Towards Quantum Multiparty Session Types

IVAN LANESE, Università di Bologna, Italy and Université Côte d'Azur, France
UGO DAL LAGO, Università di Bologna, Italy and Université Côte d'Azur, France
VIKRAMAN CHOUDHURY, Università di Bologna, Italy and Université Côte d'Azur, France

Multiparty Session Types (MPSTs) are a typing discipline for message-passing protocols that guarantee communication safety properties, such as deadlock-freedom. We propose a quantum extension of MPSTs, called Quantum MPSTs (QMPSTs), with the aim of specifying quantum protocols. QMPSTs guarantee usual communication safety properties, in addition to safety properties specific to quantum information, such as no-cloning and no-deleting. We exhibit the use of QMPSTs to verify Quantum Teleportation. The full paper (to appear in SEFM'24 proceedings) with complete details, metatheoretic results, and examples of other quantum protocols is available at arXiv:2409.11133 [Lanese et al. 2024].

Additional Key Words and Phrases: Multiparty Session Types, Linear Types, Quantum Protocols, Quantum Processes, Quantum Computing

1 Introduction

Quantum protocols involve the exchange of (quantum) information between multiple parties in (quantum) networks, giving rise to complex interaction patterns, interleaved with manipulations of quantum states. This raises the need for tools and techniques to specify, analyse, and verify such protocols. In fact, there does not exist a mainstream formal way to describe quantum protocols, witnessed by the fact that the Quantum Protocol Zoo [*The Quantum Protocol Zoo* 2024], a well-known library of quantum protocols, relies on natural language – hence ambiguous – descriptions, paired with Python implementations.

Existing formalisms for quantum protocols in the literature include imperative languages, such as LanQ [Mlnarık 2006] and QMCLANG [Davidson et al. 2012; Papanikolaou 2009], and process calculi, such as, CQP [Gay and Nagarajan 2005], CCS^{q} [Ardeshir-Larijani et al. 2018] and lqCCS [Ceragioli et al. 2024]. These systems, however, only have a rudimentary type system for values, which does not allow for an abstract description or specification of the quantum protocol, and provides little safety guarantees for communication. The analysis in [Gay and Nagarajan 2005] on the shortcomings of their CQP approach reports: "The proliferation of channels is a consequence of the fact that our type system associates a unique type with each channel. Introducing session types would allow a single channel to be used for the entire protocol".

Following this hint in [Gay and Nagarajan 2005], we propose to use session types to describe quantum protocols. In particular, we start from Multiparty Session Types (MPSTs) [Honda et al. 2016; Hüttel et al. 2016], and propose a quantum extension of them, dubbed Quantum MPSTs (QMPSTs), as a formal session-typed language to describe

Authors' Contact Information: Ivan Lanese, ivan.lanese@unibo.it, Dipartimento di Informatica – Scienza e Ingegneria, Università di Bologna, Bologna, Italy , OLAS Team, INRIA and Université Côte d'Azur, Valbonne, France; Ugo Dal Lago, ugo.dallago@unibo.it, Dipartimento di Informatica – Scienza e Ingegneria, Università di Bologna, Bologna, Italy , OLAS Team, INRIA and Université Côte d'Azur, Valbonne, France; Vikraman Choudhury, vikraman.choudhury@unibo.it, Dipartimento di Informatica – Scienza e Ingegneria, Università di Bologna, Bologna, Italy , OLAS Team, INRIA and Université Côte d'Azur, Valbonne, France.

quantum protocols. QMPSTs provide both an abstract view – the *global type*, describing the expected pattern of interactions, and a concrete view – a multiparty system made of named *quantum processes*. The two views are formally related – *type checking* ensures that processes actually behave as prescribed by the global type, and, the framework of MPSTs also ensures relevant communication properties by construction, such as progress.

In this short essay, we demonstrate the main features of QMPSTs by explaining the quantum teleportation protocol as typed and implemented in our system, and refer to the full paper for details of the system, metatheoretic results, proofs, and additional examples of quantum protocols [Lanese et al. 2024].

