Review Article

A Review of Formal Methods in Quantum Circuit Verification

Arun Govindankutty[1]

1. North Dakota State University, United States

Quantum computing uses the laws of quantum mechanics to perform computation. Information is stored in qubits and processed using quantum gates arranged as quantum circuits. As quantum hardware grows in size and complexity, verifying these systems becomes increasingly difficult. Traditional verification methods do not scale well and quickly become computationally infeasible. This creates a strong need for more powerful and scalable verification techniques. Formal methods provide a potential solution to this problem. Techniques such as theorem proving, model checking, and symbolic reasoning offer mathematically rigorous ways to verify correctness, equivalence, and implementation. They also help detect design errors early in the development process. This review examines how formal methods are applied to quantum circuit verification. It focuses on barrier certificates, abstract interpretation, model checking, theorem proving approaches. The review discusses both the theoretical foundations and practical applications of these techniques. Their strengths and limitations are analysed through representative case studies. Finally, the review highlights open challenges and identifies promising directions for future research. An extensive set of references is included to support further study and exploration.

Correspondence: papers@team.qeios.com — Qeios will forward to the authors

1. Introduction

Quantum computing has advanced rapidly in recent years [1][2][3][4][5][6]. Progress has been driven by key contributions from industry and academia. Quantum processors are becoming more realistic, and error-correction techniques are steadily improving [7][8][9][10][11].

Major technology companies have played a key role in this progress. Google, IBM, and Amazon have significantly improved their quantum hardware platforms. They have also experimentally demonstrated

 doi.org/10.32388/H7D715.3

the effectiveness of modern quantum error-correction codes [12][13][14].

Microsoft has taken a different approach by focusing on topological quantum computing. Their work demonstrates architectures that promise more robust qubits and gate operations [15].

Together, these advances help quantum computing transition beyond the Noisy Intermediate Scale Quantum (NISQ) era. They point toward quantum architectures capable of reliable and fault-tolerant computation. The goal of fault-tolerant quantum computing is thus being realized.

As quantum technology advances, it is essential to ensure that quantum computers behave as intended. Correct execution of quantum algorithms is a critical concern. Quantum circuits are the primary abstraction for describing quantum algorithms. Thus, their correctness verification is critical and urgent.

Verifying quantum systems is fundamentally difficult [16]. Most quantum processes cannot be efficiently simulated on classical computers due to its complexity and state space explosion. As a result, many traditional verification techniques do not scale. Quantum circuits also differ from classical circuits in how they can be tested. Empirical testing alone is not sufficient to validate quantum behaviour. Such testing provides limited coverage and weak correctness guarantees. For these reasons, alternative verification approaches are necessary. These approaches must be systematic and mathematically rigorous [17].

Formal verification provides a rigorous way to validate complex systems. It uses mathematical reasoning and proofs to ensure design correctness. Unlike traditional testing, formal verification explores the entire design space. It does not rely on sampled behaviours alone. This makes it possible to detect subtle corner case bugs that testing may miss. Because of these strong guarantees, formal verification is widely used in safety and reliability critical domains. These include software systems, hardware design, and VLSI design verification [18][19][20].

Formal methods are now being applied to quantum circuits [16][17]. These methods are designed to handle features unique to quantum computation. These include superposition, entanglement, interference and noise. By addressing these challenges, formal verification provides strong guarantees for quantum circuit verification.

This work presents a concise review of formal methods for quantum circuit verification. It covers both the theoretical foundations and practical applications of these techniques. The review examines several key approaches. These include barrier certificates, abstract interpretation, model checking, theorem proving, and hybrid methods that combine multiple techniques. It compares their strengths and

limitations in a systematic manner. Finally, it highlights promising research directions for addressing the verification challenges posed by increasingly complex quantum systems.

2. Background

This section introduces the background needed to understand qubits and quantum circuits. It also presents the formal verification techniques in general used in verification.

2.1. Qubits & Quantum Circuits

Quantum circuits are fundamentally different from classical circuits. This difference arises from the unique properties of quantum information. Classical bits can take only one of two definite values, 0 or 1. In contrast, quantum bits, or qubits, can exist in superpositions of basis states. Qubit states are described using complex valued probability amplitudes [21][22].

A single-qubit state is written as

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

where $|\psi\rangle$ represents the quantum state. The vectors $|0\rangle$ and $|1\rangle$ form the computational basis. The coefficients α and β are complex numbers. The squared magnitudes of these coefficients determine measurement outcomes. Specifically, $|\alpha|^2$ and $|\beta|^2$ give the probabilities of measuring $|0\rangle$ and $|1\rangle$. These probabilities must sum to one, which leads to the normalization condition

$$|\alpha|^2 + |\beta|^2 = 1.$$

The computational basis vectors are defined as

$$|0\rangle = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, |1\rangle = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Similarly, the state of an n-qubit quantum system, denoted by $|\psi\rangle$, is described using 2^n complex probability amplitudes. The state can be written as

$$|\psi\rangle = \sum_{x \in \{0,1\}^n} c_x |x\rangle.$$

Each basis state $|x\rangle$ has an associated amplitude c_x. The probability of measuring the system in state x is given by

$$p_x = |c_x|^2.$$

All measurement probabilities must sum to one. This normalization condition is expressed as

$$\sum_{x \in \{0,1\}^n} |c_x|^2 = 1.$$

Qubits can also become entangled. Entanglement creates non-classical correlations that cannot be reproduced by any classical system. These properties allow quantum algorithms to outperform classical algorithms for certain tasks. At the same time, they introduce significant challenges for verifying quantum circuits.

A quantum circuit is represented as a sequence of quantum gates acting on a set of qubits [21]. Each gate applies a unitary transformation to the qubits. Common single-qubit gates include the Pauli operators (X, Y, Z), the Hadamard gate, and various phase gates. Multi-qubit gates, such as the CNOT (controlled-NOT) gate, can create entanglement between qubits.

