Analyzing quantum programs using the power of interaction

AGNES VILLANYI, Massachusetts Institute of Technology, USA
CHARLES YUAN, Massachusetts Institute of Technology, USA
CHRIS MCNALLY, Massachusetts Institute of Technology, USA

This work proposes two new approaches to verifying the correctness of quantum programs, based on interactive proofs and foundational results in quantum complexity theory. One approach eliminates the need for making intermediate test measurements by embedding circuit execution into a Hamiltonian, and the other allows classical verification of a quantum program without needing direct access to intermediate states. We also discuss and connect the two meanings of verification within the contexts of quantum program analysis and complexity theory.

1 CHALLENGES IN CHECKING QUANTUM PROGRAMS

Checking the correctness of a quantum program is challenging due to the partial observability of quantum states. Following inherently from the postulates of quantum mechanics, any test checking a program's intermediate state requires probing the quantum system, performing a measurement that may destroy the quantum state and render the remainder of the computation useless.

1.1 Quantum Assertion Schemes

To date, some of the most significant contributions to runtime quantum testing and debugging tools have been variants of quantum assertion schemes, inspired by classical Floyd-Hoare logic [4, 6, 7, 11, 12]. However, current approaches are not sufficiently expressive because they fail to precisely detect all bugs on arbitrary quantum states:

- *Statistical assertions* [4] rely on a combination of the chi-square test and contingency table analysis to identify three types of bugs which the authors classify as *classical, superposition,* and *entangled*. In addition to requiring repeated program executions and therefore being inefficient, this method fails to detect bugs that fall outside of these three classes. The classes themselves are limited as well: for example, it is not possible using these statistical assertions to check whether an intermediate state of a program is the superposition $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$.
- *Circuit-based assertions* [7] inspired by error correction introduce an ancillary circuit to detect bugs. This method mitigates the efficiency problem of the previous approach. The detectable classes of bugs, however, are still ultimately limited to the three categories mentioned above, with some additional 2-qubit and and 3-qubit entangled states.
- *Projection-based assertions* [6] are defined formally using projection-based predicates and quantum logic [1], significantly expanding the set of detectable bugs. However, projections still fail to precisely capture all erroneous states, as correct and buggy states with equivalent supports cannot be distinguished using projective measurements. For example, classifying the mixed states $\rho_1 = \frac{1}{10}|0\rangle\langle 0| + \frac{9}{10}|1\rangle\langle 1|$ and $\rho_2 = \frac{1}{2}|0\rangle\langle 0| + \frac{1}{2}|1\rangle\langle 1|$ as correct and incorrect respectively is impossible using projections alone [11].

Information Asymmetry. The mentioned techniques to check a quantum program using quantum analogues of classical assertions neglect an inherent and unique characteristic of quantum computation: *delegation*, the outsourcing of quantum computation to a more powerful machine than the classical one verifying its correctness. Namely, the goal of a quantum program is to complete a task which is intractable on a classical machine, thereby demonstrating quantum advantage [10].

This information asymmetry between a quantum and classical machine makes it difficult for a classical reasoning tool to observe intermediate states of program execution. As a result, we expect any classical assertion-based verification scheme to fall short of full soundness and completeness.

1.2 Verification of Delegated Quantum Computation

To resolve the challenge introduced by information asymmetry, we look to the well-studied *verification* problem in the setting of complexity theory, which asks whether the outcome of a more powerful machine can be checked for correctness by a less powerful one. Formally, quantum verification is the following task: given a language $L \in$ BQP, an instance x, and a BQP *prover*, the BPP *verifier* must certify whether $x \in L$ using some small number of queries to the prover.

Quantum verification is possible by means of the celebrated Mahadev measurement protocol [8], which relies on interactive arguments and cryptography. The intuition of the protocol is as follows: post-quantum cryptography bounds the BQP prover, giving the BPP verifier leverage over the prover and equilibrating the computational asymmetry. The verifier may then securely request measurements from the prover, such that the verifier can extract a witness to a correct computation, a quantum state ρ, through purely classical communication and computation.

In program analysis, *verification* denotes a method, e.g. an assertion scheme, for checking whether a *program* satisfies a high-level logical predicate defined by the programmer. By contrast, verification in complexity theory denotes a process for checking whether a quantum *server*, i.e. the *prover*, running the program acts honestly and accurately executes the program provided by the classical client. Here, the Mahadev protocol surmounts the fundamental challenge of information asymmetry and enables classically reasoning about a state that is only fully revealed to a quantum process.