2 Quantum Teleportation

The quantum teleportation protocol [Bennett et al. 1993] allows transmitting quantum information from a sender at one location to a receiver at another location, using a *non-quantum* medium. In particular, only classical bits are sent over the communication medium, and an entangled quantum state is used for the actual teleportation of quantum information.

We use our QMPST system to describe the quantum teleportation protocol – using the variant described in [Gay and Nagarajan 2005, Fig. 5] under the name "Quantum teleportation with EPR source". The protocol involves two participants, **Alice** and **Bob**, an EPR **Source** which produces entangled pairs of qubits, and an environment **Env**, where other participants can take input from and send output to. Participant names are also called *roles* in the context of MPSTs, which are used in type signatures for typing processes.

As is standard in MPSTs, we first define a *global type* G, which describes the global (or abstract) view of the protocol, which is the expected pattern of interactions between all participants. The types **qbit** and **bit** denote quantum and classical bits, respectively.

$$
\begin{aligned}
G \triangleq\ & \mathbf{Env}\to\mathbf{Alice}{:}(\mathbf{qbit}).\\
& \quad \mathbf{Source}\to\mathbf{Alice}{:}(\mathbf{qbit})\,.\,\mathbf{Source}\to\mathbf{Bob}{:}(\mathbf{qbit}).\\
& \quad\quad \mathbf{Alice}\to\mathbf{Bob}{:}(\mathbf{bit}^2)\,.\,\mathbf{Bob}\to\mathbf{Env}{:}(\mathbf{qbit}).\\
& \quad\quad\quad \mathbf{end}
\end{aligned}
$$

This global type is meant to be read sequentially, from left to right. First, **Env** sends a **qbit** to **Alice**, or, **Alice** takes a **qbit** from **Env**, which describes both the *send* and *receive* actions of the interaction. Next, **Alice** takes a **qbit** from **Source**, and **Bob** takes a **qbit** from **Source**. These two qubits are meant to be entangled, which will be implemented by the program for the **Source** process. Then, following protocol, **Alice** sends *two classical bits* to **Bob** via a classical channel, which is written as the type $\mathbf{bit}^2$. Finally, **Bob** sends a **qbit** to **Env**, and the protocol ends.

Two important points are worth mentioning. The global type does not specify the actual quantum operations that are performed, but only the abstract view of the system, given by the types of the messages exchanged between participants. The participant **Alice** only exchanges classical bits with **Bob**, and qubits are only exchanged during interactions with **Source** and **Env**, which is expected of the quantum teleportation protocol.

The global type G can be *projected* onto each participant **p** to obtain a *local type* for each participant, $G \upharpoonright \mathbf{p}$, where $\upharpoonright$ is a projection operator (with merging), defined in [Lanese et al. 2024, Def. 2]. Local types are similar to *session types* (in the binary sense), but importantly,

differ in the fact that they are annotated with the roles of the participants. The local type for each participant describes its behaviour in the context of the whole multiparty session. We give the the local type for each participant below, which is calculated by projection.

$$\begin{aligned} G\upharpoonright \mathbf{Alice} &= \mathbf{Env}\&(\mathbf{qbit}).\mathbf{Source}\&(\mathbf{qbit}).\mathbf{Bob}\oplus(\mathbf{bit}^2).\mathbf{end} \\ G\upharpoonright \mathbf{Bob} &= \mathbf{Source}\&(\mathbf{qbit}).\mathbf{Alice}\&(\mathbf{bit}^2).\mathbf{Env}\oplus(\mathbf{qbit}).\mathbf{end} \\ G\upharpoonright \mathbf{Source} &= \mathbf{Alice}\oplus(\mathbf{qbit}).\mathbf{Bob}\oplus(\mathbf{qbit}).\mathbf{end} \\ G\upharpoonright \mathbf{Env} &= \mathbf{Alice}\oplus(\mathbf{qbit}).\mathbf{Bob}\&(\mathbf{qbit}).\mathbf{end} \end{aligned}$$