Verification of quantum circuits is challenging because the state space grows exponentially with the number of qubits. An n-qubit system occupies a Hilbert space of dimension 2^n, making classical simulation not feasible for large circuits.

2.2. Formal Verification Principles

As introduced in Section 1, formal verification involves mathematically proving or disproving that a system meets a formal specification. Unlike testing, which checks only a finite set of inputs or execution paths, formal methods reason about the entire state space and all possible evolutions of a system. This allows them to guarantee correctness, safety, or compliance with specifications for every possible input, timing, or quantum state and not just the cases tested. In other words, formal methods provide mathematical certainty, while testing can only show that errors are absent in the scenarios that were actually checked.

Formal verification approaches can be grouped into three main categories:

1. **Deductive verification:** Proving correctness properties using mathematical proof systems.
2. **Model checking:** Verifying finite state systems against temporal logic specifications.
3. **Abstract interpretation:** Analysing system behaviour using sound semantic approximations.

In quantum systems, formal verification faces unique challenges. These arise from superposition, entanglement, measurement, and decoherence. Classical methods must be carefully adapted to be

employed for quantum circuit verification. They need to handle the quantum effects while staying computationally feasible.

3. Barrier Certificates

This section introduces barrier certificates, a key technique employed in the formal verification of quantum circuits.

3.1. Foundations

Barrier certificates are an effective method for verifying safety properties of dynamical systems [23]. They guarantee that system trajectories never reach unsafe or undesirable states. More recently, barrier certificates have been extended to the verification of quantum circuits [24][25][26][23]. A barrier certificate captures all possible executions of a system using a single real-valued function. This function separates safe regions from unsafe regions of the state space.

Formally, a barrier certificate for a quantum system is defined as a function $B : \mathbb{X} \to \mathbb{R}$. It must satisfy three main conditions:

1. **Initial state condition:** $B(x) \leq 0$ for all initial states $x \in \mathbb{X}_0$.
2. **Unsafe state condition:** $B(x) > 0$ for all unsafe states $x \in \mathbb{X}_u$.
3. **Decrement condition:** $B(x') - B(x) \leq 0$ for all $x \in \mathbb{X}$ and all $x' \in f(x)$.

Here, $\mathbb{X}$ is the state space. The set $\mathbb{X}_0 \subseteq \mathbb{X}$ contains all initial states. The set $\mathbb{X}_u \subseteq \mathbb{X}$ contains all unsafe states. The function $f : \mathbb{X} \Rightarrow \mathbb{X}$ is a set-valued transition map that defines the system dynamics. In addition, the function B must not increase under the circuit dynamics. These conditions ensure that trajectories starting from any initial state never enter the unsafe region at any given time step. The existence of such a barrier certificate guarantees safety over an *unbounded time horizon*. This approach avoids explicit state or time enumeration and therefore mitigates the state-space explosion problem.

Quantum circuits can be viewed as dynamical systems, which makes barrier certificates well-suited for their verification. An n-qubit state $|\psi\rangle$ resides in $\mathbb{C}^{2^n}$ and satisfies the normalization condition $\sum_i |\psi_i|^2 = 1$. Each quantum gate acts as a unitary linear transformation on the state. Thus, a quantum circuit can be modelled as a discrete-time system $S = (Z, Z_0, F, f)$. Here $Z \subset \mathbb{C}^{2^n}$ is the continuous state space, Z_0 is the set of initial states, $F = \{U_1, U_2 \ldots, U_m\}$ is a finite set of unitary transitions, and f selects which U_i applies at each step [26]. Safety and correctness properties can be expressed as

polynomial or semi-algebraic constraints on the amplitudes. These include properties like bounds on measurement probabilities, and preservation of logical subspaces etcetera. This algebraic structure naturally aligns with polynomial barrier certificates and sum-of-squares reasoning.

3.2. Scenario-based Approach

A key advancement in barrier certificate synthesis for quantum circuits is the scenario-based approach [27][28][24]. This method uses sampling to construct barrier certificates over both finite and infinite time horizons. It explicitly accounts for uncertainties in initial states and system dynamics. These features make it well suited for noisy quantum systems, where exact knowledge of all parameters is often impossible.

The scenario-based approach converts the verification problem into a convex optimization problem using sampled constraints. This makes it possible to efficiently compute a barrier certificate that satisfies the safety conditions with high probability. The method has several advantages for quantum circuit verification:

1. Supports the continuous state spaces found in quantum systems.
2. Handles uncertain dynamics caused by quantum noise and device imperfections.
3. Provides probabilistic guarantees of correctness.
4. Works for both finite and infinite time horizons.

3.3. Application

Barrier certificates have been successfully used to verify a variety of quantum circuits. These include Grover's search algorithm and other quantum oracles. Researchers have tested different classes of barrier certificates, such as polynomial, exponential, and rational functions. This helps determine the most effective type for each application. Table 1 summarizes these barrier certificate types. It highlights their strengths, limitations, and typical use cases.

Barrier Certificate Type	Strengths	Limitations	Ideal Use Cases
Polynomial	Efficient synthesis, good scalability	Limited expressiveness	Linear and mildly nonlinear systems
Exponential	Handles exponential dynamics	Numerical stability issues	Systems with exponential convergence
Rational	High expressiveness	Complex optimization	Highly nonlinear systems
Scenario-based	Handles uncertainty, probabilistic guarantees	Sampling may miss rare cases	Noisy quantum systems

Table 1. Barrier Certificates and Their Applications

To verify a quantum circuit using barrier certificates, the circuit is first encoded as a discrete-time complex dynamical system. Polynomial descriptions of the initial and unsafe sets are also defined. Next, a barrier function template is chosen. This is usually a real-valued polynomial in the real and imaginary parts of the quantum state. Verification conditions are then imposed to ensure that the initial and unsafe regions are separated and that the barrier does not increase along any gate induced transition. These conditions can be solved using Hermitian sum-of-squares optimization to provide exact guarantees [26]. Scenario-based optimization combined with SMT solving can improve scalability [24]. Once the barrier certificate is synthesized and formally validated, it provides a sound and global correctness proof for the quantum circuit.

