applied sciences

Review

An Introduction to Quantum Model Checking

Andrea Turrini [1,2,†]

[1] State Key Laboratory of Computer Science, Institute of Software Chinese Academy of Sciences, Beijing 100190, China; turrini@ios.ac.cn
[2] Institute of Intelligent Software Guangzhou, Guangzhou 511458, China
[†] Current address: Zhongguancun 4# South Fourth Street, Beijing 100190, China.

Abstract: Model checking is a well-established and widely adopted framework used to verify whether a given system satisfies the desired properties. Properties are usually given by means of formulas from a specific logic; there are several logics that can be used, such as CTL and LTL, which permit the expression of different types of properties on the branching-time or on the linear-time evolution of the system. In this paper, we will consider the problem of model checking quantum systems and present the solutions given in literature for solving such a problem with respect to different types of properties.

Keywords: model checking; quantum Markov chain; linear-time properties; branching-time properties; fidelity

check for updates

Citation: Turrini, A. An Introduction to Quantum Model Checking. *Appl. Sci.* **2022**, *12*, 2016. https://doi.org/10.3390/app12042016

Academic Editor: Augusto Ferrante

Received: 20 December 2021
Accepted: 11 February 2022
Published: 15 February 2022

1. Introduction

One of the main questions a designer needs to answer while developing a new system, being it a program, a protocol, or some hardware, is: *how can I be sure my system works as expected?* There are several ways to answer this question.

- One can test the system, by feeding it with several inputs and looking at the correctness of the generated output. However, as Dijkstra said in his Turing Award lecture [1],

  > "program testing can be a very effective way to show the presence of bugs, but is hopelessly inadequate for showing their absence."

  In fact, if the testing inputs do not trigger the bug, it remains undetected, ready to cause damages once the system is deployed to production.
- One can manually prove that the system is correct by applying the techniques learned in programming and cryptography courses, such as by using Hoare logic/triples to prove properties of programs or by showing that the cryptographic protocol is provably secure. This approach provides the desired guarantees, but it is tedious, error prone, and reasonably applicable only to very small systems.
- One can apply one of the several techniques developed by researches specifically for this purpose, such as model checking, abstract interpretation, and high-order theorem proving. These techniques can be usually applied automatically to the system and are able to scale to large systems.

  This latter approach provides the positive answer to our initial question.

1.1. The Successful Story of Model Checking

Since the seminal works of Clarke and Emerson [2] and Queille and Sifakis [3], model checking has gained a wide popularity in the formal verification community and in industry. This is mainly motivated by the fact that once the model checker gives green light to a system, we can be sure that the system that has been fed into the model checker fulfills all checked properties. On the other hand, if the model checker gives red light to some property, it is usually able to provide a counterexample witnessing why the specific property is violated; such a counterexample can be used by the developer to more easily trace the source of the bug and fix it.

Model checking has been used in a variety of scenarios, from hardware [4–8] to software [9–13], from communication [14–17] and security protocols [18] to pacemakers [19], from airplane and train control [20,21] to satellites and spacecrafts [22–25], just to cite a few examples; more applications of model checking can be found in literature and in several books summarizing the techniques and the results achieved by the researchers, see, e.g., [26–33]. We can identify a few core aspects from these applications, which should be considered when choosing the model checking technique that best fits our needs.

- The system can have a probabilistic behavior, as a result of internal random choices (such as the sampling of nonces in a security protocol) or of external uncertainty (the message is sent on a noisy channel that may alter it).
- The system can have a nondeterministic behavior, as the result of multiple components running in parallel and interacting with each other in no predefined order.
- The system needs to take care of the amount of time available for reacting to input or for completing its tasks: a collision avoidance system installed on an airplane needs to warn the pilot as soon as possible about the risk of the collision; it is certain that it needs to do so within the time limits imposed by the aviation authorities.

Depending on the presence or the absence of these and other aspects, different models and model checking algorithms have been developed specifically for them.

Given the successful application of model checking in several fields, researchers working on quantum systems considered the application of model checking techniques to verify the quantum protocols they were proposing. One of the obstacles in the manual verification of a quantum system is the fact that quantum mechanics is not really intuitive, since there are quantum phenomena for which researchers have no first-hand experience. Entanglement and superposition of the states, for instance, are not something that a person can encounter in daily life; probabilistic choices, instead, are at hand by just flipping a coin or rolling a dice. As a result, it is more likely for quantum designers to unintentionally introduce bugs in a protocol than for protocol designers working with classical systems, where they need to take care of nondeterministic and probabilistic aspects that can be experienced in our macro-scale world. Having erroneous protocols can have a large impact on the usability and reliability of quantum systems, in particular when their usage will become more practical by the advancements in the realization of quantum computers.

Researchers working on quantum systems have already proposed quantum protocols, such as quantum coin-flipping [34,35], quantum key distribution [34–36], and super-dense coding [37], that are at the core of quantum cryptography. Quantum cryptography can be considered a reasonable replacement for classical cryptography, in case the bases of the latter are mined by the achievements in quantum computation. Nowadays, classical cryptography is based on the assumption that operations, such as number factorization and discrete logarithm computation, are difficult to perform, so an attacker needs to spend a considerable amount of time and resources in order to obtain, say, the prime factors p and q out of the given large number $n = pq$. Since factorization and discrete logarithm computation are indeed difficult problems to be solved by ordinary computers, designers rely on them when proposing cryptographic protocols providing security and privacy to our daily life online activities. Quantum computing, however, makes them easy: for instance, Shor [38] already provided quantum algorithms for integer factorization and discrete logarithm computation that work in polynomial time. This justifies the need of quantum protocols that are strong enough to provide the desired levels of security and privacy; moreover, such protocols need to work as expected, and to be trusted to do so.

Model checking is an appropriate technique for verifying that quantum protocols behave correctly and to provide evidence that this is the case, since such a verification task is exactly the motivation why the model checking framework has been developed. In order to apply model checking on quantum protocols, we first need to answer two questions.

1. How can we model formally the given quantum protocol?
2. How can we specify the desired properties?

The reminder of the paper is devoted to the answer to these questions.

1.2. How Can We Model Formally the Given Quantum Protocol?

When modelling quantum systems for model checking them, one of the main characteristics we need to take into consideration is that a quantum system is, by its own nature, continuous. The state space of a quantum system is usually represented as a finite d-dimensional Hilbert space $\mathcal{H}_d$ (the vector space $\mathbb{C}^d$ equipped with an inner product operation), which is a continuum. Although the Hilbert space $\mathcal{H}_d$ can be finitely represented, for instance by taking d pairwise orthogonal vectors forming a basis for $\mathcal{H}_d$, the states in the state space are infinitely many. However, model checking usually expects to deal with finite state spaces, even when applied to continuous systems. For example, for continuous time Markov chains, the model still has a finite state space and the continuous time is represented by exponential distributions governing the delay in taking the transition from a state to another; for timed automata, the automaton has a finite number of locations and the continuous time is represented by clocks that increase uniformly and that enable, and possibly force, to take transitions to other locations. Depending on the formula to be analyzed, the continuous behavior of the system is first made discrete by applying different discretization techniques (such as regions and zones for timed automata), and then the obtained system is checked against the given property (cf. [26,27]).

Developing similar discretization techniques for quantum systems does not seem to be an easy task. Thus, researchers started to look for models with a finite number of states still able to encode the continuous quantum states. Instead of considering all quantum states of the system to analyze, Gay et al. [39,40] restricted the state space to a set of finitely describable stabilizer states; in addition, they also considered only quantum operations of the Clifford group. In this way, they were able to present an efficient model checker for quantum protocols based on purely classical algorithms; it has been implemented as an automatic tool that can be used to model check quantum communication protocols [41]. This approach, while being useful, is also limited: since it is based on stabilizer states, it does not extend to more general protocols involving non-stabilizer states. Independently from Gay et al., Hung et al. [42,43] adopted a similar approach for the synthesis of quantum circuits. In their work, they use a symbolic reachability analysis to formulate the problem of the quantum logic synthesis problem; in this way, they reduce the original problem to a multiple-valued logic synthesis problem, which simplifies the search space and the complexity of the synthesis problem.

In [44], Feng et al. overcome these limitations. The model for quantum protocols they introduced is defined on top of classical Markov chains: it keeps the classical, discrete state space of a Markov chain and replaces the probability values decorating the transitions between states with super-operators. This means that quantum states and operations are unrestricted, thus it is possible to model general quantum protocols.