1.3 Enabling Classical Reasoning for Quantum States

In this work, we explore how techniques from Mahadev's construction enable a classical client to reason about whether a quantum program satisfies its high-level logical specification. Our aim is to make steps towards a verification scheme which allows programmers to determine whether a program reaches an arbitrary target quantum state, especially one that is difficult to describe classically and cannot be identified using existing assertion schemes.

Contributions. We begin by defining and comparing the meanings of *verification* in the two disciplines of program analysis and complexity theory, which to our knowledge have not previously been formally connected. Upon this basis, we follow with two proposals that use foundational results from complexity theory to verify the correctness of quantum programs:

- We eliminate the need for testing intermediate states via measurements by embedding circuit execution into a Hamiltonian that contains an assertion term imposing constraints on the program state between adjacent time steps. A *correct* program is then one for which the corresponding Hamiltonian has an eigenstate with sufficiently low energy [5].
- We create a quantum analogue of a zero-knowledge static analysis scheme [2], enabling the verifier to test properties of a quantum program without having access to concrete states and overcome the information asymmetry between the verifier and the prover.

Connecting these approaches is the use of well-studied techniques from complexity theory, in particular interactive proofs, to verify quantum programs. While this research remains in progress, we believe that the proposals above are key to surmounting the challenge of information asymmetry, paving the way to more precise and expressive systems for verifying quantum programs.

2 BACKGROUND: PROJECTION-BASED ASSERTIONS

As the method of projection-based assertions [6] was the first to formally define checkable predicates of quantum programs, we recall relevant background about projection-based assertions and discuss some of the shortcomings of this scheme. A more detailed background may be found in [6, 9].

2.1 Projections and Assertions

Projective measurements have the special property that they are non-destructive for states which lie in the subspace of the projector applied during measurement. This state preservation property makes projection-based assertion schemes an attractive tool: quantum states satisfying the predicates are unaffected by their respective assertions. Multiple assertions per execution of the program can therefore be made, introducing efficiency gains over alternative methods [6, 11].

Assertions. In [6], a quantum predicate using projections is defined as the following:

Definition 2.1. Given a finite dimensional Hilbert space $\mathcal{H}$ and a quantum state $\rho \in \mathcal{H}$ with spectral decomposition $\rho = \sum_i \lambda_i |\psi_i\rangle \langle\psi_i|$, let P be a projection operator on $\mathcal{H}$ with subspace $\mathcal{S}_P$. Let the *support* $\mathrm{supp}(\rho)$ be the subspace of ρ spanned by $\{\psi_i\}$ such that $\lambda_i > 0$. We say that a state ρ *satisfies* a predicate P (i.e. $\rho \vDash P$) if $\mathrm{supp}(\rho) \subseteq \mathcal{S}_P$.

An *assertion* is made by applying a projective measurement corresponding to a projector P. For example, asserting that all qubits in an n-qubit quantum state ρ are initialized in the ground state would require the projection $P_0 = (|0\rangle \langle 0|)^{\otimes n}$.

2.2 Incompleteness of Projection-Based Assertions

The fact that projective assertions are limited to capturing erroneus states which are distinguishable from correct ones via a projective measurement greatly diminishes their expressive power. Two quantum states ρ_1, ρ_2 sharing the same support, $\mathrm{supp}(\rho_1) = \mathrm{supp}(\rho_2)$, cannot be distinguished using projective measurements alone. For example, mixed states with non-identical density matrices may still share the same support: e.g. the mixed state consisting of $\{|0\rangle, |+\rangle\}$ with probabilities $\{\frac{1}{2}, \frac{1}{2}\}$ and the mixed state consisting of $\{|0\rangle, |+\rangle\}$ with probabilities $\{\frac{1}{3}, \frac{2}{3}\}$ share the same basis of eigenvectors with non-zero eigenvalues, and therefore have the the same support.

To summarize, all assertion-based quantum verification schemes share a fundamental shortcoming: they fail to fully characterize the desired quantum state and precisely identify all bugs that might occur, leaving holes in their soundness and completeness guarantees.

3 CONNECTING VERIFICATION AND CORRECTNESS

In this section, we define the *verification* of a quantum computation in the context of complexity theory. We discuss the underlying principles of complexity-theoretic protocols that may be transferred to the verification of program correctness in the the discipline of program analysis.

3.1 Quantum Verification and Delegation

Verification is the fundamental problem in complexity theory of, given a language L and a program p acting as a decision procedure for instances of L, checking that the outputs of the program p are correct, i.e. the program p correctly decides the language L.

Interaction. In the complexity class NP, verification is simple: the program may generate in addition to its output a witness for its correctness, such that there definitionally exists a polynomial-time procedure to validate the witness. The complexity class IP extends the class of problems for which programs can be efficiently verified [3], by generalizing NP verification to include adaptive communication between two agents. Interactive proofs allow a computationally bounded agent, the *verifier*, to *delegate* the execution of a program to a more powerful and potentially faulty or malicious machine, the *prover*, and validate that the prover accurately performs the computation.