The local type $G\upharpoonright\mathbf{Alice}$ describes the expected interactions for **Alice**: first, **Alice** receives a **qbit** from **Env**, which is denoted by $\mathbf{Env}\&(\mathbf{qbit})$ – the external choice (or branching, or simply, receive) type (corresponding to the "with" additive in linear logic), with only a single branch of type **qbit** annotated with the sender's role **Env**. Next, **Alice** receives a **qbit** from **Source**, using the same external choice type. Then, **Alice** sends two classical bits to **Bob**, which uses the internal choice (or selection, or simply, send) type (corresponding to the "plus" additive in linear logic), with only a single selection of type $\mathbf{bit}^2$ annotated with the receiver's role **Bob**. Finally, **Alice** ends the protocol. Note that each branching or selection type has a single branch or selection in this example, forcing the protocol to follow a linear sequence of interactions. In general, these can have multiple branches or selections, indexed by sets of labels, allowing for more complex interactions. Similarly, the local types for **Bob**, **Source**, and **Env** are as expected.

Now we write terms defining the actual processes in the quantum teleportation protocol and assign them roles, realizing the actual multiparty system which implements the quantum teleportation protocol. We take inspiration from the syntax in [Ardeshir-Larijani et al. 2018, Fig. 11], but adapt it to include primitives for session-typed communication.

$$\mathcal{M} \triangleq \mathbf{Alice} \triangleright P_A \mid \mathbf{Bob} \triangleright P_B \mid \mathbf{Source} \triangleright P_S \mid \mathbf{Env} \triangleright P_E$$

where:

$$\begin{aligned} P_A &\triangleq \mathbf{Env}\&(w).\mathbf{Source}\&(x).\mathbf{CNot}(w,x).\mathbf{H}(w).\tilde{r} := \mathbf{meas}\,(w,x).\mathbf{Bob}\oplus\langle\tilde{r}\rangle.\mathbf{0} \\ P_B &\triangleq \mathbf{Source}\&(y).\mathbf{Alice}\&(\tilde{r}).\boldsymbol{\sigma}_{\mathbf{r}}(y).\mathbf{Env}\oplus\langle y\rangle.\mathbf{0} \\ P_S &\triangleq \mathbf{new}\,x_s,y_s.\mathbf{H}(x_s).\mathbf{CNot}(x_s,y_s).\mathbf{Alice}\oplus\langle x_s\rangle.\mathbf{Bob}\oplus\langle y_s\rangle.\mathbf{0} \\ P_E &\triangleq \mathbf{Alice}\oplus\langle q\rangle.\mathbf{Bob}\&(y).\mathbf{0}_y \end{aligned}$$

The term language combines standard syntax of session types, with quantum operations on qubits, (and classical operations on bits). We describe each process in turn.

Alice is associated to the process P_A, which takes the desired **qbit**s from **Env** and **Source**, then applies two unitary transformations to them: a controlled not **CNot** (which negates the second **qbit** if the first one is true, just propagates it otherwise – and acts linearly if the states of the qubits are not classical states but a superposition of them), and a Hadamard gate **H** (which turns classical states into a superposition), measures the two resulting **qbit**s obtaining two classical **bit**s in r (the **meas** operator here works on pairs), and sends them to **Bob**.

Bob, or P_B, gets the two **bit**s and uses them to select which of the 4 Pauli transformations $\boldsymbol{\sigma}_{\mathbf{0}},\ldots,\boldsymbol{\sigma}_{\mathbf{3}}$ to apply to y, where $\boldsymbol{\sigma}_{\mathbf{0}}$ is the identity matrix, and $\boldsymbol{\sigma}_{\mathbf{1}}$, $\boldsymbol{\sigma}_{\mathbf{2}}$, $\boldsymbol{\sigma}_{\mathbf{3}}$ are the three 2×2 Pauli matrices.

Source, or P_S, just creates a pair of entangled **qbit**s, and sends its components to, respectively, **Alice** and **Bob**. **Env**, or P_E, just gives the initial **qbit** to **Alice** and gets the result from **Bob**.

