1

Introduction

1.1 Second Quantum Revolution Requires New Verification Techniques

We are currently in the midst of a second quantum revolution: *transition from quantum theory to quantum engineering* [41]. The aim of quantum theory is to find fundamental rules that govern the physical systems already existing in the nature. Instead, quantum engineering intends to design and implement new systems (machines, devices, etc.) that did not exist before to accomplish some desirable tasks, based on quantum theory. Active areas of quantum engineering include quantum computing, quantum cryptography, quantum communication, quantum sensing, quantum simulation, quantum metrology and quantum imaging.

Experiences in today's engineering indicate that it is not guaranteed that a human designer completely understands the behaviours of the systems she or he designed, and a bug in her or his design may cause some serious problems and even disasters. So, correctness, safety and reliability of complex engineering systems have attracted wide attention and have been systematically studied in various engineering fields. In particular, in the past four decades, computer scientists have developed various verification techniques for correctness of both hardware and software as well as security of communication protocols.

As is well known, human intuition is much better adapted to the classical world than to the quantum world. This implies that human engineers will commit many more faults in designing and implementing complex quantum systems such as quantum computer hardware and software and quantum communication protocols. Thus, correctness, safety and reliability problems will be even more critical in quantum engineering than in today's engineering. However, because of the essential differences between the classical and quantum worlds, verification techniques developed for classical engineering systems cannot be directly used to quantum systems. Novel verification techniques will be indispensable for the coming era of quantum engineering and quantum technology [32].

1.2 Model-Checking Techniques for Classical Systems

Model checking is an effective automated technique that checks whether a desired property is satisfied by a system, for example, a computing or communication system. The properties that are checked are usually specified in a logic, in particular, temporal logic; typical properties are deadlock freedom, invariants, safety and request–response properties. The systems under checking are mathematically modelled as, for example (finite-state) automata, transition systems, Markov chains and Markov decision processes [7, 35].

Model checking has become one of the dominant techniques for verification of computer (hardware and software) systems 30 years after its inception. Many industrial-strength systems have been verified by employing model-checking techniques. Recently, it has also successfully been used in systems biology; see [68] for example.

With quantum engineering and quantum technology emerging, a question then naturally arises: *Is it possible to use model-checking techniques to verify correctness and safety of quantum engineering systems* and if so, how?

1.3 Difficulty in Model Checking Quantum Systems

Unfortunately, it seems that the current model-checking techniques cannot be directly applied to quantum systems because of some essential differences between the classical world and the quantum world. To develop model-checking techniques for quantum systems, the following three problems must be systematically addressed:

- ***System modelling and property specification*:** The classical system modelling method cannot be used to describe the behaviours of quantum systems, and the classical specification language is not suited to formalise the properties of quantum systems to be checked. So, we need to carefully and clearly define a conceptual framework in which we can properly reason about quantum systems, including *formal models of quantum systems* and *formal description of temporal properties of quantum systems*.
- ***Quantum measurements*:** Model checking is usually applied to check long-term behaviours of the systems. But to check whether a quantum system satisfies a certain property at a time point, one has to perform a quantum measurement on the system, which can change the state of the system. This makes studies of the long-term behaviours of quantum systems much harder than those of classical systems [22, 23, 60].
- ***Algorithms*:** The state spaces of the classical systems that model-checking algorithms can be applied to are usually finite or countably infinite. However, the

state spaces of quantum systems are inherently continuous even when they are finite dimensional. To develop algorithms for model checking quantum systems, we have to exploit some deep mathematical properties of the systems so that it suffices to examine only a finite number of (or at most countably infinitely many) representative elements, for example, those in an orthonormal basis, of their state spaces. Also, a linear algebraic structure always resides in the state space of a quantum system. So, an algorithm checking a quantum system should be carefully developed so that the linear algebraic structure is nicely preserved and fully exploited.

