking, simulation and type theory.

References

[1] S. Abramsky and B. Coecke. A categorical semantics of quantum protocols. In *Proceedings, Nineteenth Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society Press, 2004.

[2] A. Ambainis, A. Nayak, A. Ta-Shma, and U. Vazirani. Dense quantum coding and a lower bound for 1-way quantum finite automata. Quantum Physics Archive: `arXiv:quant-ph/9804043`, April 1998.

[3] 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, Bangalore, India*, pages 175–179, December 1984.

[4] C. H. Bennett, G. Brassard, C. Crépeau, R. Jozsa, A. Peres, and W. K. Wootters. Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels. *Phys. Rev. Lett.*, 70:1895–1899, 1993.

[5] S. Bettelli, T. Calarco, and L. Serafini. Toward an architecture for quantum programming. *Eur. Phys. J. D*, 25:181–200, 2003.

[6] H. de Riedmatten, I. Marcikic, W. Tittel, H. Zbinden, D. Collins, and N. Gisin. Long distance quantum teleportation in a quantum relay configuration. *Phys. Rev. Lett.*, 92(4), 2004.

[7] J.-Y. Girard. Linear Logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[8] P. Jorrand and M. Lalire. Toward a quantum process algebra. Quantum Physics Archive: `arXiv:quant-ph/0312067`, December 2003.

[9] E. Knill. Conventions for quantum pseudocode. Technical Report LAUR-96-2724, Los Alamos National Laboratory, 1996.

[10] N. Kobayashi, B. C. Pierce, and D. N. Turner. Linearity and the Pi-Calculus. *ACM Transactions on Programming Languages and Systems*, 21(5):914–947, September 1999.

[11] D. Mayers. Unconditional Security in Quantum Cryptography. *Journal of the ACM*, 48(3):351–406, May 2001.

[12] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, I and II. *Information and Computation*, 100(1):1–77, September 1992.

[13] B. Ömer. Quantum programming in QCL. Master's thesis, Technical University of Vienna, 2000.

[14] B. C. Pierce and D. Sangiorgi. Typing and subtyping for mobile processes. *Mathematical Structures in Computer Science*, 6(5), 1996.

[15] A. Poppe, A. Fedrizzi, T. Lorünser, O. Maurhardt, R. Ursin, H. R. Böhm, M. Peev, M. Suda, C. Kurtsiefer, H. Weinfurter, T. Jennewein, and A. Zeilinger. Practical quantum key distribution with polarization entangled photons. Quantum Physics Archive: `arXiv:quant-ph/0404115`, April 2004.

[16] P. Ryan, S. Schneider, M. Goldsmith, G. Lowe, and B. Roscoe. *Modelling and Analysis of Security Protocols*. Addison-Wesley, 2001.

[17] J. W. Sanders and P. Zuliani. Quantum programming. In *Mathematics of Program Construction*, volume 1837 of *Springer LNCS*, 2000.

[18] D. Sangiorgi and D. Walker. *The π-calculus: a Theory of Mobile Processes*. Cambridge University Press, 2001.

[19] P. Selinger. Towards a quantum programming language. *Mathematical Structures in Computer Science*, 2004. To appear.

[20] A. van Tonder. Quantum computation, categorical semantics and linear logic. Quantum Physics Archive: `arXiv:quant-ph/0312174`, December 2003.

[21] A. K. Wright and M. Felleisen. A syntactic approach to type soundness. *Information and Computation*, 115(1):38–94, 1994.