This is an untyped multiparty system, and we now typecheck the system, using our typing rules [Lanese et al. 2024, Fig. 4]. We explain how typechecking works informally. To typecheck a multiparty system, one typechecks each process using the local type of its participant. For example, the typing of **Alice**'s process P_A is derived as follows.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{\emptyset \cdot r{:}\mathbf{bit}^2 \vdash r : \mathbf{bit}^2} \quad \dfrac{}{\emptyset \cdot r{:}\mathbf{bit}^2 \vdash \mathbf{0} : \mathbf{end}}}{\emptyset \cdot r{:}\mathbf{bit}^2 \vdash \mathtt{Bob}\oplus\langle r\rangle.\mathbf{0} : \mathtt{Bob}\oplus\{(\mathbf{bit}^2).\mathbf{end}\}}}{\emptyset \cdot z{:}\mathbf{qbit}, x{:}\mathbf{qbit} \vdash r \coloneqq \mathtt{meas}\,(z,x).\mathtt{Bob}\oplus\langle r\rangle.\mathbf{0} : \mathtt{Bob}\oplus\{(\mathbf{bit}^2).\mathbf{end}\}}}{\emptyset \cdot z{:}\mathbf{qbit}, x{:}\mathbf{qbit} \vdash \mathtt{H}(z).r \coloneqq \mathtt{meas}\,(z,x).\mathtt{Bob}\oplus\langle r\rangle.\mathbf{0} : \mathtt{Bob}\oplus\{(\mathbf{bit}^2).\mathbf{end}\}}}{\emptyset \cdot z{:}\mathbf{qbit}, x{:}\mathbf{qbit} \vdash \mathtt{CNot}(z,x).\mathtt{H}(z).r \coloneqq \mathtt{meas}\,(z,x).\mathtt{Bob}\oplus\langle r\rangle.\mathbf{0} : \mathtt{Bob}\oplus\{(\mathbf{bit}^2).\mathbf{end}\}}}{\emptyset \cdot z{:}\mathbf{qbit} \vdash \mathtt{Source}\&(x).\mathtt{CNot}(z,x).\mathtt{H}(z).r \coloneqq \mathtt{meas}\,(z,x).\mathtt{Bob}\oplus\langle r\rangle.\mathbf{0} : \mathtt{Source}\&\{(\mathbf{qbit}).\mathtt{Bob}\oplus\{(\mathbf{bit}^2).\mathbf{end}\}\}}}{\emptyset \cdot \emptyset \vdash \mathtt{Env}\&(z).\mathtt{Source}\&(x).\mathtt{CNot}(z,x).\mathtt{H}(z).r \coloneqq \mathtt{meas}\,(z,x).\mathtt{Bob}\oplus\langle r\rangle.\mathbf{0} : G \upharpoonright \mathtt{Alice}}$$

We read the typing derivation in a bottom-up fashion, maintaining a typing context for classical (non-linear) variables (bound to values at runtime), and quantum (linear) references (bound to references in a quantum register at runtime), in each evident judgement. The process P_A and its local type $G \upharpoonright \mathtt{Alice}$ evolves in tandem (in fact, their transitions are in bisimulation). The received qubits z and x are bound in the linear typing context, and used by the unitary operations $\mathtt{CNot}(z,x)$ and $\mathtt{H}(z)$, which *do not consume* the qubit references. The measurement operation $r \coloneqq \mathtt{meas}\,(z,x)$ measures both qubits, consuming both of them in the typing context, and binds the (classical) variable r to a pair of bits in the (non-linear) typing context. This is used in the selection, when r is sent to **Bob**. Finally, P_A ends the protocol, discarding r. Similarly, processes P_B and P_S are typechecked using their respective local types, in empty contexts. The environment process P_E requires a free qubit q, and is typechecked in context $q{:}\mathbf{qbit}$. Completing all derivations, the following judgement is evident.