Delegation. Concretely in the quantum setting, the problem of program delegation is as follows: given a verifier V that may perform BPP computation and a prover P that may perform BQP

computation, construct a protocol such that V can send a classical description of a quantum circuit, C, and an input x, to P, and can trust that the output that P returns for $C(x)$ is correct.

In other terms, verification in complexity theory can be understood as the challenge of efficiently checking the output of an untrusted interpreter executing a fixed, computationally expensive program. Any deviation from the protocol by an adversarial prover should be detected by the verifier. Namely, a (potentially adversarial) prover's claim that it provided a witness to a correct computation should be verifiable in randomized polynomial time.

3.2 Kitaev's Hamiltonian Reduction

In this work, we focus on quantum verification as a special instance of delegated computation in which a circuit C decides whether, for a given promise language $L = L_{yes} \cup L_{no}$, it holds that a particular instance $x \in L_{no}$ or $x \in L_{yes}$. Kitaev showed that it is possible to reduce this formulation of the quantum verification problem to that of finding the lowest energy eigenstate of a 2-local Hamiltonian [5]. The key points of the construction are as follows:

- The execution of $C = \{U_T, \ldots, U_0\}$ on an input $x \in L$ is represented by a *history state*:

$$|\psi_{hist}\rangle = \frac{1}{\sqrt{T+1}} \sum_{t=0}^{T} U_t \ldots U_1 |x\rangle |t\rangle \tag{1}$$

- We can construct a Hamiltonian H_C of circuit C such that *correct* quantum computation is one for which there exists an input x such that $\langle\psi_{hist}| H_C |\psi_{hist}\rangle$ is minimized. Namely, a correct execution of a program is recorded in the Hamiltonian H_C: the existence of a ground state below a certain threshold certifies that there was a correct execution of the program, since otherwise that state would incur energy penalties as defined by the terms of H_C.

Applying this construction, the verification problem can be reduced to showing that a state $|\psi_{hist}\rangle$ with sufficiently low energy statistics exists in the prover's space. This problem remains difficult when the verifier is classical: how can it identify and process a quantum state with purely classical means? Mahadev solved this problem through introducing cryptography as a tool for bounding the prover's abilities, allowing the verifier to trust the prover to make measurements and reproduce the desired measurement statistics for verification.

3.3 Connecting Hoare, Kitaev, and Mahadev

Classical program analysis frameworks such as assertion logics verify that a program meets its high-level logical specification, that is to say the program is written without bugs. By contrast, a protocol for complexity-theoretic verification, such as Mahadev, verifies that the quantum machine running the program meets its specification, that is to say it runs the program as instructed.

The challenge is then to identify how intuition gained from reasoning about the correct behavior of the quantum machine might be lifted to reasoning about the correctness of a program. We identify two critical takeaways for how Kitaev's circuit-to-Hamiltonian construction and the Mahadev measurement protocol apply to checking program correctness:

- Kitaev provides a method for following the execution of a quantum program without needing to make intermediate measurements of the program state.
- Mahadev enables a classical client to make conclusions about the outputs of a program as executed by a quantum machine, without needing to access the internals of the machine.

As described in the next section, these two principles address the main challenge faced by assertion-based reasoning for quantum programs and are transferable to quantum program analysis.

4 INTERACTIVE PROOFS AND QUANTUM PROGRAM ANALYSIS

The goal of this work is to be able to detect arbitrary bugs, which means being able to verify the correctness of arbitrary quantum states, going beyond the guarantees of projective assertions. We propose two potential directions for an interaction-based quantum program analysis scheme.

4.1 Kitaev-Mahadev-Inspired Assertions

Kitaev's circuit-to-Hamiltonian construction gives a method for encoding a quantum circuit into the terms of a 2-local Hamiltonian. Mahadev's measurement protocol gives a method for using this construction to classically verify that the Hamiltonian has a sufficiently low energy eigenstate. In the case of complexity theoretic quantum verification, this can then be used to conclude that some problem instance x is in the *yes*-instance of a promise language $L \in \mathsf{BQP}$, as shown by Kitaev.

Lifting this to program analysis, the challenge is then to use Kitaev's construction and the Mahadev protocol to conclude that a quantum program is bug-free. This could be done by adding an additional term H_a to the Hamiltonian H_C that imposes assertions on the execution of the program between time steps $i, i+1$. The exact form that this assertion term would take, such that the class of detectable bugs surpasses projective assertions, is an open question in this ongoing work.