Case studies show that the choice of barrier function greatly affects both the efficiency and effectiveness of verification(Table 1). For many quantum circuits, polynomial barrier certificates offer a good balance between expressiveness and computational efficiency. More complex barrier functions may be needed for circuits with highly non-linear dynamics.

4. Abstract Interpretation

This section provides an overview of how abstract interpretation is applied to the formal verification of quantum circuits. Abstract interpretation is a static analysis technique that verifies program properties by computing sound over-approximations of program semantics in an abstract domain [29]. In quantum circuit verification, abstract interpretation is useful because it avoids explicitly enumerating the exponentially large Hilbert space. Instead of tracking exact quantum states, it reasons over abstract representations, such as projection subspaces, stabilizers, or interval bounds on amplitudes [30]. Quantum circuits consist of structured, sequential unitary transformations. This make them well suited to abstract interpretations. This allows abstract transformers to be defined compositionally for each quantum gate. Verification using abstract interpretation typically involves defining an abstract domain and a mapping to the original state space, implementing abstract gate semantics that conservatively approximate unitary evolution, and checking safety or correctness properties directly on the abstract states. This approach guarantees soundness while achieving polynomial time scalability [30].

4.1. Semantic Framework

Abstract interpretation provides a theoretical framework for approximating the semantics of computational systems. It enables static analysis of program properties [30][31][32][33][34][35]. Recent work has extended this approach to variational quantum circuits (VQCs) [36][37]. VQCs form the basis of many quantum machine learning algorithms. Similar to classical deep neural networks, VQCs are vulnerable to adversarial inputs. Small deviations or perturbations in the input can cause incorrect outputs or predictions.

Assolini et al. [36] propose a semantic framework based on abstract interpretation for verifying VQCs. Their approach explicitly handles quantum specific properties, such as state normalization. Normalization introduces dependencies between variables, which complicates traditional verification methods. This framework provides a formal way to define the verification problem for VQCs. It also offers tools to analyse the computational complexity of the verification process.

4.2. Interval-based Reachability

Interval-based reachability analysis is a key technique in the abstract interpretation of quantum circuits. It computes over approximations of the reachable states at each layer of the quantum circuit [36][38][39]

[40]. This method propagates interval bounds through the layers of circuit. In doing so, it provides formal guarantees about circuit behaviour. However, quantum effects such as superposition and entanglement create dependencies between variables. These dependencies make interval analysis more challenging.

To address these challenges, researchers have developed enhanced abstraction techniques. These techniques explicitly track dependencies between variables. However, this added precision comes with higher computational cost [41]. Finding the right balance between precision and efficiency remains an active research problem in quantum abstract interpretation.

4.3. Verification of Robustness Properties

Abstract interpretation is quite effective for verifying robustness properties of VQCs. It can be used to study how adverse perturbations affect circuit behaviour. These techniques analyse how small changes in the input state influence the final measurement probabilities. In doing so, they provide formal robustness certificates. These certificates are similar to those used for classical neural networks [42][43][44][45][46]. The verification process typically follows four main steps:

1. Defining a perturbation model for the input states.
2. Propagating these perturbations through the quantum circuit using abstract domains.
3. Computing bounds on the output measurement probabilities.
4. Checking that classification decisions remain stable within the allowed perturbation range.

This methodology has been tested on standard verification benchmarks. The results show its potential for certifying the reliability of quantum machine learning models. This makes it specifically important for safety critical applications.

5. Other Formal Methods

This section highlights additional formal methods used in the verification of quantum circuits.

5.1. Model Checking

Model checking is a formal verification technique that systematically explores all possible states of a system. In quantum circuit verification, it checks whether a circuit satisfies specified temporal logic properties. Model checking is valuable because it provides complete, counterexample-driven verification for finite or symbolically representable quantum systems [47][48]. Quantum circuits can be modelled as

finite-state transition systems or quantum Markov chains. This is done by discretizing their evolution or focusing on basis states and measurement outcomes. This structure allows the use of temporal logics, including probabilistic extensions, to specify correctness, safety, or termination properties. This approach has been the subject of extensive research [49][50][51][52][53][54][55][56][57].

For quantum circuits, model checking usually follows three main steps:

1. Encoding the quantum circuit as a finite-state model.
2. Expressing the desired properties using suitable temporal logics.
3. Checking these properties against the encoded model.

Early work in this area verified quantum circuits by mapping them to quantum Markov chains. These models were then analysed using model checking [48]. Other approaches extended probabilistic model checking. They account for the inherent randomness of quantum measurements and decoherence [58].

In spite of having superior theoretical power, model checking faces serious scalability issues in quantum systems. The state space grows exponentially with the increase of number of qubits. This makes exhaustive exploration of state space impractical. To address this state explosion problem, symbolic model checking techniques have been developed. These methods use binary decision diagrams (BDDs) and related compression strategies [59][60][61][62][63][64][65]. They have achieved varying levels of success in improving scalability.

5.2. Theorem Proving

Theorem proving is a deductive method for verifying quantum circuits. In this approach, correctness properties are established through formal logical proofs checked by a proof assistant. It builds rigorous mathematical proofs of correctness, providing the strongest possible guarantees because every proof step is mechanically verified within a sound logical framework [66].

Quantum circuits have rich algebraic and mathematical structures, such as unitary operators, linearity, and reversibility. These properties make them well suited for formal reasoning in higher-order logic or dependent type theory. Theorem-proving methods typically formalize the semantics of quantum gates and circuits within a proof assistant. They then prove equivalence, invariants, or correctness theorems with respect to a specification. This approach has been implemented in proof assistants like Coq, Isabelle/HOL, and Lean. These tools are often enhanced with specialized libraries for quantum computation [67][68][69][70][71].