We will present the model proposed by [44] in more details in Section 2 but, as an example, consider the simple loop program shown in the left part of Figure 1. This program first applies at line ℓ_0 the $\frac{1}{2}$-amplitude damping channel $\mathcal{A} = \{E_0, E_1\}$, where $E_0 = |0\rangle\langle 0| + \frac{1}{\sqrt{2}}|1\rangle\langle 1|$ and $E_1 = \frac{1}{\sqrt{2}}|0\rangle\langle 1|$, on the quantum system q as initialization. Then, at line ℓ_1 it measures q, by means of the two-outcome projective measurement $M = \lambda_0|0\rangle\langle 0| + \lambda_1|1\rangle\langle 1|$. If the observed outcome is λ_1, then the program applies the Hadamard super-operator $\mathcal{G}$ to q and its outcome is measured again in line ℓ_1. The application of $\mathcal{G}$ is repeated until the outcome λ_0 is observed: the program just terminates at line ℓ_3. The corresponding quantum Markov chain is depicted on the right of Figure 1: the single lines of the program are taken as the classical states of the quantum Markov chain, while the instruction performed on each line is used to decorate the corresponding transition; note that we split the projective measurement M into the two super-operators $\mathcal{E}_0 = \{|0\rangle\langle 0|\}$ and $\mathcal{E}_1 = \{|1\rangle\langle 1|\}$, since there are two possible outcomes λ_0 and λ_1, respectively, leading to two different states/program lines. This simple loop program cannot be represented in the model given by Gay et al., since the $\frac{1}{2}$-amplitude damping channel $\mathcal{A}$ is not an operation available in the Clifford group.

```
$\ell_0$:   $q := \mathcal{A}(q)$;
$\ell_1$:   while $M[q]$ do
$\ell_2$:       $q := \mathcal{G}(q)$;
$\ell_3$:   end
```

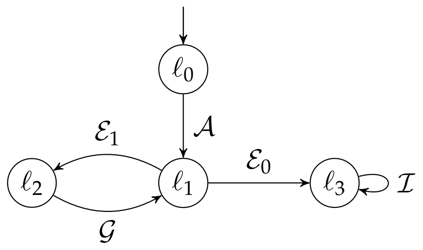

Figure 1. A simple loop program and its model as a quantum Markov chain.

In this paper we use the model of quantum Markov chains, as introduced by [44], to which we refer to for a more detailed comparison with other types of super-operator weighted Markov chains available in literature. This model is well suited to model quantum programs and protocols: as we have seen for the loop program above, representing the given program or protocol as a quantum Markov chain is not so complicated, since the classical states correspond to the single lines/steps of the quantum system, and the transitions are decorated with the (components of the) super-operators.

The main aspect to remember while modelling a quantum protocol in this framework is that the classical states *do not* represent the effect of the quantum operations on the quantum states, but only the current step in the algorithm. For instance, consider the quantum Markov chain shown in Figure 1: state ℓ_1 represents the fact that we applied the operation $\mathcal{A}$ to the input quantum state q and that the next operation is its measurement, with two possible outcomes $\mathcal{E}_0(\mathcal{A}(q))$ and $\mathcal{E}_1(\mathcal{A}(q))$. However, state ℓ_1 does not represent the actual quantum state $\mathcal{A}(q)$ and how it is modified by the different operations, or whether successive visits to ℓ_1 correspond to the same state of the quantum system.

1.3. How Can We Specify the Desired Properties?

Once the quantum protocol has been modelled as a quantum Markov chain, we need to consider how to formalize the properties we want to analyze. For classical model checking problems, properties are expressed as formulas in some given logic; the most common ones are the Computation Tree Logic (CTL) [2], the Linear-time Temporal Logic (LTL) [45], and their combination CTL* [46].

CTL is a branching time logic that focuses on the possible future behaviors of a system, given by the different successors of the current state. This means that whenever we are in a state with multiple successors, we branch our future execution by considering all possible successors, such as the branches of a tree. CTL has two kinds of formulas: state formulas and path formulas. State formulas, as the name suggests, refer to the properties that each single state satisfies: basic facts represented by atomic propositions, conjunctions and negations of states formulas, and quantified path formulas, where the quantification is given by either the universal ($\forall$) or the existential ($\exists$) quantifier on the paths, i.e., the branches of the tree. Path formulas, instead, refer to what happens in the states that are visited along a single path, with different temporal operators characterizing the behavior of the sequence of states visited by the path.

LTL, as the name indicates, is a linear time logic that considers what happens along a single execution of the system, instead of considering all alternatives as in CTL. This means that along the execution, all choices about the successor states have already been resolved. Since LTL focuses on a single, whole execution at a time, it has only one kind of formulas that combine freely Boolean and temporal operators. ω-regular properties extend LTL, by directly specifying the executions that are "good"; ω-regular properties are given as the language of an automaton accepting infinite words (cf. [27]).

Depending on the characteristics of the system, such as the presence of probabilistic choices, rewards, or time information, these logics have been extended and adapted to use specific operators to take care of such additional characteristics. For instance, the probabilistic version of CTL, PCTL [47], replaces the path quantifiers of CTL with a single probability operator that is used to compute the probability of all paths satisfying the given path formula. PLTL [48] is similar: instead of requiring all paths to satisfy the formula, it just computes their probability and then checks whether it is bounded by the given threshold. This is a reasonable choice: in the probabilistic setting, a single path can violate the given path property; however, we can ignore it as long as its probability is negligible.

When moving to quantum systems, the logic also needs to be adapted. In this work, we consider the quantum extension of CTL given in [44] and of LTL as presented in [49], as well as the extension of CTL with a fidelity operator proposed in [50].

The former two logics can be considered the natural extension of PCTL and PLTL to quantum systems: instead of computing the probability of the paths, we compute the super-operator $\mathcal{E}$ accumulated along these paths and then we compare it with the given threshold super-operator $\mathcal{F}$. Such a comparison intuitively represents the fact that the success probability of performing $\mathcal{E}$ is bounded by the probability of performing $\mathcal{F}$, independently from the initial state. The idea underlying this choice is to use the logics to characterize *classical* aspects of the quantum system, given the fact that the quantum protocols usually aim at achieving some classical tasks, such as exchanging encryption keys or flipping coins. As an example of the properties we can express in this logic, we have the formula $\mathbb{Q}_{\gtrsim\mathcal{E}}[\mathbf{F}\ell_3]$. This formula holds whenever the probability that the loop program shown in Figure 1 eventually terminates is at least $\mathcal{E}$, for each initial quantum state. Therefore, to check whether the program terminates almost surely on any input, we can just use the formula $\mathbb{Q}_{\gtrsim\mathcal{I}}[\mathbf{F}\ell_3]$, where $\mathcal{I}$ is the identity super-operator which can be regarded as the value 1 in the context of probabilistic model checking.

Although the quantum counterparts of CTL and LTL seen above are able to quantify classical properties of quantum protocols, they are not able to quantify the similarity between two quantum states. In order to consider this similarity, the authors of [50] adopt the fidelity operator to represent the homonym popular measure used by the quantum community [51–54]. Fidelity can be used, for instance, to measure the effect of noise on the transmission of a quantum bit on a channel: if the received qubit is not affected by the noise, then fidelity reaches its maximal value 1; on the other hand, if the noise flips the qubit, then fidelity obtains its minimal value 0.

To summarize, quantum CTL can be used to formalize the branching-time behavior of the quantum system with respect to classical properties; quantum LTL considers linear-time properties instead of branching time properties, in the same setting; fidelity CTL, on the other hand, focuses on the quantum branching-time effects of the quantum system.

1.4. Organization of the Paper

After recalling from literature the preliminary definitions about quantum Markov chains in Section 2, we present in Section 3 how to model check quantum Markov chains against properties given in the quantum CTL; this section is mainly based on the results of [44]. Then, in Section 4, we consider the problem of checking properties involving the fidelity operator, which provides a means to know how well a super-operator preserves quantum states; we use [50] as reference. We present, in Section 5, how to check LTL and ω-regular properties, with material taken from [49]. Lastly, in Section 6 we provide some final remark about quantum model checking.

We refer the readers interested in the technical details of the different constructions to the literature we used as reference. They can also refer to the recently published book [55], which provides more technical details about model checking quantum systems against the quantum CTL and LTL logics. It also gives a more comprehensive presentation of the basics of quantum theory and the challenges in adapting classical model checking techniques to the quantum setting; for instance, the authors of the book consider also quantum automata, that are well suited to represent closed quantum systems, i.e., quantum systems without interference from the environment. The quantum Markov chains we consider in this paper, instead, can be used to model open quantum systems, i.e., those that are affected by the environment, such as the presence of an eavesdropper trying to intercept the cryptographic key during the execution of a quantum key-exchange protocol. The book [55] is based on the same work available in literature about quantum CTL and LTL, so this paper can be considered also as an introduction, or a summary, for [55]. In this paper, however, we also present the results developed for model checking fidelity properties, not present in [55].

2. Quantum Markov Chains

In this section, we recall from literature the required notions and notations about quantum systems and quantum Markov chains. For a more thorough discussion, we refer the interested reader to, e.g., [44,50,56].

