DisQ: A Model of Distributed Quantum Processors

Le Chang†, Saitej Yavvari‡, Rance Cleaveland†, Samik Basu‡, Liyi Li‡

†University of Maryland, ‡Iowa State University

Abstract

We present **DisQ**, the first formal model for distributed quantum processors, enabling execution of quantum algorithms across remote QPUs connected by quantum networking. DisQ combines *Chemical Abstract Machine (CHAM)* and *Markov Decision Processes (MDP)* to model both concurrent and distributed quantum behaviors. It includes a distributed quantum programming language and a simulation-based equivalence checker to verify correctness of distributed implementations against their sequential versions.

Introduction

Quantum computing promises major advantages, but current NISQ-era hardware is limited to ∼50 coherent qubits—far fewer than the ∼5,000 needed for large-scale algorithms like Shor's. To overcome this, the next generation of architectures uses Distributed Quantum Computing, where multiple QPUs are connected via *photonic links* to share entangled states.

DisQ is a new formal model for DQC. It enables the *design, analysis, and verification* of distributed quantum programs by combining ideas from process algebra, Markov Decision Processes, and the Chemical Abstract Machine.

DisQ addresses three key challenges:

- Verifying equivalence between sequential and distributed quantum programs.
- Modeling *intra-QPU parallelism* and *inter-QPU communication*.
- Leveraging classical simulation via a *locus-based type system* that preserves physical constraints like no-cloning.

Unlike prior quantum process algebras, DisQ adheres closely to classical foundations, adapting bisimulation to reason about both **nondeterministic** and **quantum probabilistic** behavior, enabling practical development of distributed quantum software.

Background

DisQ models distributed computation using classical foundations to support formal analysis and verification.

Chemical Abstract Machine (CHAM): CHAM captures concurrency through *chemical reactions* between molecules (processes) inside *membranes* (local contexts). Each membrane represents a local computing region (e.g., a QPU), while *airlocks* enable controlled communication across membranes. This abstraction naturally aligns with the physical layout of distributed quantum processors.

Markov Decision Processes (MDPs): An MDP models execution as a sequence of steps, each consisting of: • *nondeterministic choice* (e.g., which membrane performs the next operation), followed by • *probabilistic transition* (e.g., the stochastic result of a quantum operation). This structure allows DisQ to capture both structural distribution and probabilistic dynamics of computation.

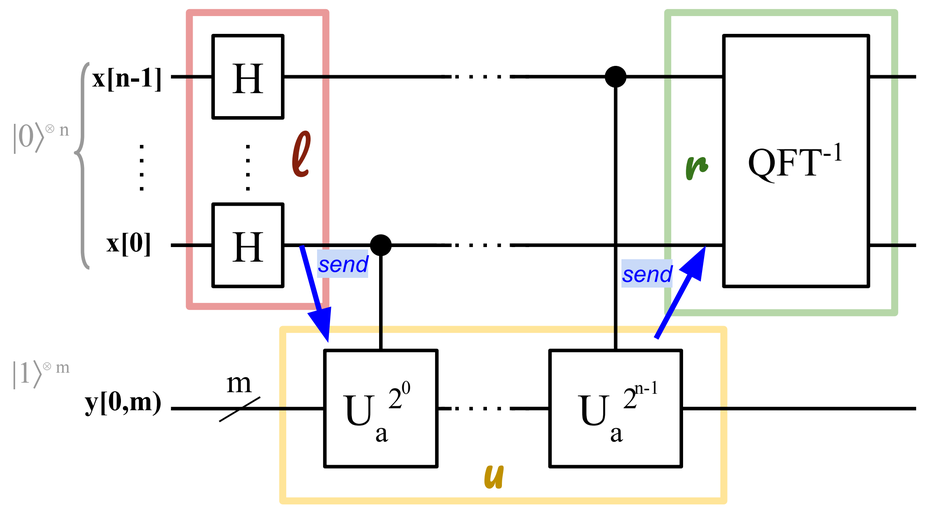
Probabilistic Bisimulation over MDPs: To reason about program equivalence, DisQ uses *probabilistic bisimulation*, defined over the MDP semantics. Two programs are bisimilar if their reachable distributions over behaviors are indistinguishable. This enables verification that a distributed implementation preserves the behavior of its sequential counterpart under both nondeterminism and probability.

Syntax

Unitary Expr	μ		
Bool Expr	B		
Channels	α	$::=$	$a \mid c(n)$
Local Action	U	$::=$	$\partial x(n) \mid \kappa \leftarrow \mu \mid x \leftarrow \mathcal{M}(\kappa)$
Communication Action	D	$::=$	$\partial c(n) \mid \alpha!v \mid \alpha?(y)$
Process	R, T, M, N	$::=$	$\mathbb{0} \mid D.R \mid U.R \mid \texttt{if}\ (B)\ R\ \texttt{else}\ T$
Membrane	P, Q	$::=$	$\{\!\mid \overline{R} \mid\!\}_l \mid R\{\!\mid \overline{T} \mid\!\}_l$

Examples

Shor's Algorithm: Split across 3 QPUs (init, modexp, QFT), connected via teleportation.



One Step Distributed Shor's Algorithm We show the three membranes, l, u, and r, below for performing one step Shor's algorithm computation, assuming that we have a 1-qubit quantum channel $c(1)$ between l and u, as well as $c'(1)$ between u and r.

$\{\!\mid x[i] \leftarrow \mathsf{H}.c(1)!x[i].\mathbb{0} \mid\!\}_l$, $\{\!\mid c(1)?(w).w \uplus y[0,n) \leftarrow \mathsf{CU}(v^{2^i}).c'(1)!w.\mathbb{0} \mid\!\}_u$, $\{\!\mid c'(1)?(q)....\mid\!\}_r$

Conclusion

We introduce **DisQ**, the first formal language model for distributed quantum processors. DisQ enables users to rewrite sequential quantum programs into distributed forms, supporting execution on interconnected QPUs while preserving correctness via a simulation-based equivalence mechanism.

Unlike prior quantum process algebras, which focus on concurrency and assume pre-existing distributed structure, DisQ *constructs* distributed programs and verifies them using **classical probabilistic bisimulation** over a **Markov Decision Process (MDP)** semantics. It incorporates a **locus-based type system** to ensure deadlock-free, physically valid execution.

We validate DisQ through case studies including Shor's algorithm and quantum addition, demonstrating that distributed variants can maintain equivalence with their sequential forms. Looking ahead, we aim to support temporal logics, automate verification, and analyze communication failures in distributed quantum systems.