Rotational abstractions for verification of quantum Fourier transform circuits [72], and Superposition-Based Abstractions for Quantum Data Encoding Verification [73] elucidate a symbolic deductive abstraction approach to quantum circuit verification using SMT solvers. The Giallar tool, uses SMT solvers to verify quantum circuit compiler passes. It ensures that quantum semantics are preserved at each pass [74]. Theorem proving provides very high assurance of correctness. However, it requires significant expertise and manual effort. This makes it less suitable for rapid iteration in quantum circuit design workflows.

5.3. Runtime Verification

Runtime verification monitors the execution of a quantum circuit to check whether it satisfies specified properties. Although it does not provide full formal guarantees, it can detect property violations during testing or actual operation. This makes runtime verification practical for near term quantum applications [75][76][77][78][79][74][80][81]. The runtime verification process typically involves three main steps:

1. Instrumenting the quantum circuit with extra measurement operations.
2. Defining assertion like properties for specific circuit states.
3. Performing statistical testing of these properties during execution.

Runtime verification is particularly useful for validating specific executions on quantum hardware. It serves as a complement to formal methods that analyse entire quantum circuit designs.

6. Applications

This section presents case studies that demonstrate the practical application of quantum circuit verification.

6.1. Quantum Error Correction

Quantum error correction (QEC) is a key area where formal verification methods are employed. Fault tolerant quantum computation depends on the correct operation of QEC circuits [82][83][84]. Formal methods have been used to verify different aspects of QEC implementations [85][86][87][88][89], including:

1. The correctness of stabilizer measurements.
2. Fault tolerance thresholds.

3. The implementation of logical operations.

For example, barrier certificates effectively ensure that errors stay within correctable regions of the state space. Model checking is used to verify the sequential behaviour of QEC protocols under various fault models.

6.2. Compiler Verification

Quantum circuits are usually written in high-level programming languages and then compiled into hardware-specific instructions. This makes verifying the correctness of the compilation process both critical and essential [35][75][90][77][79][91][92][93]. The Giallar tool [74] uses SMT solvers to check that each compiler pass preserves the semantic integrity of the quantum circuit. This verification process usually involves three main steps:

1. Translating quantum circuits into logical formulae.
2. Checking equivalence between the original and compiled circuits.
3. Verifying that the optimization rules applied during compilation are correct.

Compiler verification is especially important for quantum applications where stakes are high. Even small compilation errors in such settings could lead to catastrophic failures.

Domain	Verification Methods	Challenges	Tools
Quantum Error Correction	Barrier certificates, Model checking	Complexity of feedback control	QECVerifier, QuaVer
Quantum Compilers	Theorem proving, SMT solvers	Semantic preservation across layers	Giallar, Quartz
Quantum Algorithms	Abstract interpretation, Barrier certificates	Handling exponential state spaces	QVVerify, CertiQ
Quantum Hardware	Model checking, Runtime verification	Modeling physical imperfections	HQVer, PulseVerifier

Table 2. Applications of Formal Verification in Quantum Computing

6.3. Quantum Algorithms

Formal methods have been used to verify the correctness of various quantum algorithms [94][95]. These include Grover's search algorithm [96], quantum phase estimation [97][98], and quantum approximate optimization algorithms (QAOA) [99] among others.

Each algorithm presents its own verification challenges:

1. Grover's algorithm: Verification requires proving convergence and establishing bounds on the success probability.
2. Quantum phase estimation: Verification involves ensuring precision guarantees under different noise models.
3. QAOA: Verification focuses on approximation ratios and convergence properties.

These verifications often use a combination of formal techniques. For example, barrier certificates can be used to establish safety properties, while theorem proving ensures functional correctness. The benefits of each method can thus be exploited depending on the application.

The choice of verification method generally depends on the type of problem. Barrier certificates or abstract interpretation are suitable for systems with large or continuous state spaces. Model checking works well when exhaustive exploration of discrete transitions is feasible. Theorem proving or SMT-based methods are used when semantic correctness or equivalence must be guaranteed across transformations. Runtime verification is useful for detecting errors during actual execution on hardware. Hybrid approaches can combine these techniques to balance scalability, automation, and formal guarantees. This allows the verification method to be adapted to the structure and requirements of the quantum system.

As shown in Table 2, verification methods are chosen based on the characteristics and challenges of each quantum computing domain. For quantum error correction, barrier certificates and model checking are particularly suitable. They rigorously guarantee the stability and correctness of feedback controlled stabilizer circuits across all possible error trajectories. Barrier certificates handle continuous state-space dynamics, while model checking systematically explores discrete syndrome transitions. This combination makes them ideal for the safety critical nature of error correction.

In quantum compilers, theorem proving and SMT solvers focus on semantic preservation across multiple compilation layers. The algebraic and logical structure of quantum gates allows formal deduction. It helps

to verify that transformations and optimizations preserve program semantics. This ensures correctness for all possible input states.

Quantum algorithms, which often operate on exponentially large state spaces, benefit from abstract interpretation and barrier certificates. Abstract interpretation approximates sets of quantum states to efficiently check invariants. Barrier certificates provide mathematically rigorous safety guarantees without enumerating all states. Both approaches enable scalable verification.

For quantum hardware, model checking and runtime verification are effective. They capture both the discrete and continuous aspects of physical imperfections. Model checking systematically explores control sequences and configurations. Runtime verification monitors live signals to detect deviations from expected behaviour.

7. Challenges and Limitations

This section presents a critical assessment of the challenges and limitations in current quantum circuit verification methods.

7.1. Scalability

A major challenge in quantum circuit verification is the exponential growth of the state space as the number of qubits increases linearly. Even though classical hardware continues to improve, the inherent complexity of representing exact quantum states limits the scalability of all verification methods. Current approaches try to address this challenge using:

1. Abstraction and approximation techniques that trade completeness for scalability.
2. Modular verification, which breaks large circuits into smaller components.
3. Symbolic methods that represent sets of quantum states compactly.