Given a finite dimensional Hilbert space $\mathcal{H}$, let $\mathcal{L}(\mathcal{H})$ be the set of linear operators on it. Let $\mathcal{S}(\mathcal{H})$ be the set of *super-operators*, where a super-operator is a completely positive linear operator on $\mathcal{L}(\mathcal{H})$. We identify two specific super-operators in $\mathcal{S}(\mathcal{H})$, namely $\mathcal{I}_{\mathcal{H}}$ and $0_{\mathcal{H}}$, that are the identity and the null super-operator, respectively. Given a finite set J of indices, for simplicity of presentation, we exchange freely a super-operator and its Kraus representation, that is, for a given super-operator $\mathcal{E}$, we write $\mathcal{E} = \{ E_j \mid j \in J \}$ where $\{ E_j \mid j \in J \}$ is a set of Kraus operators of $\mathcal{E}$; that is, $\mathcal{E}(A) = \sum_{j \in j} E_j A E_j^\dagger$ for all $A \in \mathcal{L}(\mathcal{H})$, where $E_j^\dagger$ is the complex conjugate and transpose of E_j. For any $\mathcal{E}, \mathcal{F} \in \mathcal{S}(\mathcal{H})$, we denote the composition of $\mathcal{E}$ and $\mathcal{F}$ by $(\mathcal{E} \circ \mathcal{F})(A) = \mathcal{E}(\mathcal{F}(A))$. To simplify the notation, we may drop the symbol $\circ$ and just write $\mathcal{E}\mathcal{F}$ instead of $\mathcal{E} \circ \mathcal{F}$.

Super-operators can be (pre-)ordered according to their preservation of the trace. The *trace* of a linear operator $A \in \mathcal{L}(\mathcal{H})$ is defined as $\sum_{i=1}^{d} \langle \eta_i | A | \eta_i \rangle$ where $\{ |\eta_i\rangle \mid 1 \leq i \leq d \}$ is an orthonormal basis of the Hilbert space $\mathcal{H}$ with dimension d. Let $\mathcal{D}(\mathcal{H})$ denote the set of partial density operators in $\mathcal{L}(\mathcal{H})$, where we say that a linear operator $A \in \mathcal{L}(\mathcal{H})$ is a partial density operator if A is positive (i.e., $\langle \eta | A | \eta \rangle \geq 0$ for all $|\eta\rangle \in \mathcal{H}$) and its trace $\mathrm{tr}(A)$ satisfies $\mathrm{tr}(A) \leq 1$; we denote by $\mathcal{D}^1(\mathcal{H})$ the set of density operators, that is, the partial density operators whose trace is 1. Recall that the trace of a partial density operator denotes the probability that the corresponding (normalized) quantum state is reached [57]. Given two super-operators $\mathcal{E}, \mathcal{F} \in \mathcal{S}(\mathcal{H})$, we write $\mathcal{E} \lesssim \mathcal{F}$ if for any $\rho \in \mathcal{D}(\mathcal{H})$, we have $\mathrm{tr}(\mathcal{E}(\rho)) \leq \mathrm{tr}(\mathcal{F}(\rho))$. Intuitively, $\mathcal{E} \lesssim \mathcal{F}$ means that the success probability of performing $\mathcal{E}$ is always at most the probability of performing $\mathcal{F}$, whatever the initial state is. We define $\eqsim$ to be $\lesssim \cap \gtrsim$.

For any $\rho \in \mathcal{D}(\mathcal{H})$, the *support* $\mathrm{supp}(\rho)$ is defined to be the space spanned by the eigenvectors of ρ with non-zero eigenvalues. Let $\{X_k\}$ be a family of subspaces of $\mathcal{H}$. The *join* of $\{X_k\}$ is defined to be $\bigvee_k X_k = \mathrm{span}(\bigcup_k X_k)$.

We denote by $\mathcal{S}^{\mathcal{I}}(\mathcal{H}) = \{ \mathcal{E} \in \mathcal{S}(\mathcal{H}) \mid 0_{\mathcal{H}} \lesssim \mathcal{E} \lesssim \mathcal{I}_{\mathcal{H}} \}$ the set of trace-nonincreasing super-operators over $\mathcal{H}$. In particular, this means that $\mathcal{E} \in \mathcal{S}^{\mathcal{I}}(\mathcal{H})$ if, and only if, for any $\rho \in \mathcal{D}(\mathcal{H})$, we have that $\mathrm{tr}(\mathcal{E}(\rho)) \in [0,1]$. For the reader familiar with the probabilistic setting, it is natural to consider the set $\mathcal{S}^{\mathcal{I}}(\mathcal{H})$ as the quantum counterpart of the probability domain $[0,1]$. This is the underlying idea of the quantum Markov chains defined in [44].

Definition 1. *A* super-operator weighted Markov chain $\mathfrak{M}$ *over a Hilbert space $\mathcal{H}$ is a pair $\mathfrak{M} = (S, \mathbf{Q})$, where*

- *S is a finite set of* classical states*;*
- *$\mathbf{Q}\colon S \times S \to \mathcal{S}^{\mathcal{I}}(\mathcal{H})$ is called the* transition matrix *where for each $s \in S$, the super-operator $\sum_{s' \in S} \mathbf{Q}(s, s')$ is trace-preserving, that is $\sum_{s' \in S} \mathbf{Q}(s, s') \eqsim \mathcal{I}_{\mathcal{H}}$.*

We just refer to this kind of Markov chains as *quantum Markov chains* (QMCs).

Similar to the setting of classical Markov chains, to talk about the (probability) measure of different events in a quantum Markov chain, we need to introduce a measurable space and the measure we want to use. The basic events of the measurable space will be based on the notion of path.

Given a QMC $\mathfrak{M} = (S, \mathbf{Q})$, we call a finite or infinite sequence $\sigma = s_0 s_1 \cdots$ of states in S a *path* of $\mathfrak{M}$ if, for each valid index $i \geq 1$, we have $\mathbf{Q}(s_{i-1}, s_i) \neq 0_{\mathcal{H}}$. We denote by $\mathrm{Path}^{\mathfrak{M}}_{\mathit{fin}}$ the set of finite paths and by $\mathrm{Path}^{\mathfrak{M}}$ the set of infinite paths. We let $\sigma[i]$ denote the state s_i.

The basic events of our measurable space are the cylinder sets of finite paths. We define the *cylinder set* of the finite path $\sigma = s_0 s_1 \cdots s_n$ to be the set $Cyl(\sigma) = \{\, \sigma' \in \mathrm{Path}^{\mathfrak{M}} \mid \forall 0 \leq i \leq n : \sigma[i] = \sigma'[i] \,\}$. In practice, the cylinder set of a finite path σ contains all possible infinite paths that are an extension of σ.

Let $(\mathrm{Path}^{\mathfrak{M}}, \Sigma)$ be a measurable space where Σ is the σ-algebra generated by all the cylinder sets $Cyl(\sigma)$ where $\sigma \in \mathrm{Path}^{\mathfrak{M}}_{\mathit{fin}}$. For any $s \in S$, we define the measure $Q^{\mathfrak{M}}_s \colon \mathrm{Path}^{\mathfrak{M}}_{\mathit{fin}} \to \mathcal{S}(\mathcal{H})$ as follows. Given the finite path $\sigma = s_0 s_1 \cdots s_n$,

$$Q^{\mathfrak{M}}_s(s_0 s_1 \cdots s_n) = \begin{cases} 0_{\mathcal{H}} & \text{if } s_0 \neq s; \\ \mathcal{I}_{\mathcal{H}} & \text{if } s_0 = s \text{ and } n = 0; \\ \mathbf{Q}(s_{n-1}, s_n)\mathbf{Q}(s_{n-2}, s_{n-1}) \cdots \mathbf{Q}(s_0, s_1) & \text{if } s_0 = s \text{ and } n > 0. \end{cases}$$

Then, $Q^{\mathfrak{M}}_s$ induces a (super-operator valued) measure on $(\mathrm{Path}^{\mathfrak{M}}, \Sigma)$, also denoted by $Q^{\mathfrak{M}}_s$ for simplicity, by setting $Q^{\mathfrak{M}}_s(Cyl(\sigma)) = Q^{\mathfrak{M}}_s(\sigma)$. According to [44] (Theorem 3.2), this measure is unique up to $\eqsim$.

In the reminder of the paper, we are interested in analyzing properties of a given QMC. We assign labels to the states of the QMC to indicate the basic properties that hold in each state, that is, we decorate each state of the QMC with the atomic propositions it satisfies. Atomic propositions represent some simple known facts we want to express about the single states of the system, such as "the execution is terminated", "the measured qubits are different", "eavesdropping detected", and so on. Thus, we consider an extension of QMCs in which each state obtains assigned atomic propositions by a labelling function.

Definition 2. *A* labelled quantum Markov chain *(LQMC) is a tuple $\mathfrak{M} = (S, \mathbf{Q}, AP, L)$ where $(S, \mathbf{Q})$ is a QMC and*

- *AP is a finite set of* atomic propositions *and*
- *$L \colon S \to 2^{AP}$ is a* labelling function.

The notions of paths, measures, etc. given above extend in the natural way to LQMCs; for the labelling from states to paths, we set $L(s_0 s_1 s_2 \cdots) = L(s_0)L(s_1)L(s_2) \cdots$.