1.4 Current Research on Model Checking of Quantum Systems

Despite the difficulties discussed in the previous section, quite a few model-checking techniques for quantum systems have been developed in the past 10 years. The earliest work mainly targeted checking quantum communication protocols:

- Taking the probabilism arising from quantum measurements into account, [54] used the probabilistic model-checker PRISM [75] to verify the correctness of quantum protocols, including superdense coding, quantum teleportation and quantum error correction.
- A branching-time temporal extension (called quantum computation tree logic or QCTL for short) of exogenous quantum propositional logic [88] was introduced and then the model-checking problem for this logic was studied in [8, 9], with verification of the correctness of quantum key distribution BB84 [15] as an application.
- A linear temporal extension QLTL of exogenous quantum propositional logic [88] was then defined and the corresponding model-checking problem was investigated in [87].
- Model-checking techniques were developed in [38, 39] for quantum communication protocols modelled in process algebra CQP (Communicating Quantum Processes) [56]. The checked properties are specified by the quantum computation tree logic QCTL defined in [8].
- A model checker for quantum communication protocols was also developed in [55, 57, 96], where the checked properties are specified by QCTL [8] too, but only the protocols that can be modelled as quantum circuits expressible in the stabiliser formalism [59] were considered. In [5, 6], this technique was extended beyond stabiliser states and used to check equivalence of quantum protocols.

A research line pursued by the authors and their collaborators is to develop model-checking techniques that can be used not only for quantum communication

protocols but also for quantum computing hardware and software and other quantum engineering systems:

- In retrospect, our research on model checking quantum systems stemmed from termination analysis of quantum programs. The termination problem of quantum loop programs with unitary transformation as loop bodies (in a finite-dimensional state Hilbert space) was first examined in [118]. The semantics of this class of quantum programs can be modelled by *quantum automata*. The main results of [118] were generalised in [123] to quantum loops with general quantum operations (or super-operators) as loop bodies by introducing *quantum Markov chains* as their semantic models. These researches naturally motivated us to the studies of model checking quantum systems, because termination can be seen as a kind of reachability, which is central to model-checking algorithms.
- The model-checking problem for *quantum automata* was first considered in [119], where closed subspaces of the state Hilbert space are used as the atomic propositions about the behaviour of the system, following the basic idea of Birkhoff-von Neumann quantum logic, and the checked linear-time properties are defined as infinite sequences of sets of atomic propositions. Furthermore, decidability or undecidability of several reachability problems (eventually reachable, globally reachable, ultimately forever reachable and infinitely often reachable) for quantum automata were proved in [82].
- The reachability problem of *quantum Markov chains* was first investigated in [123], where an algorithm for computing the reachable space of a quantum Markov chain was presented and applied to termination analysis of concurrent quantum programs. A more systematic study in this direction was carried out in [120] by developing a new graph theory in Hilbert spaces; in particular, an algorithm for computing several kinds of reachability probabilities of quantum Markov chains was found based on the BSCC (bottom strongly connected components) decomposition of their state Hilbert spaces, and undecidability of some other reachability problems were proved. The same problems for *quantum Markov decision processes* were studied in [121].
- The notion of a *super-operator-valued Markov chain* was introduced in [51] as a higher-level model of quantum programs and quantum cryptographic protocols, where the (classical) control flow of a quantum program is depicted as a (classical) directed graph, but each edge is associated with a super-operator that describes one step of quantum computation. A corresponding computation tree logic (CTL) was also defined, and algorithms for checking CTL properties of super-operator-valued Markov chains are developed. Furthermore, the reachability of the recursive extension of super-operator-valued Markov chains was studied in [52].

1.5 Structure of the Book

This book is a systematic exposition of the currently existing principles and algorithms for model checking quantum systems. The remainder of this book is divided into the following chapters:

- **Chapters 2 and 3** are the preliminary part of this book. For convenience of the reader, we briefly review model checking in Chapter 2, from mathematical *models of systems* to *temporal logics* for specifying properties of systems and basic *model-checking algorithms*. In Chapter 3, we review the basics of quantum theory needed in the subsequent chapters, including *static* and *dynamic* descriptions of a quantum system and quantum *measurements*.
- **Chapter 4:** From this chapter on, we develop the techniques for model checking quantum systems step by step, from a simple model of quantum systems to more and more complicated ones.