Despite these strategies, verifying circuits with many qubits remains difficult. This underscores the need for further research into scalable verification techniques and methodologies.

7.2. Quantum-specific Capabilities

Quantum phenomena such as entanglement, superposition, and measurement create verification challenges that do not have classical equivalent. These effects produce complex correlations between qubits that are hard to abstract or approximate without losing important information. Measurement is

especially difficult because it collapses the quantum state and is inherently non-unitary, making continuous verification more complicated. Approaches to address these challenges include:

1. Creating specialized abstract domains to capture quantum correlations.
2. Designing verification techniques that are aware of measurements.
3. Using relational logics to represent and reason about entanglement.

7.3. Tool Support

Many tools have been developed for quantum circuit verification. The ecosystem is still less mature than that for classical software verification. Most tools require significant expertise to use. They are often tailored to specific verification methods or quantum programming languages. Improving tool support will require:

1. Standardizing interfaces between verification tools and quantum programming frameworks.
2. Creating user friendly interfaces for specifying verification properties.
3. Establishing elaborate benchmark suites to evaluate and compare verification tools.
4. Automating the choice of the most suitable verification method for a given circuit.

8. Future Directions

Here, several potential frontiers and directions for future research in quantum circuit verification are outlined, as identified by the reviewer.

8.1. Hybrid Verification Methods

Future verification frameworks are likely to combine multiple techniques into hybrid approaches. These approaches can leverage the strengths of each method. For example, abstract interpretation can quickly identify potential problem areas. These areas can then be examined in detail using more precise methods, such as theorem proving or model checking. Promising hybrid combinations include:

1. Using barrier certificates together with abstract interpretation for safety verification.
2. Combining theorem proving with model checking to verify both functional and temporal properties.
3. Augmenting runtime verification with formal methods to provide practical assurance.

8.2. Machine Learning Assisted Verification

Machine learning offers promising ways to enhance formal verification of quantum circuits. Potential applications include:

1. Learning barrier certificates directly from simulation data.
2. Predicting which circuit components are hard to verify, to focus verification efforts.
3. Guiding abstract interpretation using learned heuristics.
4. Speeding up model checking with learned representations of the state space.

These approaches could greatly improve the scalability and automation of quantum circuit verification while maintaining formal guarantees.

8.3. Verifying Fault-Tolerant Quantum Computing

As quantum computing moves toward fault-tolerant operation, new verification challenges and opportunities emerge. Future research directions include:

1. Verifying quantum error correction protocols under realistic noise models.
2. Developing verification techniques for distributed quantum systems.
3. Establishing certification frameworks for quantum hardware components.
4. Creating standards for quantum software verification.

Advances in these areas will be essential for building reliable and trustworthy quantum computing systems for critical applications.

9. Conclusion

Formal methods for quantum circuit verification have made significant progress in recent years. They have evolved from theoretical frameworks to practical tools that can be applied to real quantum circuits. This review studies the current landscape of these methods, including barrier certificates, abstract interpretation, model checking, and theorem proving. Each method offers unique advantages and is suited to different verification scenarios.

Barrier certificates provide a strong method for safety verification. When combined with scenario-based approaches, they can handle uncertainties in initial states and system dynamics. Abstract interpretation gives a semantic framework for analysing variational quantum circuits and verifying their robustness.

Model checking allows exhaustive verification of temporal properties. Theorem proving offers the highest level of assurance through rigorous mathematical proofs.

In spite of these advances, major challenges remain. Scaling verification to larger quantum systems and handling quantum specific features like entanglement, superposition and measurement are still difficult. Future research should focus on developing hybrid approaches that combine multiple verification techniques. It should also explore using machine learning to improve scalability and address the verification needs of fault-tolerant quantum computing.

As quantum computing moves closer to practical applications, formal verification methods will be increasingly important for ensuring the reliability and correctness of quantum software and hardware. Developing robust verification tools and methods will be key to building trust in quantum computing systems and unlocking their full potential for solving challenging and complex computational problems.

References

1. ^ *Hassija V, Chamola V, Goyal A, Kanhere SS, Guizani N (2020). "Forthcoming Applications of Quantum Computing: Peeking into the Future." IET Quantum Commun. **1**(2):35–41. doi:10.1049/iet-qtc.2020.0026.*
2. ^ *Jose P, et al. (2024). "Enhanced QSVM with Elitist Non-Dominated Sorting Genetic Optimisation Algorithm for Breast Cancer Diagnosis." IET Quantum Commun. doi:10.1049/qtc2.12113.*
3. ^ *Arun G, Mishra V (2014). "A Review on Quantum Computing and Communication." In: 2014 2nd International Conference on Emerging Technology Trends in Electronics, Communication and Networking, pp. 1–5. doi:10.1109/ET2ECN.2014.7044953.*
4. ^ *Brooks M (2019). "Beyond Quantum Supremacy: The Hunt for Useful Quantum Computers." Nature. **574**(7776):19–21. doi:10.1038/d41586-019-02936-3.*
5. ^ *Chen S (2019). "Quantum Computing Scientists: Give Them Lemons, They'll Make Lemonade." APS News. https://www.aps.org/apsnews/2019/05/quantum-computing-lemons-lemonade.*
6. ^ *Preskill J (2018). "Quantum Computing in the NISQ Era and Beyond." Quantum. **2**:79. doi:10.22331/q-2018-08-06-79.*
7. ^ *Porter J (2021). "Google Wants to Build a Useful Quantum Computer by 2029." The Verge. https://www.theverge.com/2021/5/19/22443453/google-quantum-computer-2029-decade-commercial-useful-qubits-quantum-transistor.*