3. Model Checking CTL Properties

As we have seen in the introduction, the branching time logic CTL focuses on the possible future behaviors of a system, given by the different successors of the current state, instead of a single execution as in LTL and ω-regular properties. CTL has two kinds of formulas: state formulas and path formulas. State formulas, as the name suggests, refer to the properties that the single states satisfy: basic facts represented by atomic propositions, conjunctions and negations of states formulas, and quantified path formulas, where the quantification is given by either the universal ($\forall$) or the existential ($\exists$) quantifier. Path formulas, instead, refer to what happens to the states that are visited along a single paths. This "state-based" nature of CTL makes the CTL model checking problem, i.e., whether a system satisfies a given CTL formula, rather easy and intuitive, which results in the development of efficient algorithms; see, e.g., [26,27] for more details. The quantum counterpart of CTL, QCTL [44], replaces the path quantifiers $\forall$ and $\exists$ with a single quantum quantifier $\mathbb{Q}_{\sim\mathcal{E}}$, which accumulates the super-operators corresponding to the paths satisfying the given path formula, and then compares it with $\mathcal{E}$. This is similar to the probabilistic CTL, where the path quantifiers are replaced by the probabilistic operator $\mathbb{P}_{\sim p}$ computing the overall probability of the satisfying paths and comparing it with p.

3.1. The Model Checking Problem

We start with the LQMC $\mathfrak{M}$ and the property φ we want to verify. The property is usually given as a QCTL state formula φ, that is, a formula expressible according to the following grammar:

$$\begin{aligned}\varphi &::= a \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathbb{Q}_{\sim\mathcal{E}}[\psi]\\ \psi &::= \mathbf{X}\varphi \mid \varphi\,\mathbf{U}^{\leq k}\,\varphi \mid \varphi\,\mathbf{U}\,\varphi,\end{aligned}$$

where $a \in AP$ is taken from the given finite set of atomic propositions, $\sim \in \{\lesssim, \gtrsim, \eqsim\}$, $\mathcal{E} \in \mathcal{S}^{\mathcal{I}}(\mathcal{H})$, and $k \in \mathbb{N}$ is a bound. The ψ formulas $\mathbf{X}\varphi$, $\varphi_1\,\mathbf{U}^{\leq k}\,\varphi_2$, and $\varphi_1\,\mathbf{U}\,\varphi_2$ are usually referred to as path formulas. In addition to the usual Boolean operators $\wedge$ and $\neg$, the QCTL logic makes use of the temporal operators Next $\mathbf{X}$, Until $\mathbf{U}$, and bounded Until $\mathbf{U}^{\leq k}$. We freely use the usual syntactic sugar like the Boolean operators **false**, **true**, $\vee$, $\Rightarrow$, and $\Leftrightarrow$ and the temporal operator Finally/Eventually $\mathbf{F}\varphi = \mathbf{true}\,\mathbf{U}\,\varphi$. In literature, it is possible to find the temporal operators $\bigcirc$ and $\Diamond$, corresponding to $\mathbf{X}$ and $\mathbf{F}$, respectively.

The semantics of QCTL is given by means of the satisfaction relation $\models$, defined as follows. For state formulas, given a state $s \in S$, we have

$$\begin{aligned} s &\models a && \text{if } a \in L(s);\\ s &\models \varphi_1 \wedge \varphi_2 && \text{if } s \models \varphi_1 \text{ and } s \models \varphi_2;\\ s &\models \neg\varphi && \text{if it is not the case that } s \models \varphi; \text{ and}\\ s &\models \mathbb{Q}_{\sim\mathcal{E}}[\psi] && \text{if } Q^{\mathfrak{M}}_s(\{\,\sigma \in \mathrm{Path}^{\mathfrak{M}} \mid \sigma \models \psi\,\}) \sim \mathcal{E}.\end{aligned}$$

For path formulas, given a path $\sigma \in \mathrm{Path}^{\mathfrak{M}}$, we have

$$\begin{aligned} \sigma &\models \mathbf{X}\varphi && \text{if } \sigma[1] \models \varphi;\\ \sigma &\models \varphi_1\,\mathbf{U}^{\leq k}\,\varphi_2 && \text{if } \exists 0 \leq n \leq k : \sigma[n] \models \varphi_2 \text{ and } \forall 0 \leq i < n : \sigma[i] \models \varphi_1;\\ \sigma &\models \varphi_1\,\mathbf{U}\,\varphi_2 && \text{if } \exists n \in \mathbb{N} : \sigma[n] \models \varphi_2 \text{ and } \forall 0 \leq i < n : \sigma[i] \models \varphi_1.\end{aligned}$$

The model checking problem for LQMCs against a QCTL state formula φ is formalized as follows.

Definition 3. *Let $\mathfrak{M}$ be an LQMC, $s \in S$ be a state, and φ be a QCTL state formula. The model checking problem for $\mathfrak{M}$ against φ asks to verify whether $s \models \varphi$ holds.*

3.2. The Standard Bottom-Up Approach

As said previously, the QCTL properties are kind of "state-based" properties. The algorithm for checking whether an LQMC $\mathfrak{M}$ satisfies a given QCTL state property φ mimics the similar algorithms developed for CTL and PCTL: the state formula φ is analyzed from bottom to top, and for each of its state subformulas, the set of states satisfying the subformula is computed. When a quantified path formula needs to be analyzed, different techniques are used according to the underlying temporal operator.

Let $Sat(\varphi)$ denote the set of states satisfying the QCTL state formula φ, that is, $Sat(\varphi) = \{\,s \in S \mid s \models \varphi\,\}$. The set $Sat(\varphi)$ can be computed recursively on the structure of φ as follows.

$$\begin{aligned} Sat(a) &= \{\,s \in S \mid a \in L(s)\,\}\\ Sat(\varphi_1 \wedge \varphi_2) &= Sat(\varphi_1) \cap Sat(\varphi_2)\\ Sat(\neg\varphi) &= S \setminus Sat(\varphi)\\ Sat(\mathbb{Q}_{\sim\mathcal{E}}[\psi]) &= \{\,s \in S \mid Q^{\mathfrak{M}}_s(\{\,\sigma \in \mathrm{Path}^{\mathfrak{M}} \mid \sigma \models \psi\,\}) \sim \mathcal{E}\,\}\end{aligned}$$

The above algorithm for computing the satisfaction set $Sat(\varphi)$ of a QCTL state formula φ relies on the computation of the super-operator that is $\eqsim$-equivalent to $Q_s^{\mathfrak{M}}$ and then $\sim$-compares it with $\mathcal{E}$. The computation of $Q_s^{\mathfrak{M}}$ depends on the actual path formula that needs to be satisfied; to simplify the notation, below we just write $Q_s^{\mathfrak{M}}(\psi)$ to mean $Q_s^{\mathfrak{M}}(\{\, \sigma \in \mathrm{Path}^{\mathfrak{M}} \mid \sigma \models \psi \,\})$.

Consider the Next operator $\mathbf{X}$. The computation of $Q_s^{\mathfrak{M}}(\mathbf{X}\varphi)$ is rather easy: by hypothesis, we already have computed $Sat(\varphi)$, so $Q_s^{\mathfrak{M}}(\mathbf{X}\varphi)$ reduces to compute the overall super-operator leading to the successors of s that belong to $Sat(\varphi)$, namely,

$$Q_s^{\mathfrak{M}}(\mathbf{X}\varphi) \eqsim \sum_{s' \in Sat(\varphi)} \mathbf{Q}(s, s').$$

Consider, now, the bounded Until operator $\mathbf{U}^{\leq k}$. The computation of $Q_s^{\mathfrak{M}}(\varphi_1 \,\mathbf{U}^{\leq k}\, \varphi_2)$ is a bit more involved: since k is a given finite natural number, we can just consider $\mathfrak{M}$ and unfold it k times; then, we accumulate the super-operators layer by layer, as long as the states satisfy the conditions given by the semantic of $\mathbf{U}^{\leq k}$. Formally, we have that $Q_s^{\mathfrak{M}}(\varphi_1 \,\mathbf{U}^{\leq k}\, \varphi_2)$ is $\eqsim$-equivalent to

$$\begin{cases} \mathcal{I}_{\mathcal{H}} & \text{if } s \in Sat(\varphi_2); \\ \sum_{s' \in S} Q_{s'}^{\mathfrak{M}}(\varphi_1 \,\mathbf{U}^{\leq k-1}\, \varphi_2)\mathbf{Q}(s, s') & \text{if } k > 0 \text{ and } s \in Sat(\varphi_1) \setminus Sat(\varphi_2); \\ 0_{\mathcal{H}} & \text{otherwise.} \end{cases}$$

Intuitively, this is essentially the application of the Next $\mathbf{X}$ operator k times, while checking the satisfaction of φ_1 and φ_2, since $\varphi_1 \,\mathbf{U}^{\leq k}\, \varphi_2$ is equivalent to φ_2 when $k = 0$ and to $\varphi_2 \vee (\varphi_1 \wedge \mathbf{X}(\varphi_1 \,\mathbf{U}^{\leq k-1}\, \varphi_2))$ when $k \geq 1$.