  This chapter starts from defining *linear time properties* of quantum systems and then focuses on the study of a special linear time property, namely *reachability* of *quantum automata*, in which the system's transition is modelled as a unitary transformation that is a discrete-time description of the dynamics of a closed quantum system.
- **Chapter 5:** In this chapter, we consider reachability problems of *quantum Markov chains* and *quantum Markov decision processes*, which, as suggested by their names, are the quantum counterparts of Markov chains and Markov decision processes. Their dynamics is described as a super-operator rather than a unitary transformation. We first introduce some necessary mathematical tools, in particular graph theory in Hilbert spaces, and then present several algorithms solving these reachability problems.
- **Chapter 6:** In this chapter, we first define the notion of a super-operator-valued Markov chain (SVMC) and both a computation tree logic (CTL) and a linear temporal logic (LTL) for specifying properties of SVMCs. The majority of this chapter is devoted to introducing a series of algorithms for checking CTL or LTL properties of SVMCs.
- **Chapter 7:** This is the concluding chapter, where we discuss some possible improvements and potential applications of the model-checking techniques for quantum systems introduced in this book and point out several directions for the further developments of this area.
- **Appendices:** For readability, the proofs of some technical lemmas are omitted in Chapters 4–6. But we provide these proofs in the appendices for the readers who are interested in them.

2

Basics of Model Checking

Model checking is an algorithmic technique for verifying certain properties of (mainly) finite state systems. The systems are usually modelled as a transition system (or a finite state automaton, a labelled graph). The properties are specified in a temporal logic. The checking algorithm is based mainly on systematic inspection of all reachable states of the model. Because of its complete automation and ability of finding counterexamples, model checking has been successfully and widely adopted in the information and communications technology industries. On the other hand, it has a major drawback, namely the *state space explosion* problem – the number of states can grow exponentially in the number of variables. Several techniques have been introduced to mitigate this drawback, including symbolic model checking, bounded model checking, abstraction and partial order reduction. In real-world applications, model checking is facing the *validation* problem that all branches of science have: are the model and the properties being checked a proper and adequate description of the system's behaviour?

Model checking was first proposed for verification of classical non-probabilistic systems and then extended for probabilistic systems. In this book, we will further extend the technique of model checking for quantum systems. As preliminaries, this chapter introduces basics of model checking for both classical non-probabilistic and probabilistic systems.

The ideas and techniques introduced in this chapter cannot be directly applied to quantum systems, but they provide us with a guideline to develop an appropriate framework and to ask the right questions in the later chapters.

2.1 Modelling Systems

First of all, we need a formal model describing the possible behaviour of the system under consideration. One of the most commonly used models is a transition system.

Definition 2.1 A transition system is a 6-tuple

$$\mathcal{M} = (S, Act, \rightarrow, I, AP, L),$$

where

(i) S is a (finite) set of states;
(ii) $I \subseteq S$ is a set of initial states;
(iii) Act is a set of (the names of) actions;
(iv) $\rightarrow \subseteq S \times Act \times S$ is a transition relation;
(v) AP is a set of atomic propositions;
(vi) $L : S \rightarrow 2^{AP}$ is a labelling function, where 2^{AP} stands for the power set of AP, that is, the set of all subsets of AP.

Several ingredients in the foregoing definition deserve careful explanation:

- $(s, \alpha, s') \in \rightarrow$, usually written as $s \xrightarrow{\alpha} s'$, means that the action α causes the system's state to change from s to s'.
- The transition relation $\rightarrow$ can be equivalently represented by a family of transition relations indexed by action names:

$$\rightarrow = \left\{ \xrightarrow{\alpha} : \alpha \in Act \right\},$$

where for each $\alpha \in Act$,

$$\xrightarrow{\alpha} = \left\{ (s, s') : s \xrightarrow{\alpha} s' \right\} \subseteq S \times S$$

is the set of transitions enabled by action α.
- Elements of AP are atomic propositions chosen to describe the basic properties of the system's states.
- For each $s \in S$, $L(s)$ denotes the set of those atomic propositions that hold in state s.

For each $s \in S$ and $\alpha \in Act$, let

$$post(s, \alpha) = \{ s' \in S : s \xrightarrow{\alpha} s' \}$$

be the set of α-successors of s. We write $|X|$ for the number of elements in X.