8. ^ Gambetta J (2022). "Expanding the IBM Quantum Roadmap to Anticipate the Future of Quantum-Centric Supercomputing." IBM Research. https://research.ibm.com/blog/ibm-quantum-roadmap-2025.
9. ^ Krewell K, Research T (2022). "The Next Generation Of IBM Quantum Computers." Forbes. https://www.forbes.com/sites/tiriasresearch/2022/06/22/the-next-generation-of-ibm-quantum-computers/.
10. ^ Jay G, Ismael F, Karl W (2021). "IBM's Roadmap for Building an Open Quantum Software Ecosystem." IBM Research. https://research.ibm.com/blog/quantum-development-roadmap.
11. ^ Hsu J (2018). "CES 2018: Intel's 49-Qubit Chip Shoots for Quantum Supremacy." IEEE Spectrum. https://spectrum.ieee.org/intels-49qubit-chip-aims-for-quantum-supremacy.
12. ^ Acharya R, et al. (2025). "Quantum Error Correction Below the Surface Code Threshold." Nature. **638**(8052):920–926. doi:10.1038/s41586-024-08449-y.
13. ^ Bravyi S, Cross AW, Gambetta JM, Maslov D, Rall P, Yoder TJ (2024). "High-Threshold and Low-Overhead Fault-Tolerant Quantum Memory." Nature. **627**(8005):778–782. doi:10.1038/s41586-024-07107-7.
14. ^ Putterman H, et al. (2025). "Hardware-Efficient Quantum Error Correction Via Concatenated Bosonic Qubits." Nature. **638**(8052):927–934. doi:10.1038/s41586-025-08642-7.
15. ^ Aghaee M, et al. (2025). "Interferometric Single-Shot Parity Measurement in InAs–Al Hybrid Devices." Nature. **638**(8051):651–655. doi:10.1038/s41586-024-08445-2.
16. a, b Quist A-J, Mei J, Coopmans T, Laarman A (2025). "Advancing Quantum Computing with Formal Methods." In: Formal Methods. Cham: Springer Nature Switzerland, pp. 420–446.
17. a, b Chareton C, et al. (2022). "Formal Methods for Quantum Programs: A Survey." arXiv preprint arXiv:2109.06493.
18. ^ Hasan O, Tahar S (2015). "Formal Verification Methods." In: Encyclopedia of Information Science and Technology, Third Edition. IGI Global, pp. 7162–7170.
19. ^ Gupta A (1992). "Formal Hardware Verification Methods: A Survey." Formal Methods Syst Des. **1**:151–238.
20. ^ Schubert T, Seligman MVA KKE (2015). Formal Verification. Elsevier.
21. a, b Nielsen MA, Chuang IL (2011). Quantum Computation and Quantum Information: 10th Anniversary Edition. 10th ed. USA: Cambridge University Press.
22. ^ Bernhardt C (2020). Quantum Computing for Everyone. MIT Press.
23. a, b Prajna S, Jadbabaie A, Pappas GJ (2007). "A Framework for Worst-Case and Stochastic Safety Verification Using Barrier Certificates." IEEE Trans Autom Control. **52**(8):1415–1428.

24. ^a ^b ^cHu S, Lopata V, Soudjani S, Zuliani P (2025). "Verification of Quantum Circuits Through Barrier Certific ates Using a Scenario Approach." In: 2025 IEEE International Conference on Quantum Software (QSW), pp. 151–161.

25. ^Lewis M, Soudjani S, Zuliani P (2024). "Verification of Quantum Circuits Through Discrete-Time Barrier C ertificates." arXiv preprint arXiv:2408.07591.

26. ^a ^b ^cLewis M, Zuliani P, Soudjani S (2023). "Verification of Quantum Systems Using Barrier Certificates." In: International Conference on Quantitative Evaluation of Systems. Springer, pp. 346–362.

27. ^Murali V, Trivedi A, Zamani M (2022). "A Scenario Approach for Synthesizing K-Inductive Barrier Certifica tes." IEEE Control Syst Lett. **6**:3247–3252.

28. ^Akella P, Ames AD (2022). "A Barrier-Based Scenario Approach to Verifying Safety-Critical Systems." IEEE Robot Autom Lett. **7**(4):11062–11069.

29. ^Cousot P, Cousot R (1977). "Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints." In: Proceedings of the 4th ACM SIGACT-SIGPLAN POPL, p pp. 238–252.

30. ^a ^b ^cYu N, Palsberg J (2021). "Quantum Abstract Interpretation." In: Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, pp. 542–558.

31. ^Ying M, Zhang Z (2024). "Verification of Recursively Defined Quantum Circuits." arXiv preprint arXiv:2404. 05934.

32. ^Rizvi SMA, et al. (2026). "Controlled Quantum Semantic Communication for Industrial CPS Networks." IE EE Trans Netw Sci Eng. **13**:996–1009.

33. ^Chen Y-F, et al. (2025). "An Automata-Based Framework for Verification and Bug Hunting in Quantum Cir cuits." Commun ACM. **68**(6):85–93.

34. ^Li L, et al. (2024). "Qafny: A Quantum-Program Verifier." arXiv preprint arXiv:2211.06411.

35. ^a ^bShi Y, et al. (2020). "CertiQ: A Mostly-Automated Verification of a Realistic Quantum Compiler." arXiv pre print arXiv:1908.08963.

36. ^a ^b ^cAssolini N, Marzari L, Mastroeni I, di Pierro A (2025). "Formal Verification of Variational Quantum Circ uits." arXiv preprint arXiv:2507.10635.

37. ^Peham T, Burgholzer L, Wille R (2023). "Equivalence Checking of Parameterized Quantum Circuits: Verifyi ng the Compilation of Variational Quantum Algorithms." In: Proceedings of the 28th Asia and South Pacific Design Automation Conference, pp. 702–708.

38. ^Ye K, et al. (2024). "A Mutual-Influence-Aware Heuristic Method for Quantum Circuit Mapping." IEEE Trans Comput. **73**(12):2855–2867.

39. ^Hung WNN, Song X, Yang G, Yang J, Perkowski M (2004). "Quantum Logic Synthesis by Symbolic Reachability Analysis." In: Proceedings of the 41st Annual Design Automation Conference, pp. 838–841.