The last case, the (unbounded) Until operator $\mathbf{U}$, is more challenging: intuitively, it is just the limit for k to infinity of the bounded case. However, a naïve computation of such a limit is likely to be inefficient, so it is preferable to adopt a different approach, such as the one underlying the PCTL model checking algorithm that is based on a fixed point computation. To this end, we partition the state space S into three sets: $S^{\mathcal{I}_{\mathcal{H}}}$, with all states that we are sure satisfy $\varphi_1 \,\mathbf{U}\, \varphi_2$; we can assign the super-operator $\mathcal{I}_{\mathcal{H}}$ to these states. $S^{0_{\mathcal{H}}}$, containing the states that we are sure do not satisfy $\varphi_1 \,\mathbf{U}\, \varphi_2$; we can assign the super-operator $0_{\mathcal{H}}$ to these states. $S^{?}$, for the remaining states for which we do not know whether they satisfy $\varphi_1 \,\mathbf{U}\, \varphi_2$ and for which we cannot assign a predefined super-operator. States that belong to $S^{\mathcal{I}_{\mathcal{H}}}$ are clearly those in $Sat(\varphi_2)$, as well as those in $Sat(\varphi_1)$ that cannot avoid to reach $S^{\mathcal{I}_{\mathcal{H}}}$ in one step. Symmetrically, it is certain that the states in $S \setminus (Sat(\varphi_1) \cup Sat(\varphi_2))$ belong to $S^{0_{\mathcal{H}}}$, since they clearly violate $\varphi_1 \,\mathbf{U}\, \varphi_2$; moreover, also the states that have no way to reach $Sat(\varphi_2)$ in the underlying graph can be included in $S^{0_{\mathcal{H}}}$.

Regarding the super-operator to assign at the states in $S^{?}$, by [44] (Theorem 5.1) it is the least fixed point of the function $f(X) = X\mathcal{T} + \mathcal{G}$, where $\mathcal{T}$ and $\mathcal{G}$ are two matrices indexed by the states in $S^{?}$ and whose entries depend on $\mathbf{Q}$. More formally, Theorem 5.1 of [44] shows that, given the matrices $\mathcal{T} = (\mathbf{Q}(s', s))_{s,s' \in S^{?}}$ of size $|S^{?}| \times |S^{?}|$ and $\mathcal{G} = (\sum_{s' \in S^{\mathcal{I}_{\mathcal{H}}}} \mathbf{Q}(s, s'))_{s \in S^{?}}$ of size $1 \times |S^{?}|$, it holds that $Q_s^{\mathfrak{M}}(\varphi_1 \,\mathbf{U}\, \varphi_2) \eqsim \widetilde{\mathcal{E}}_s$, where $\widetilde{\mathcal{E}}$ is the least fixed point of $f(X) = X\mathcal{T} + \mathcal{G}$ and $\widetilde{\mathcal{E}}_s$ is the entry of $\widetilde{\mathcal{E}}$ relative to the state s. The matrix $\mathcal{G}$ represents the super-operators that lead each state $s \in S^{?}$ to the goal set $S^{\mathcal{I}_{\mathcal{H}}}$ in one step; the matrix $\mathcal{T}$ instead contains the super-operators that in one step lead each state $s \in S^{?}$ to other states also in $S^{?}$. Note that we do not need to consider the super-operators leading to $S^{0_{\mathcal{H}}}$ since they would lead to a failure in satisfying the formula $\varphi_1 \,\mathbf{U}\, \varphi_2$. Detailed algorithms for the computation of $Q_s^{\mathfrak{M}}(\varphi_1 \,\mathbf{U}\, \varphi_2)$, based on Jordan decomposition or matrix inversion, can be found in [49,50].

Example 1. *As an example of application of quantum CTL, consider the QMC shown in Figure 1 and its labelled counterpart where each classical state ℓ_i obtains label ℓ_i, that is, we only assign the state name as the label for the state. As formula, consider $\mathbb{Q}_{\gtrsim \mathcal{I}_{\mathcal{H}}}[(\neg \ell_2) \,\mathbf{U}\, \ell_3]$, which requires to check* *whether the super-operator corresponding to the paths that eventually reach ℓ_3 without passing through ℓ_2 is at least $\mathcal{I}_{\mathcal{H}}$. The algorithm for model checking this property works as follows: we start from the state properties in the leaves of the parsing tree of $\mathbb{Q}_{\gtrsim \mathcal{I}_{\mathcal{H}}}[(\neg \ell_2)\,\mathbf{U}\,\ell_3]$, that is, ℓ_2 and ℓ_3, and we compute $Sat(\ell_2) = \{\ell_2\}$ and $Sat(\ell_3) = \{\ell_3\}$. Then, we compute $Sat(\neg \ell_2) = \{\ell_0, \ell_1, \ell_3\}$. Now we can compute the super-operator associated with the paths satisfying the path formula $(\neg \ell_2)\,\mathbf{U}\,\ell_3$.*

We initialize the sets $S^{\mathcal{I}_{\mathcal{H}}}$, $S^{0_{\mathcal{H}}}$, and $S^{?}$ as $S^{\mathcal{I}_{\mathcal{H}}} = \{\ell_3\}$, $S^{0_{\mathcal{H}}} = \{\ell_2\}$, and $S^{?} = \{\ell_0, \ell_1\}$. The Bellman equations for the states in $S^{?}$ are the following, once we plug in the known values $0_{\mathcal{H}}$ and $\mathcal{I}_{\mathcal{H}}$ for the states in $S^{0_{\mathcal{H}}}$ and $S^{\mathcal{I}_{\mathcal{H}}}$, respectively: $X_{\ell_0} \eqsim X_{\ell_1}\mathcal{A}$ and $X_{\ell_1} \eqsim \mathcal{E}_0$, which correspond to the least fixed point of the function $f(X) = X\mathcal{T} + \mathcal{G}$, where

$$\mathcal{T} = \begin{pmatrix} 0_{\mathcal{H}} & 0_{\mathcal{H}} \\ \mathcal{A} & 0_{\mathcal{H}} \end{pmatrix} \qquad\qquad \mathcal{G} = \begin{pmatrix} 0_{\mathcal{H}} & \mathcal{E}_0 \end{pmatrix}$$

used in [44] (Theorem 5.1). The final result is $X_{\ell_0} \eqsim \mathcal{E}_0\mathcal{A}$ and $X_{\ell_1} \eqsim \mathcal{E}_0$, for which we have $X_{\ell_0} \not\gtrsim \mathcal{I}_{\mathcal{H}}$, as well as $X_{\ell_1} \not\gtrsim \mathcal{I}_{\mathcal{H}}$. Thus, we obtain $Sat(\mathbb{Q}_{\gtrsim \mathcal{I}_{\mathcal{H}}}[(\neg \ell_2)\,\mathbf{U}\,\ell_3]) = \{\ell_3\}$, since ℓ_3 is the only state from which the computed super-operator is at least $\mathcal{I}_{\mathcal{H}}$.

3.3. Complexity of the QCTL Model Checking Problem

As said at the beginning of this section, textbooks like [26,27] provide a detailed analysis of the model checking algorithms for CTL and PCTL, and they show that such algorithms are efficient. More precisely, such algorithms are polynomial in both the size of the model $\mathfrak{M}$ and the size of the state formula φ; the size $|\varphi|$ of φ is defined [47] as the number of logical connectives and temporal operators in φ plus the sum of the size $\log(k)$ of each bounded temporal operators $\mathbf{U}^{\leq k}$ occurring in φ.

This polynomial complexity carries over the QCTL model checking problem: according to [44], the complexity is polynomial in the size of $\mathfrak{M}$, of φ, and the dimension of the Hilbert space $\mathcal{H}$.

4. Model Checking Fidelity Properties

The quantum CTL logic we have considered in the previous sections focus on the probability of events, as carried by the computed super-operator. For instance, when model checking an LQMC against a CTL formula $\mathbb{Q}_{\sim \mathcal{E}}[\psi]$, we solve the problem to check whether $Q_s^{\mathfrak{M}}(\{\, \sigma \in \mathrm{Path}^{\mathfrak{M}} \mid L(\sigma) \in \mathit{Words}(\psi) \,\}) \sim \mathcal{E}$ for some given $\sim \in \{\lesssim, \gtrsim, \eqsim\}$ and $\mathcal{E} \in \mathcal{S}^{\mathcal{I}}(\mathcal{H})$ (cf. Definition 3). This means, more precisely, to check whether, for each $\rho \in \mathcal{D}(\mathcal{H})$, the trace of $Q_s^{\mathfrak{M}}(\{\, \sigma \in \mathrm{Path}^{\mathfrak{M}} \mid L(\sigma) \in \mathit{Words}(\psi) \,\})(\rho)$ is bounded by the trace of $\mathcal{E}(\rho)$.

Quantum CTL is not able to characterize the effect of the super-operators on the quantum states, unless this is reflected on the probability value of their trace. This means, for instance, that the effect of a measurement can be recognized, since the associated probability value is likely to decrease; on the other hand, a bit flip remains undetected, since the whole probability mass is preserved.

To express properties about how well the super-operator modelled by the QMC preserves the quantum state it is applied on a different operator is needed, such as a fidelity operator. The logic we consider is the *Fidelity CTL logic* (FCTL) proposed in [50], also based on CTL. (The authors of [50] call this logic *Quantum CTL*. Instead, we call it Fidelity CTL to avoid confusion with the homonym logic in Section 3.) As such, this logic is essentially the same as the classical and probabilistic CTL from literature, and QCTL considered in Section 3. The only difference is on the operator wrapping the path formulas: instead of using the universal $\forall$, existential $\exists$, probabilistic $\mathbb{P}_{\sim p}$, or quantum $\mathbb{Q}_{\sim \mathcal{E}}$ operators, FCTL uses the fidelity operator $\mathbb{F}_{\sim \tau}$, to check whether the fidelity is bounded by τ. (For uniformity of notation, we use the operator $\mathbb{F}_{\sim \tau}$ instead of the operator $\mathfrak{F}_{\sim \tau}$ that is used in [50]).

