A Gentle Introduction to Level-Synchronized Tree Automata (Tentative)

No Author Given

No Institute Given

Abstract.

1 Introduction

We may get substantial added value when connecting two complementary areas of computer science. This applies particularly when adapting mature techniques developed in one area to solve complex problems that arise in another (tyupically) new area. The current paper illustrates one such a case. It describes the application of techniques developed in logic, automata, and symbolic verification to analyze the correctness of quantum circuits.

The current quest of quantum computing is achieving so called *quantum supremacy*, meaning that we reach a stage in the technology development where we can solve problems that are practically unsolvable using conventional computing. To that end, substantial effoprt is ongoing to implement quantum hardware and develop programming languages for quantum computers. In many applications, system correctness is of critical importance. For instance, identifying a subtle bug in a quantum circuit used for quantum chemistry simulations could prevent incorrect predictions about molecular interactions, which are critical for drug discovery and material design.

Given the complexity of quantum systems and the above mentioned correctness requirements, tools for analyzing and verifying quantum programs' correctness are of critical importance. However, building verification tools for quantum programs is a formiddable challenge. First, due their behaviors are probabilistic in nature, and their computational space is exponential in size. Furthermore, there is a fundamental difference in the manner in which conventional and quantum computers store information: conventional computers use digital bits (0s and 1s), while quantum computers use quantum bits (qubits) that probabilistically have values in nthe intgerval between 0 and 1.

The current paper reports on research in whose goal is adating the research community's vast experience in conventional program verification to develop tools for verifying quantum systems. Verification tools, in general, and for quantum systems in particular, would ideally have the following properties: (1) Flexibility: allows flexible specification of properties of interest, (2) Diagnostics: provides precise bug diagnostics, (3) Automation: Operates automatically, and (4) Scalability: scales efficiently to verify useful programs. Symbolic verification is

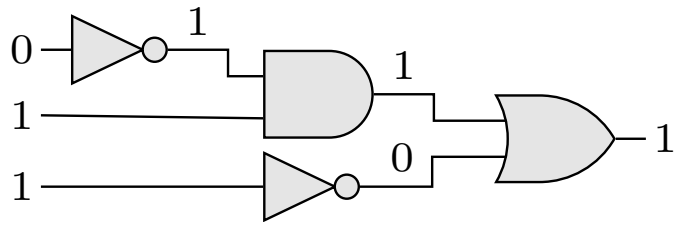
one of the most successful techniques that satisfy the above criteria for conventional programs. Notable instances, are invariant certification (e.g., in the form of Hoare triples), and model checking. Despite the overwheling success achieved in the verification of conventional hardware and software, we are lacking an extension to the quantum realm. The principal reasonm isthe latter's unique mathematical structure and different operational principles.

In this work we describe an innovative application of automata theory to quantum circuit verification. More precisley, we combine we use tree automata-based symbolic representations for represeting quantum states. The class of tree we consider is tailored for quantum circuits, considerably extending the sacalbility of quantum circuit verification comnpared to existing techniques. As expected, when dealing with an entirely new application area, we we go back to the very basics of program verification. We consider classical Hoare triples $\{P\}C\{Q\}$, where P, the *pre-condition*, represents a set of initial quantum states, Q, the *post-conditions* represents a set of quantum *target* states, and C is a quantum circuit. It uses symbolic verification to check the validity of the triple, ensuring that all executions of C from states in P result in states within Q. More precisely, we will resprent P and Q by a special form of tree automata, and provide algorithms to check whether we reach Q from P if we execute C. The framework is based on the observation that we can lift core concepts of classical verification, such as state-space exploration and symbolic reasoning, into the quantum setting.

Although this is the first attempt to use such techniques in the context of quantum computing, implementing the framework with a tool gives spectacular results. For instance, the tool manages to verify large circuits with up to 40 qubits and 141,527 gates or catch bugs injected into circuits with up to 320 qubits and 1,758 gates, while all exiting tools fail to handle such benchmarks.

This extension allows the verification of quantum circuits, offering a path to applying well-established classical paradigms in a domain where formal guarantees are critical.

2 Circuits



I convention a circuits, we store information as sets of *bits* with values 0 and 1. The basic data transformer is a Boolean *gate*. We can see a gate as a Boolean

function taking a sequence of Binary bits as input, and producing an output bit as output. A combinatorial circuit consists of set of gates and can also been as Boolean function. The circuits takes a sequence of bits as input and a produces a sequence of bits as output. Typically, we also have a set of internal bits that represent the connection between gate In **??**, the circuit has three input bits, one output bit, and three internal bits.

Often, we consider the *state* of a sequence of bits by giving the sequence of their values. For instance, the input state in **??** is 011 since the three input bits have values 0, 1, and 1 in that order.

3 Conclusions

In addition to its early practical promise, the approach opens new research directions in automata theory, where quantum structures introduce novel mathematical challenges and opportunities. It establishes a connection between quantum program verification and automata, promoting new possibilities to exploit the richness of automata theory and automata-based verification in quantum computing. It is worth noting that the methodology is not just about catching mistakes but also about building trust in quantum computing systems as we move toward an era in which they might solve problems classical systems cannot.

In conclusion, the paper represents a confluence of disciplines, opening pathways for collaboration between automata theorists, quantum physicists, and software engineers. Looking ahead, this line of research opens up exciting opportunities. Could similar automata-based techniques be adapted to other aspects of quantum software engineering? Could this approach handle quantum programming languages' more abstract and flexible constructs?