40. ^Li X-W, Zhang X-M, Cheng B, Yung M-H (2024). "Reachability Deficit of Variational Grover Search." Phys. Rev. A. **109**(1):012414.

41. ^Hietala K, Rand R, Hung S-H, Li L, Hicks M (2021). "Proving Quantum Programs Correct." In: 12th International Conference on ITP 2021, pp. 21:1–21:19.

42. ^Lin Y, Guan J, Fang W, Ying M, Su Z (2024). "VeriQR: A Robustness Verification Tool for Quantum Machine Learning Models." arXiv preprint arXiv:2407.13533.

43. ^Guan J, Fang W, Ying M (2021). "Robustness Verification of Quantum Classifiers." In: International Conference on Computer Aided Verification. Springer, pp. 151–174.

44. ^Franco N, Wollschlaeger T, Gao N, Lorenz JM, Guennemann S (2022). "Quantum Robustness Verification: A Hybrid Quantum-Classical Neural Network Certification Algorithm." arXiv preprint arXiv:2205.00900.

45. ^Gheorghiu A, Kashefi E, Wallden P (2015). "Robustness and Device Independence of Verifiable Blind Quantum Computing." New J Phys. **17**(8):083040.

46. ^Fey G, Drechsler R (2008). "A Basis for Formal Robustness Checking." In: 9th International Symposium on Quality Electronic Design (isqed 2008), pp. 784–789.

47. ^Gay SJ, Nagarajan R, Papanikolaou N (2008). "QMC: A Model Checker for Quantum Systems." In: Computer Aided Verification. Berlin, Heidelberg: Springer, pp. 543–547.

48. a, b Feng Y, Yu N, Ying M (2013). "Model Checking Quantum Markov Chains." J Comput Syst Sci. **79**(7):1181–1198.

49. ^Ying M (2021). "Model Checking for Verification of Quantum Circuits." In: International Symposium on Formal Methods. Springer, pp. 23–39.

50. ^Do CM, Ogata K (2024). "Symbolic Model Checking Quantum Circuits in Maude." PeerJ Comput Sci. **10**:e2098.

51. ^Munir M, Gopikanna A, Fayyazi A, Pedram M, Nazarian S (2021). "QMC: A Formal Model Checking Verification Framework For Superconducting Logic." In: Proceedings of the 2021 Great Lakes Symposium on VLSI, pp. 259–264.

52. ^Davidson T, Gay S, Mlnařík H, Nagarajan R, Papanikolaou N (2012). "Model Checking for Communicating Quantum Processes." Int J Unconv Comput. **8**:73–98.

53. ^Papanikolaou NK (2009). "Model Checking Quantum Protocols." Ph.D. dissertation, University of Warwick.

54. ^Jeon S, et al. (2024). "Quantum Probabilistic Model Checking for Time-Bounded Properties." Proc ACM Program Lang. **8**(OOPSLA2):557–587.

55. ^Turrini A (2022). "An Introduction to Quantum Model Checking." Appl Sci. **12**(4):2016.

56. ^Chen Y-F, et al. (2023). "AutoQ: An Automata-Based Quantum Circuit Verifier." In: Computer Aided Verification. Springer, pp. 139–153.

57. ^Anticoli L, Piazza C, Taglialegne L, Zuliani P (2016). "Towards Quantum Programs Verification: From Quipper Circuits to QPMC." In: International Conference on Reversible Computation. Springer, pp. 213–219.

58. ^Gay S, Nagarajan R, Papanikolaou N (2005). "Probabilistic Model–Checking of Quantum Protocols." arXiv preprint quant-ph/0504007.

59. ^Seiter J, Soeken M, Wille R, Drechsler R (2012). "Property Checking of Quantum Circuits Using Quantum Multiple-Valued Decision Diagrams." In: International Workshop on Reversible Computation. Springer, pp. 183–196.

60. ^Wille R, Hillmich S, Burgholzer L (2022). "Decision Diagrams for Quantum Computing." In: Design Automation of Quantum Computers. Springer, pp. 1–23.

61. ^Wille R, Hillmich S, Burgholzer L (2022). "Tools for Quantum Computing Based on Decision Diagrams." ACM Trans Quantum Comput. **3**(3):13.

62. ^Yamashita S, Minato S, Miller DM (2008). "DDMF: An Efficient Decision Diagram Structure for Design Verification of Quantum Circuits Under a Practical Restriction." IEICE Trans Fundam Electron Commun Comput Sci. **91-A**:3793-3802.

63. ^Hong X, et al. (2023). "Decision Diagrams for Symbolic Verification of Quantum Circuits." In: 2023 IEEE International Conference on Quantum Computing and Engineering (QCE), pp. 970–977.

64. ^Wang S-A, Lu C-Y, Tsai I-M, Kuo S-Y (2008). "An XQDD-Based Verification Method for Quantum Circuits." IEICE Trans Fundam. **E91-A**(2):584–594.

65. ^Wang Z, Cheng B, Yuan L, Ji Z (2025). "FeynmanDD: Quantum Circuit Analysis with Classical Decision Diagrams." In: International Conference on Computer Aided Verification. Springer, pp. 28–52.

66. ^Abramsky S, Coecke B (2004). "A Categorical Semantics of Quantum Protocols." In: Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science, pp. 415-425.

67. ^Rand R, Paykin J, Zdancewic S (2018). "QWIRE Practice: Formal Verification of Quantum Circuits in Coq." Electron Proc Theor Comput Sci. **266**:119–132.

68. ^Amy M (2019). "Towards Large-Scale Functional Verification of Universal Quantum Circuits." Electron Pro c Theor Comput Sci. 287:1–21.

69. ^Kaivola R, Aagaard MD (2000). "Divider Circuit Verification with Model Checking and Theorem Proving." In: International Conference on Theorem Proving in Higher Order Logics. Springer, pp. 338–355.