4.1. The Model Checking Problem

Similarly to the other model checking problems, we are given the LQMC $\mathfrak{M}$ and the property φ we want to verify. The property is usually provided as an FCTL state formula φ respecting the following grammar:

$$\begin{aligned}\varphi &::= a \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathbb{F}_{\sim\tau}[\psi] \\ \psi &::= \mathbf{X}\varphi \mid \varphi \,\mathbf{U}^{\leq k}\, \varphi \mid \varphi \,\mathbf{U}\, \varphi,\end{aligned}$$

where $a \in AP$, $\sim\, \in \{\leq, \geq\}$, $\tau \in \mathbb{Q} \cap [0,1]$ is a threshold, and $k \in \mathbb{N}$. Note that all other comparisons, such as $<$, $>$, $=$, and $\neq$, that could be used as $\sim$ in $\mathbb{F}_{\sim\tau}[\psi]$ can be derived from $\leq$ and $\geq$ with the help of conjunction and negation.

As we can see, the only difference between QCTL and FCTL is that $\mathbb{Q}_{\sim\mathcal{E}}[\psi]$ has been replaced with $\mathbb{F}_{\sim\tau}[\psi]$. This means that also the semantics of FCTL differs from the one of QCTL only for this path quantifier. We give the full semantics of FCTL for completeness. Given a state $s \in S$ and a path $\sigma \in \mathrm{Path}^{\mathfrak{M}}$, we have

$$\begin{aligned}
s &\models a && \text{if } a \in L(s); \\
s &\models \varphi_1 \wedge \varphi_2 && \text{if } s \models \varphi_1 \text{ and } s \models \varphi_2; \\
s &\models \neg\varphi && \text{if it is not the case that } s \models \varphi; \\
s &\models \mathbb{F}_{\sim\tau}[\psi] && \text{if } \underline{\mathrm{Fid}}(Q_s^{\mathfrak{M}}(\{\, \sigma \in \mathrm{Path}^{\mathfrak{M}} \mid \sigma \models \psi \,\})) \sim \tau; \\
\sigma &\models \mathbf{X}\varphi && \text{if } \sigma[1] \models \varphi; \\
\sigma &\models \varphi_1 \,\mathbf{U}^{\leq k}\, \varphi_2 && \text{if } \exists 0 \leq n \leq k : \sigma[n] \models \varphi_2 \text{ and } \forall 0 \leq i < n : \sigma[i] \models \varphi_1; \text{ and} \\
\sigma &\models \varphi_1 \,\mathbf{U}\, \varphi_2 && \text{if } \exists n \in \mathbb{N} : \sigma[n] \models \varphi_2 \text{ and } \forall 0 \leq i < n : \sigma[i] \models \varphi_1.
\end{aligned}$$

Given a super-operator $\mathcal{E}$, the $\underline{\mathrm{Fid}}$ operator is defined as

$$\underline{\mathrm{Fid}}(\mathcal{E}) = \min_{\rho \in \mathcal{D}^1(\mathcal{H})} \mathrm{Fid}(\mathcal{E}, \rho)$$

where

$$\mathrm{Fid}(\mathcal{E}, \rho) = \sqrt{\rho^{1/2}\mathcal{E}(\rho)\rho^{1/2}}.$$

When ρ is a pure state $\rho = |\eta\rangle\langle\eta|$, then $\mathrm{Fid}(\mathcal{E}, |\eta\rangle\langle\eta|) = \sqrt{\langle\eta|\,\mathcal{E}(|\eta\rangle\langle\eta|)\,|\eta\rangle}$; moreover, by the joint concavity property (cf. [56] (Exercise 9.19)), it holds that $\min_{\rho \in \mathcal{D}^1(\mathcal{H})} \mathrm{Fid}(\mathcal{E}, \rho) = \min_{|\eta\rangle \in \mathcal{H}} \mathrm{Fid}(\mathcal{E}, |\eta\rangle\langle\eta|)$.

The model checking problem for LQMCs against an FCTL state formula φ is the natural counterpart of the one against QCTL (cf. Definition 3).

Definition 4. *Let $\mathfrak{M}$ be an LQMC, $s \in S$ be a state, and φ be an FCTL state formula. The model checking problem for $\mathfrak{M}$ against φ asks to verify whether $s \models \varphi$ holds.*

4.2. The Standard Bottom–Up Approach

As one can reasonably expect, the model checking algorithm for FCTL is analogous to the one for QCTL: it follows a bottom-up approach to compute the sets $Sat(\varphi')$ for each the state subformula φ' occurring in the formula φ we need to verify. Everything is the same, except for the evaluation of $Sat(\mathbb{F}_{\sim\tau}[\psi])$. For this, first the matrix representation of the super-operator $\mathcal{E}_s = Q_s^{\mathfrak{M}}(\{\, \sigma \in \mathrm{Path}^{\mathfrak{M}} \mid \sigma \models \psi \,\})$ is obtained, which corresponds to the analogous computation for QCTL. Then, the expression $\underline{\mathrm{Fid}}(\mathcal{E}_s) \sim \tau$ is evaluated by converting it to the quantified constraint $\mathtt{Q}_{\sim}\, |\eta\rangle \in \mathcal{H} : \mathrm{Fid}(\mathcal{E}_s, |\eta\rangle\langle\eta|) \sim \tau$, where $\mathtt{Q}_{\leq} = \exists$ and $\mathtt{Q}_{\geq} = \forall$. The resulting constraint, after some appropriate manipulation, is then verified by means of the existential theory of the reals.

Example 2. *As an example of the application of the fidelity operator, consider the QMC for the loop program depicted in Figure 1 and the fidelity formula $\mathbb{F}_{\geq 1}[(\neg\ell_2)\,\mathbf{U}\,\ell_3]$. We have already computed* *in Example 1 the super-operators X_{ℓ_i} corresponding to the path formula $(\neg\ell_2)\,\mathbf{U}\,\ell_3$ for each state ℓ_i, that is, $X_{\ell_0} \approx \mathcal{E}_0\mathcal{A}$, $X_{\ell_1} \approx \mathcal{E}_0$, $X_{\ell_2} \approx 0_{\mathcal{H}}$, and $X_{\ell_3} \approx \mathcal{I}_{\mathcal{H}}$. The formula $\mathbb{F}_{\geq 1}[(\neg\ell_2)\,\mathbf{U}\,\ell_3]$ holds in the state ℓ_i if the corresponding super-operator X_{ℓ_i} is such that $\min_{|\eta\rangle\in\mathcal{H}} \mathrm{Fid}(X_{\ell_i}, |\eta\rangle\langle\eta|) \geq 1$; this holds only when $X_{\ell_i} \approx \mathcal{I}_{\mathcal{H}}$ (cf. [50]), which is true only for ℓ_3.*

The technical details for the constructions presented in this section, as well as some improvements on the computation of $Q^{\mathfrak{M}}_s(\{\,\sigma \in \mathrm{Path}^{\mathfrak{M}} \mid \sigma \models \psi\,\})$, are rather involved; the interested reader can find them in [50] (Section 5), together with examples showing their application.

4.3. Complexity of the FCTL Model Checking Problem

Since FCTL differs from CTL, PCTL, and QCTL only for one operator, the detailed complexity analysis available in literature (see, e.g., [26,27,58]) for this kind of logic carries over to FCTL [50]. However, the presence of the fidelity operator $\mathbb{F}_{\sim\tau}$ increases the resulting complexity from polynomial to exponential [50] (Theorem 5.12).

5. Model Checking LTL and ω-Regular Properties

As we mentioned in the introduction, LTL and ω-regular properties are well suited for describing the behavior of a system in the long run, such as its liveness (e.g., the system never crashes), repeated reachability (e.g., the system is ready to accept requests again and again), and fairness (e.g., if the system has to make a choice infinitely often, all options are chosen infinitely often) properties. This kind of properties has been studied extensively for classical Markov chains [27,59–61]. In the classical setting, the usual approach to compute the probability $\mathcal{P}(\varphi)$ that a certain LTL property φ is satisfied in a Markov chain $\mathfrak{M}$ is to adopt an automaton-based approach, as the one introduced by Vardi and Wolper for program verification [62]. This approach works as follows; see, e.g., [26,27] for more details.

5.1. The Model Checking Problem

We start with the model $\mathfrak{M}$ and the property φ we want to verify. The property is usually given as an LTL formula, that is, a formula expressible according to the following grammar:

$$\varphi ::= a \mid \varphi \wedge \varphi \mid \neg\varphi \mid \mathbf{X}\varphi \mid \varphi\,\mathbf{U}\,\varphi,$$

where $a \in AP$ is taken from the given finite set of atomic propositions. In addition, the syntactic sugar shared with CTL, we also consider the Globally/Always temporal operator $\mathbf{G}\varphi = \neg\mathbf{F}\neg\varphi$, sometimes written as $\Box\varphi$.