Definition 2.2 A transition system $\mathcal{M}$ is called deterministic if

(i) there is at most one initial state; that is, $|I| \leq 1$;
(ii) for each action α, each state s has at most one α-successor; that is, it holds that $|post(s, \alpha)| \leq 1$ for every $s \in S$ and $\alpha \in Act$.

Otherwise, it is non-deterministic.

Definition 2.3 A state $s \in S$ is called a terminal state of the transition system $\mathcal{M}$ if it has no outgoing transition; that is, $post(s, \alpha) = \emptyset$ for every $\alpha \in Act$.

A transition system $\mathcal{M}$ runs in the following way: it starts from some initial state $s_0 \in I$ and then evolves according to the transition relation $\rightarrow$. Formally, we have:

Definition 2.4 A path in the transition system $\mathcal{M}$ is a (finite or infinite) sequence $\pi = s_0 s_1 \dots s_{i-1} s_i \dots$ of states such that

$$s_0 \overset{\alpha_1}{\rightarrow} s_1 \overset{\alpha_2}{\rightarrow} \cdots s_{i-1} \overset{\alpha_i}{\rightarrow} s_i \overset{\alpha_{i+1}}{\rightarrow} \cdots,$$

where $s_{i-1} \overset{\alpha_i}{\rightarrow} s_i$ is a transition in $\mathcal{M}$ for each $i \geq 1$.

Note that for a non-deterministic transition system, the initial state s_0 and the next state s_i at the ith step in Definition 2.4 may be chosen non-deterministically.

For a path $\pi = s_0 s_1 \dots$ and $i \geq 0$, we write

$$\pi[i] = s_i, \qquad \pi(i) = s_i s_{i+1} \dots$$

for the $(i+1)$th state s_i and the suffix of π starting in state s_i, respectively.

Definition 2.5 A state $s \in S$ is called reachable in $\mathcal{M}$ if there is a path $\pi = s_0 s_1 \dots s_{n-1} s_n$ in $\mathcal{M}$ starting at an initial state $s_0 \in I$ and ending at $s_n = s$.

As stated at the beginning of this chapter, model checking is done by inspecting all reachable states of the system. This central notion of reachable state will therefore be generalised into various quantum systems, and computing (the space of) reachable states of a quantum system will be one of the central issues discussed in this book.

2.2 Temporal Logics

We also need a formal language to specify the required properties of the system. Since we are interested in its dynamic properties, a temporal logic(al language) is often adopted, which is an extension of propositional logic with some operators that can describe the behaviour over time. Mainly, two types of temporal logics are used in model checking. They are chosen according to two different views on the notion of (discrete) time.

2.2.1 Linear Temporal Logic

Linear temporal logic (LTL) is employed to describe linear-time properties. The linearity means:

- *Each time point has a unique possible future.*

We assume that the reader is familiar with propositional logic. The LTL language is an expansion of propositional logical language. Its alphabet consists of

- A set AP of atomic propositions, ranged over by meta-variables $a, a_1, a_2, \ldots$
- Propositional connectives: $\neg$ (not), $\wedge$ (and)
- Temporal operators: O (next), U (until)

It is worth noting that a set AP of atomic propositions is also assumed in a transition system (see Definition 2.1). Indeed, AP is the point where a temporal logical formula is connected to a transition system. More precisely, the labelling function $L : S \to 2^{AP}$ in the transition system gives an interpretation of atomic propositions:

$$\text{Atomic proposition } a \in AP \text{ is true in state } s \;\Leftrightarrow\; a \in L(s). \tag{2.1}$$

The LTL formulas are generated from atomic propositions by a finite number of applications of connectives $\neg, \wedge$ and temporal operators O, U.

Definition 2.6 (Syntax) The LTL formulas over AP are defined by the grammar

$$\varphi ::= a \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid O\varphi \mid \varphi_1 U \varphi_2.$$

The meanings of $\neg\varphi$ and $\varphi_1 \wedge \varphi_2$ are the same as in propositional logic. Intuitively, $O\varphi$ is true at the current point of time if φ is true at the next, $\varphi_1 U \varphi_2$ holds at the current time point if there is a future point of time at which φ_2 is true and φ_1 holds at all moments from the current to that future point.