70. ^Chen Y-F, Rümmer P, Tsai W-L (2023). "A Theory of Cartesian Arrays (With Applications in Quantum Circu it Verification)." In: International Conference on Automated Deduction. Springer, pp. 170–189.

71. ^Mahadev U (2018). "Classical Verification of Quantum Computations." In: 2018 IEEE 59th Annual Symposi um on Foundations of Computer Science (FOCS), pp. 259-267.

72. ^Govindankutty A, Srinivasan SK, Mathure N (2023). "Rotational Abstractions for Verification of Quantum Fourier Transform Circuits." IET Quantum Commun. 4(2):84–92.

73. ^Govindankutty A, Srinivasan SK (2025). "Superposition-Based Abstractions For Quantum Data Encoding Verification." IET Quantum Commun. doi:10.1049/qtc2.70002.

74. a, b, cTao R, et al. (2022). "Giallar: Push-Button Verification for the Qiskit Quantum Compiler." In: Proceeding s of the 43rd ACM SIGPLAN Conference on PLDI, pp. 641–656.

75. a, bHietala K, Rand R, Hung S-H, Wu X, Hicks M (2021). "A Verified Optimizer for Quantum Circuits." Proc AC M Program Lang. 5(POPL):1–29.

76. ^Tang W, Tomesh T, Suchara M, Larson J, Martonosi M (2021). "Cutqc: Using Small Quantum Computers for Large Quantum Circuit Evaluations." In: Proceedings of the 26th ACM International Conference on ASPLOS, pp. 473–486.

77. a, bBurgholzer L, Raymond R, Wille R (2020). "Verifying Results of the IBM Qiskit Quantum Circuit Compilat ion Flow." In: 2020 IEEE International Conference on Quantum Computing and Engineering (QCE), pp. 356 –365.

78. ^Villalonga B, et al. (2019). "A Flexible High-Performance Simulator for Verifying and Benchmarking Quant um Circuits Implemented on Real Hardware." npj Quantum Inf. 5(1):86.

79. a, bWille R, Hillmich S, Burgholzer L (2020). "Efficient and Correct Compilation of Quantum Circuits." In: 20 20 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1–5.

80. ^Hurwitz L, Datta K, Kole A, Drechsler R (2024). "Is Simulation the Only Alternative for Effective Verificatio n of Dynamic Quantum Circuits?" In: International Conference on Reversible Computation. Springer, pp. 20 1–217.

81. ^Xu A, Molavi A, Pick L, Tannu S (2023). "Synthesizing Quantum-Circuit Optimizers." Proc ACM Program L ang. 7(PLDI):140.

82. ^Devitt SJ, Munro WJ, Nemoto K (2013). "Quantum Error Correction for Beginners." Rep Prog Phys. **76**(7):076 001.

83. ^Kitaev AY (1997). "Quantum Computations: Algorithms and Error Correction." Russ Math Surv. **52**(6):1191.

84. ^Knill E, Laflamme R (1997). "Theory of Quantum Error-Correcting Codes." Phys. Rev. A. **55**(2):900–911.

85. ^Huang Q, Zhou L, Fang W, Zhao M, Ying M (2025). "Efficient Formal Verification of Quantum Error Correct ing Programs." Proc ACM Program Lang. **9**(PLDI):190.

86. ^Poulin D (2005). "Stabilizer Formalism for Operator Quantum Error Correction." Phys. Rev. Lett.. **95**(23):23 0504.

87. ^Fang W, Ying M (2024). "Symbolic Execution for Quantum Error Correction Programs." Proc ACM Program Lang. **8**(PLDI):189.

88. ^Wu A, Li G, Zhang H, Guerreschi GG, Xie Y, Ding Y (2021). "QECV: Quantum Error Correction Verification." ar Xiv preprint arXiv:2111.13728.

89. ^Chen K, et al. (2025). "Verifying Fault-Tolerance of Quantum Error Correction Codes." arXiv preprint arXiv: 2501.14380.

90. ^Chong FT, Franklin D, Martonosi M (2017). "Programming Languages and Compiler Design for Realistic Q uantum Hardware." Nature. **549**(7671):180–187.

91. ^Salm M, Barzen J, Leymann F, Weder B, Wild K (2021). "Automating the Comparison of Quantum Compiler s for Quantum Circuits." In: Service-Oriented Computing. Springer, pp. 64–80.

92. ^Nguyen T, et al. (2021). "Quantum Circuit Transformations with a Multi-Level Intermediate Representatio n Compiler." arXiv preprint arXiv:2112.10677.

93. ^Yan G, et al. (2025). "Quantum Circuit Synthesis and Compilation Optimization: Overview and Prospects." arXiv preprint arXiv:2407.00736.

94. ^Liu J, et al. (2019). "Formal Verification of Quantum Algorithms Using Quantum Hoare Logic." In: Compute r Aided Verification. Springer, pp. 187–207.

95. ^Amy M, Lunderville J (2025). "Linear and Non-Linear Relational Analyses for Quantum Program Optimiz ation." Proc ACM Program Lang. **9**(POPL):37.

96. ^Zuliani P (2007). "A Formal Derivation of Grover's Quantum Search Algorithm." In: First Joint IEEE/IFIP Sy mposium on TASE '07, pp. 67–74.

97. ^Witzel WM, Craft WD, Carr R, Kapur D (2023). "Verifying Quantum Phase Estimation Using an Expressive Theorem-Proving Assistant." Phys. Rev. A. **108**(5):052609.

98. ^Peng Y, et al. (2023). "A Formally Certified End-to-End Implementation of Shor's Factorization Algorithm." *Proc Natl Acad Sci.* **120**(21):e2218775120.

99. ^Alam MS, et al. (2022). "Practical Verification of Quantum Properties in Quantum-Approximate-Optimization Runs." *Phys. Rev. Appl.*. **17**(2):024026.

Declarations

Funding: No specific funding was received for this work.

Potential competing interests: No potential competing interests to declare.