The semantics of LTL is analogous to the one of CTL, in particular for the temporal operators. We call *word* an infinite sequence of sets of atomic propositions $w \in (2^{AP})^\omega$. We say that w *satisfies* φ, denoted by $w \models \varphi$, in the following cases:

$$\begin{aligned}
w &\models a && \text{if } a \in w[0];\\
w &\models \varphi_1 \wedge \varphi_2 && \text{if } w \models \varphi_1 \text{ and } w \models \varphi_2;\\
w &\models \neg\varphi && \text{if it is not the case that } w \models \varphi;\\
w &\models \mathbf{X}\varphi && \text{if } w[1\cdots] \models \varphi; \text{ and}\\
w &\models \varphi_1\,\mathbf{U}\,\varphi_2 && \text{if } \exists n \in \mathbb{N} : w[n\cdots] \models \varphi_2 \text{ and } \forall 0 \leq i < n : w[i\cdots] \models \varphi_1,
\end{aligned}$$

where $w[i\cdots] = w[i]w[i+1]\cdots$. We denote the set of all words satisfying φ by $Words(\varphi) = \{\,w \in (2^{AP})^\omega \mid w \models \varphi\,\}$.

The model checking problem for LQMCs against an LTL formula φ is formalized as follows.

Definition 5. *Let $\mathfrak{M}$ be an LQMC, $s \in S$ be a state, and φ be an LTL formula. Given $\sim \in \{\lesssim, \gtrsim, \approx\}$ and $\mathcal{E} \in \mathcal{S}^{\mathcal{I}}(\mathcal{H})$, the model checking problem for $\mathfrak{M}$ against φ asks to verify whether $Q^{\mathfrak{M}}_s(\{\,\sigma \in \mathrm{Path}^{\mathfrak{M}} \mid L(\sigma) \in Words(\varphi)\,\}) \sim \mathcal{E}$ holds.*

The model checking problem for LQMCs against an ω-regular property $\mathcal{W} \subseteq (2^{AP})^{\omega}$ is defined analogously.

Definition 6. *Let $\mathfrak{M}$ be an LQMC, $s \in S$ be a state, and $\mathcal{W} \subseteq (2^{AP})^{\omega}$ be an ω-regular property. Given $\sim \in \{\lesssim, \gtrsim, \eqsim\}$ and $\mathcal{E} \in \mathcal{S}^{\mathcal{I}}(\mathcal{H})$, the model checking problem for $\mathfrak{M}$ against $\mathcal{W}$ asks to verify whether $Q_s^{\mathfrak{M}}(\{\, \sigma \in \mathrm{Path}^{\mathfrak{M}} \mid L(\sigma) \in \mathcal{W} \,\}) \sim \mathcal{E}$ holds.*

5.2. The Standard Automata-Based Approach

In the standard automata-based approach [62] to LTL model checking in the context of classical Markov chains, the first step requires to transform the given LTL formula φ into an equivalent nondeterministic Büchi automaton accepting the same language as φ, i.e., $Words(\varphi)$; such an automaton always exists (cf. [27] (Theorem 5.37)). Since nondeterministic Büchi automata are as expressive as ω-regular languages (cf. [27] (Theorem 4.32)), we can also solve the model checking problem against ω-regular properties, by the same approach.

The nondeterminism present in the Büchi automaton, however, does not play well with the computation of the probability measure. For this motivation, the Büchi automaton needs to be made deterministic; however, deterministic Büchi automata are strictly less expressive than nondeterministic ones (cf. [27] (Theorem 4.50)), so more powerful deterministic automata need to be used. Therefore, the nondeterministic Büchi automaton is transformed into an equivalent deterministic automaton $\mathfrak{A}$ with a more complex acceptance condition, such as Rabin, Streett, Muller, or parity (cf. [63–65]). The construction of such deterministic automata usually makes use of a variant of the Safra's [66] approach, see for instance [67,68], or some of the more recent Safraless constructions, such as the ones proposed in [69–71].

Once the deterministic automaton $\mathfrak{A}$ is available, it can be combined with the Markov chain $\mathfrak{M}$ as the product $\mathfrak{M} \otimes \mathfrak{A}$; this product is in practice a Markov chain extended with the information about the paths that are accepted by the automaton $\mathfrak{A}$ (i.e., the paths that satisfy the formula φ); we call such a product a parity Markov chain, when $\mathfrak{A}$ is a parity automaton. As the last step, the probability of the accepting paths is computed. Since all accepting paths need to be infinite, the finite number of states of $\mathfrak{M} \otimes \mathfrak{A}$ implies that such paths need to become trapped into a set of states from which they cannot escape, that is, the states that form a *bottom strongly connected component* (BSCC). Intuitively, a BSCC is composed by states that can be reached by each other, and all their successors are also in the same BSCC. These BSCCs can be computed by means of algorithms operating on the graph structure of the product Markov chain; the outcome of these algorithms is a classification of the states of $\mathfrak{M} \otimes \mathfrak{A}$ as those belonging to a BSCC and the *transient states*, i.e., all other states. Since only states in a BSCC can be visited infinitely often with strictly positive probability, the acceptance condition provided by $\mathfrak{A}$ is used to mark each BSCC as either accepting or rejecting. As the last step, the probability that φ holds in a given state s of the Markov chain $\mathfrak{M}$ is computed by finding the unique solution of a systems of equations: if s belongs to an accepting BSCC, the probability p_s is 1; if s belongs to a rejecting BSCC, the probability p_s is 0; if s is a transient state, then the probability p_s satisfies the Bellman equation $p_s = \sum_{s' \in S} \mathbf{P}(s, s') \cdot p_{s'}$, where $\mathbf{P}(s, s')$ is the probability in $\mathfrak{M}$ to make a transition from s to s', i.e., the probabilistic counterpart of $\mathbf{Q}$ in quantum Markov chains.

5.3. The Standard Automata-Based Approach Does Not Work Directly for LQMCs

In the remainder of this section, we present how the approach used to check Markov chains against ω-regular properties can be extended to quantum Markov chains.

The only difference between classical Markov chains and quantum Markov chains is the use of super-operators instead of probability values to decorate the transitions; this difference, however, has a considerable effect on the model checking algorithm, as pointed out in, e.g., [49,72]. Consider the two Markov chains, enriched with a parity acceptance condition, shown in Figure 2. The parity acceptance is provided by the priorities 0 and 1 labelling the classical states s_0, s_1, and s_2. As we will see more formally below, a path is accepted if the minimum priority occurring infinitely often is even. This means, for instance, that the path $s_0 s_1 s_2 s_1 s_2 \cdots$ is accepted, since its corresponding sequence of priorities is $00101 \cdots$ whose minimum priority occurring infinitely often is 0; on the other hand, the path $s_2 s_2 \cdots$ is rejecting, since its minimum priority occurring infinitely is 1.

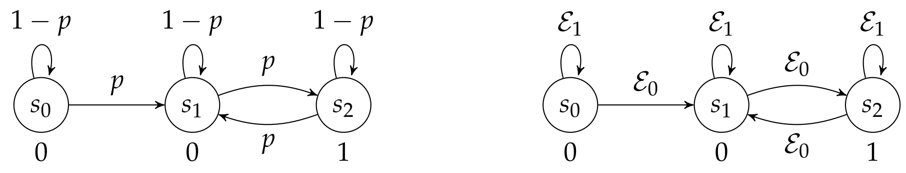

Figure 2. Example showing that BSCC decomposition for the underlying graph does not work for model checking parity QMCs.

The Markov chain on the left of Figure 2 is a classical one where transitions are governed by the two transition probabilities p and $1-p$ with $0 < p < 1$. The Markov chain on the right is a quantum one governed by the two super-operators $\mathcal{E}_0$ and $\mathcal{E}_1$, such that $\mathcal{E}_0, \mathcal{E}_1 \neq 0_{\mathcal{H}}$ and $\mathcal{E}_0 + \mathcal{E}_1 \eqsim \mathcal{I}_{\mathcal{H}}$. It is obvious that both Markov chains have the same classical state space, thus they have exactly the same underlying graph. If we would define the BSCCs just by looking at the underlying graph, they would have the same set of BSCCs, namely just the single BSCC $\{s_1, s_2\}$, and the same single transient state s_0. However, we will see that this BSCC technique does not help in the evaluation of parity QMCs.

In the classical Markov chains model, s_0 is a transient state that will eventually reach almost surely the unique BSCC $\{s_1, s_2\}$. Thus, its priority makes no difference. Since the probability that both states in the BSCC are visited infinitely often is 1, we have that the BSCC $\{s_1, s_2\}$ is marked as accepting, since the minimum priority of its states is 0. Since the BSCC $\{s_1, s_2\}$ is accepting and all paths from s_1 eventually enter $\{s_1, s_2\}$ almost surely, we find that the overall probability of the paths eventually becoming trapped in an accepting BSCC is 1 for all states.