$$q{:}\mathbf{qbit} \vdash \mathcal{M} : G$$

Processes and multiparty systems in QMPSTs can be given a dynamic semantics by giving a non-deterministic and probabilistic labelled transition system (GPLTS), over configurations of quantum states, in addition to the usual communication rules of multiparty session typed systems (see [Lanese et al. 2024, Sec. 3]). Typechecking ensures the following important properties for multiparty systems (see [Lanese et al. 2024, Sec. 4]):

- **Subject Reduction:** transitions preserve typability,
- **Session Fidelity:** Global types and multiparty systems evolve in agreement,
- **Progress:** Every well-typed system reduces, or is of global type **end**,
- **Type Safety:** Well-typed systems do not get stuck,
- **Unique Ownership Of Qubits:** Every qubit reference is owned by exactly one process, and
- **Qubit Safety:** Qubit references owned by well-typed systems are the same as their runtime qubit registers.

Acknowledgments

Partially supported by EU Marie-Skłodowska-Curie action ReGraDe-CS, grant № 101106046 (https://doi.org/10.3030/101106046), by French ANR project SmartCloud ANR-23-CE25-0012, by INdAM – GNCS 2024 project MARVEL, code CUP E53C23001670001, and by the Italian Ministry of University and Research under PNRR - M4C2 - I1.4 Project CN00000013 "National Centre for HPC, Big Data and Quantum Computing".

References

Ebrahim Ardeshir-Larijani, Simon J. Gay, and Rajagopal Nagarajan. 2018. "Automated Equivalence Checking of Concurrent Quantum Systems." *ACM Trans. Comput. Log.*, 19, 4, 28:1–28:32. DOI: 10.1145/3231597 (cit. on pp. 1, 3).

Charles H Bennett, Gilles Brassard, Claude Crépeau, Richard Jozsa, Asher Peres, and William K Wootters. 1993. "Teleporting an unknown quantum state via dual classical and Einstein-Podolsky-Rosen channels." *Physical review letters*, 70, 13, 1895 (cit. on p. 2).

Lorenzo Ceragioli, Fabio Gadducci, Giuseppe Lomurno, and Gabriele Tedeschi. 2024. "Quantum Bisimilarity via Barbs and Contexts: Curbing the Power of Non-deterministic Observers." *Proc. ACM Program. Lang.*, 8, POPL, 1269–1297. DOI: 10.1145/3632885 (cit. on p. 1).

Timothy A. S. Davidson, Simon J. Gay, Hynek Mlnarik, Rajagopal Nagarajan, and Nick Papanikolaou. 2012. "Model Checking for Communicating Quantum Processes." *Int. J. Unconv. Comput.*, 8, 1, 73–98 (cit. on p. 1).

Simon J. Gay and Rajagopal Nagarajan. 2005. "Communicating quantum processes." In: *32nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2005*. ACM, 145–157. DOI: 10.1145/1040305.1040318 (cit. on pp. 1, 2).

Kohei Honda, Nobuko Yoshida, and Marco Carbone. 2016. "Multiparty Asynchronous Session Types." *J. ACM*, 63, 1, 9:1–9:67. DOI: 10.1145/2827695 (cit. on p. 1).

Hans Hüttel et al.. 2016. "Foundations of Session Types and Behavioural Contracts." *ACM Comput. Surv.*, 49, 1, 3:1–3:36. DOI: 10.1145/2873052 (cit. on p. 1).

Ivan Lanese, Ugo Dal Lago, and Vikraman Choudhury. 2024. "Towards Quantum Multiparty Session Types." arXiv: 2409.11133. DOI: 10.48550/arXiv.2409.11133 (cit. on pp. 1, 2, 4).

Hynek Mlnarık. 2006. *Introduction to LanQ–an Imperative Quantum Programming Language*. https://lanq.sourceforge.net/doc/introToLanQ.pdf. (2006) (cit. on p. 1).

Nikolaos Papanikolaou. 2009. "Model checking quantum protocols." Ph.D. Dissertation. University of Warwick, Coventry, UK. http://wrap.warwick.ac.uk/2236/ (cit. on p. 1).

The Quantum Protocol Zoo. https://wiki.veriqloud.fr/. (2024) (cit. on p. 1).