The following abbreviations are often used to simplify the presentation of LT formulas:

$$\begin{aligned} \mathbf{true} &:= a \vee \neg a; \\ \varphi_1 \vee \varphi_2 &:= \neg(\neg\varphi_1 \wedge \neg\varphi_2); \\ \Diamond\varphi &:= \mathbf{true}\, U \varphi; \\ \Box\varphi &:= \neg\Diamond\neg\varphi. \end{aligned}$$

Again, the meanings of **true** and $\varphi_1 \vee \varphi_2$ are the same as in propositional logic. Moreover, we can see that $\Diamond\varphi$ means that φ will be true eventually (sometime in the future), and $\Box\varphi$ means that φ will be true always (from now on forever). It is worth noting that the derived formulas introduced earlier do not increase the expressive power of LTL, but LTL formulas can often be shortened using these abbreviations.

Example 2.7

(i) $\Box\Diamond\varphi$: for every time point i, there exists some $j \geq i$ such that φ is true at time point j; that is, φ holds infinitely often.
(ii) $\Diamond\Box\varphi$: there is a time point i such that φ is true at all time points $j \geq i$; that is, φ holds eventually forever.
(iii) $\Box$(request $\to \Diamond$response): every request will eventually have a response.

The semantics of the logic is obtained by extending interpretation (2.1) of atomic propositions to all LTL formulas.

Definition 2.8 (Semantics) Let $\mathcal{M} = (S, Act, \rightarrow, I, AP, L)$ be a transition system, π a path in $\mathcal{M}$, $s \in S$ and φ an LTL formula over AP. Then

(i) The satisfaction $\pi \models \varphi$ is defined by induction on the structure of φ:

(a) $\varphi = a$: $\pi \models \varphi$ iff $a \in L(\pi[0])$;
(b) $\varphi = \neg\varphi'$: $\pi \models \varphi$ iff $\pi \not\models \varphi'$;
(c) $\varphi = \varphi_1 \wedge \varphi_2$: $\pi \models \varphi$ iff $\pi \models \varphi_1$ and $\pi \models \varphi_2$;
(d) $\varphi = O\varphi'$: $\pi \models \varphi$ iff $\pi[1] \models \varphi'$;
(e) $\varphi = \varphi_1 U \varphi_2$: $\pi \models \varphi$ iff there exists $i \geq 0$ such that $\pi[i] \models \varphi_2$ and $\pi[j] \models \varphi_1$ for all $0 \leq j < i$.

(ii) $s \models \varphi$ iff $\pi \models \varphi$ for all paths π starting in s.
(iii) $\mathcal{M} \models \varphi$ iff $s_0 \models \varphi$ for all initial states $s_0 \in I$.

Essentially, Definition 2.8 is a formal description of the intuitive explanations of LTL formulas given after Definition 2.6.

Note that in (2.1), we directly considered whether a state s satisfies an atomic proposition a. In Definition 2.8, however, the satisfaction of a general LTL formula by a state needs to be formulated in two steps. In clause (i), we first consider whether an LTL formula φ is satisfied by a path which represents the notion of linear time, since φ may contain some temporal operators. Then in clause (ii), the satisfaction of an LTL formula by a state can be defined in terms of satisfaction by all paths starting from the state.

Let us further carefully explain the sub-clauses of clause (i) as follows:

- Sub-clause (a) is indeed a restatement of (2.1), with s being the initial state $\pi[0]$ of path π.
- The interpretations of connectives $\neg, \wedge$ in sub-clauses (b) and (c) are the same as in the standard propositional logic.
- Sub-clause (d) means that $O\varphi'$ is satisfied by path π iff φ' is satisfied by the tail $\pi[1]$ of π starting at the next point of time.
- Sub-clause (e) states that $\varphi_1 U \varphi_2$ is satisfied by π iff φ_2 is satisfied at some point i of time in path π, and before that point, φ_1 is satisfied.

Example 2.9 Consider transition system $\mathcal{M} = (S, Act, \rightarrow, I, AP, L)$ depicted in Figure 2.1, where

- $S = \{s_1, s_2, s_3\}$;
- $Act = \{F, B, C\}$;
- $s_1 \xrightarrow{F} s_2 \xrightarrow{F} s_3$, $s_2 \xrightarrow{B} s_1$ and $s_3 \xrightarrow{C} s_3$;