4.2 Interactive Quantum Abstract Interpretation

Classically, reasoning about programs whose structure is confidential and not accessible by an analysis framework poses unique challenges also relevant to the quantum domain. Namely, in this setting a third-party verifier cannot learn details about the program aside from certain public properties defining its specification. In [2], the authors propose a zero-knowledge abstract interpretation to address this problem: a prover interacts with a verifier in zero knowledge to prove the claim that a program is or is not bug-free, which the prover concludes based on an abstract interpretation supplied to it by the verifer.

For quantum program verification, the zero-knowledge property is unnecessary. However, pairing interactive proofs with proving program correctness would be useful in the quantum setting. This approach would require a quantum analogue of abstract interpretation, with a similar overall structure to [2]: the quantum machine would commit to a program specified by the verifier, then compute an analysis based on an abstract interpretation received from the verifier, followed by a claim about whether or not the program is bug-free. An interactive protocol between the verifier and prover would then validate this claim, thereby proving properties about the quantum program.

5 CONCLUSION

Verifying the correctness of a quantum program is difficult due to the destructive nature of observing program execution. Current methods relying on projection-based assertions only function on a limited set of quantum states. This shortcoming reduces the precision of program reasoning, for example eliminating the ability to distinguish certain correct and incorrect mixed states.

In this work, we identify the information asymmetry between classical and quantum machines as a critical challenge. Complexity-theoretic approaches to quantum verification surmount many of these challenges using interactive proofs. We present the first work to investigate the use of interactive proofs to reason about quantum programs, and propose two interactive approaches that eliminate the need to test and observe intermediate states in quantum program analysis.

A barrier to progress in quantum program verification is the inability to robustly classify bugs as either being classically checkable or fundamentally requiring quantum techniques to check. Identifying the exact boundary of bugs where classical techniques such as projective assertions are insufficient will be the immediate next step in this ongoing work.

REFERENCES

[1] Garrett Birkhoff and John Von Neumann. 1936. The Logic of Quantum Mechanics. *Annals of Mathematics* 37, 4 (1936), 823–843. http://www.jstor.org/stable/1968621

[2] Zhiyong Fang, David Darais, Joseph P. Near, and Yupeng Zhang. 2021. Zero Knowledge Static Program Analysis. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (Virtual Event, Republic of Korea) *(CCS '21)*. Association for Computing Machinery, New York, NY, USA, 2951–2967. https://doi.org/10.1145/3460120.3484795

[3] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. 2015. Delegating Computation: Interactive Proofs for Muggles. *J. ACM* 62, 4, Article 27 (sep 2015), 64 pages. https://doi.org/10.1145/2699436

[4] Yipeng Huang and Margaret Martonosi. 2019. Statistical assertions for validating patterns and finding bugs in quantum programs. In *Proceedings of the 46th International Symposium on Computer Architecture*. ACM. https://doi.org/10.1145/3307650.3322213

[5] A. Yu. Kitaev, A. Shen, and M. N. Vyalyi. 2002. *Classical and Quantum Computation.* American Mathematical Society, USA.

[6] Gushu Li, Li Zhou, Nengkun Yu, Yufei Ding, Mingsheng Ying, and Yuan Xie. 2020. Projection-Based Runtime Assertions for Testing and Debugging Quantum Programs. *Proc. ACM Program. Lang.* 4, OOPSLA, Article 150 (nov 2020), 29 pages. https://doi.org/10.1145/3428218

[7] Ji Liu, Gregory T. Byrd, and Huiyang Zhou. 2020. *Quantum Circuits for Dynamic Runtime Assertions in Quantum Computation.* Association for Computing Machinery, New York, NY, USA, 1017–1030. https://doi.org/10.1145/3373376.3378488

[8] Urmila Mahadev. 2018. Classical Verification of Quantum Computations. https://doi.org/10.48550/ARXIV.1804.01082

[9] Michael A. Nielsen and Isaac L. Chuang. 2011. *Quantum Computation and Quantum Information: 10th Anniversary Edition* (10th ed.). Cambridge University Press, USA.

[10] John Preskill. 2012. Quantum computing and the entanglement frontier. https://doi.org/10.48550/ARXIV.1203.5813

[11] Peng Yan, Hanru Jiang, and Nengkun Yu. 2022. On Incorrectness Logic for Quantum Programs. *Proc. ACM Program. Lang.* 6, OOPSLA1, Article 72 (apr 2022), 28 pages. https://doi.org/10.1145/3527316

[12] Mingsheng Ying. 2012. Floyd–Hoare Logic for Quantum Programs. *ACM Trans. Program. Lang. Syst.* 33, 6, Article 19 (jan 2012), 49 pages. https://doi.org/10.1145/2049706.2049708