Consider now the quantum Markov chain on the right part of Figure 2 and assume that $\mathcal{E}_0 = \{|0\rangle\langle 0|\}$ and $\mathcal{E}_1 = \{|1\rangle\langle 1|\}$. Note that for $i \in \{0,1\}$ it holds $\mathcal{E}_i \mathcal{E}_i = \mathcal{E}_i$ and $\mathcal{E}_i \mathcal{E}_{1-i} = 0_{\mathcal{H}}$. It is easy to check that if we start from s_0, the infinite path $s_0 s_0 \cdots$, with the corresponding *nonzero* super-operator $\lim_{n\to\infty} \mathcal{E}_1^n = \{|1\rangle\langle 1|\}$, never leaves s_0 to reach the other states s_1 and s_2. This means that s_0 cannot be considered a BSCC, since s_0 cannot reach itself almost surely, i.e., with super-operator $\mathcal{I}_{\mathcal{H}}$. At the same time, s_0 cannot be considered a transient state, since it will not eventually reach a BSCC almost surely. Therefore, the first problem we encounter is that a state in a QMC cannot be classified either as transient or as belonging to a BSCC, as happens for classical Markov chains. In addition to this problem with the classification of s_0, we also have a second problem, namely, that the infinite path $(s_0)^\omega$ has minimum priority 0, so it must be taken into account when computing the super-operator corresponding to the accepting paths. This means that it does not suffice to look only at the states as performed for classical parity Markov chains, but we need to consider all possible paths.

Consider now the state s_1: we have only two infinite paths occurring with nonzero super-operator, namely $(s_1)^\omega$ with associated super-operator $\{|1\rangle\langle 1|\}$ and priority 0, and $(s_1 s_2)^\omega$ with associated super-operator $\{|0\rangle\langle 0|\}$ and priority 0. Thus, the super-operator corresponding to the accepting paths in state s_1 is $\{|0\rangle\langle 0|\} + \{|1\rangle\langle 1|\} \eqsim \mathcal{I}_{\mathcal{H}}$. The situation for s_2 is however different: we still have only two paths, namely $(s_2 s_1)^\omega$ with associated super-operator $\{|0\rangle\langle 0|\}$ and priority 0, and $(s_2)^\omega$ having super-operator $\{|1\rangle\langle 1|\}$ and priority 1. Thus, the value in s_2 is $\{|0\rangle\langle 0|\}$, that is different from $\mathcal{I}_{\mathcal{H}}$ in s_1. Therefore, we have a third problem, namely, we cannot associate the same value to all states in a BSCC, as we can do for classical Markov chains, but each state must be evaluated individually.

Let us reason more about what happens along the path $s_2 s_2 s_1 s_1 s_1 \cdots$: its corresponding super-operator is $0_{\mathcal{H}}$, as effect of the composition of the two orthogonal super-operators $\mathcal{E}_1 = \{|1\rangle\langle 1|\}$ and $\mathcal{E}_0 = \{|0\rangle\langle 0|\}$ in the first two transitions $s_2 \xrightarrow{\mathcal{E}_1} s_2$ and $s_2 \xrightarrow{\mathcal{E}_0} s_1$, respectively. In practice, it is as the first transition from s_2 to s_2, decorated with $\mathcal{E}_1$, "disables" the second transition from s_2 to s_1, decorated with $\mathcal{E}_0$. In other words, it is like we are missing the Markovian property that allows us to forget about the past, namely that the transitions we can perform depend only on the current state, not on how we reached it. Therefore, we cannot assume that the underlying graph is immutable, since the previous transitions we performed to reach the current state might change the available edges in the underlying graph. This means that we cannot just use the standard BSCC decomposition algorithms on the underlying graph to solve the LTL and ω-regular model checking problem for LQMCs: neither are BSCCs reached with certainty, nor do all states of a BSCC have the same value. In addition, the value of a state in a BSCC can be different from both $0_{\mathcal{H}}$ and $\mathcal{I}_{\mathcal{H}}$.

In order to overcome these problems, in [49] we proposed an alternative approach: instead of relying on the graph structure of the product $\mathfrak{M} \otimes \mathfrak{A}$, we transformed it into a single super-operator and we considered the BSCC decomposition of the classical-quantum Hilbert space induced by this super-operator, i.e., the tensor product of the classical state space with the quantum one. By operating on this larger Hilbert space, we have the information we need to solve the model checking problem for ω-regular properties for labelled quantum Markov chains.

5.4. Parity Automata and Parity Quantum Markov Chains

We shortly recall a well-known mechanism to decide whether a word is included in a given ω-regular language. Recall that $\mathcal{W}$ is called ω-regular if it is the language of some nondeterministic Büchi automaton (cf. [27] (Theorem 4.32)) and that nondeterministic Büchi automata are equivalent to parity automata (cf. [63–65]). A parity automaton is defined formally as follows.

Definition 7. *A* (deterministic) parity automaton *(PA) is a tuple* $\mathfrak{A} = (A, \bar{a}, AP, t, \mathsf{pri})$, *where*

- A *is a finite set of* automaton states*;*
- $\bar{a} \in A$ *is the* initial state*;*
- AP *is a finite set of* atomic propositions*;*
- $t \colon A \times 2^{AP} \to A$ *is a* transition function*; and*
- $\mathsf{pri} \colon A \to \mathbb{N}$ *is a* priority function.

A *path* of $\mathfrak{A}$ is an infinite sequence $\sigma = a_0 L_0 a_1 L_1 \cdots \in (A \times 2^{AP})^{\omega}$, such that $a_0 = \bar{a}$ and for all $i \geq 0$, $a_{i+1} = t(a_i, L_i)$. We extend the priority function to paths by setting $\mathsf{pri}(\sigma) = \liminf_{i\to\infty} \mathsf{pri}(a_i)$. We use $\mathrm{Path}^{\mathfrak{A}}$ to denote the set of all paths of $\mathfrak{A}$. The *language* of $\mathfrak{A}$ is defined as

$$\mathcal{L}(\mathfrak{A}) = \{\, L_0 L_1 \cdots \in (2^{AP})^{\omega} \mid \exists \sigma = a_0 L_0 a_1 L_1 \cdots \in \mathrm{Path}^{\mathfrak{A}} : \mathsf{pri}(\sigma) \text{ is even} \,\}.$$

As we have enriched QMCs with labels, obtaining LQMCs (cf. Definition 2), we also decorate QMCs with parity conditions, so to obtain parity QMCs; these will be useful to represent the result of the product of an LQMC and a PA, that we will present later.

Definition 8. *A* parity quantum Markov chain *(PQMC) is a tuple* $\mathfrak{M} = (S, \mathbf{Q}, \mathsf{pri})$, *where* $(S, \mathbf{Q})$ *is a QMC and* $\mathsf{pri} \colon S \to \mathbb{N}$ *is a* priority function.

As we have done for LQMCs, we extend in the natural way the notion of paths, measures, etc. to PQMCs and we set $\mathsf{pri}(s_0 s_1 s_2 \cdots) = \liminf_{i\to\infty} \mathsf{pri}(s_i)$. We define the *value* of a PQMC $\mathfrak{M}$ in $s \in S$ as

$$\mathrm{val}_s^{\mathfrak{M}} = Q_s^{\mathfrak{M}}(\{\, \sigma \in \mathrm{Path}^{\mathfrak{M}} \mid \mathsf{pri}(\sigma) \text{ is even} \,\}).$$

Parity QMCs can be obtained by combining together an LQMC and a PA; in practice, the PA just follows the evolution of the LQMC by synchronizing on the labels of the classical states that are visited by the LQMC.

Definition 9. *The* product *of an LQMC $\mathfrak{M} = (S, \mathbf{Q}, AP, L)$ and a PA $\mathfrak{A} = (A, \bar{a}, AP, t, \mathsf{pri})$ with the same set of atomic propositions AP is the PQMC $\mathfrak{M} \otimes \mathfrak{A} = (S', \mathbf{Q}', \mathsf{pri}')$ where*

- *$S' = S \times A$;*
- *$\mathbf{Q}'((s, a), (s', a')) = \mathbf{Q}(s, s')$ if $a' = t(a, L(s))$, and $\mathbf{Q}'((s, a), (s', a')) = 0_{\mathcal{H}}$ otherwise; and*
- *$\mathsf{pri}'((s, a)) = \mathsf{pri}(a)$.*

An important property that is used in the model checking problem of an LQMC $\mathfrak{M} = (S, \mathbf{Q}, AP, L)$ against an ω-regular property $\mathcal{W}$ recognized by $\mathfrak{A} = (A, \bar{a}, AP, t, \mathsf{pri})$ is that the value of $\mathfrak{M} \otimes \mathfrak{A} = (S', \mathbf{Q}', \mathsf{pri}')$ is trace equivalent to the super-operator corresponding to $\mathcal{W}$ in $\mathfrak{M}$. Formally, by [49] (Lemma 9) we have that for any $s \in S$,

$$Q_s^{\mathfrak{M}}(\mathcal{L}(\mathfrak{A})) \eqsim \mathrm{val}_{(s,\bar{a})}^{\mathfrak{M}\otimes\mathfrak{A}}.$$

This property is the expected quantum counterpart of the product Markov chain used for solving the model checking problem of classical Markov chains against LTL and ω-regular properties (cf. [27]). This equivalence ensures that the super-operator we obtain in the product PQMC is equivalent to the one corresponding to the paths of the LQMC satisfying the given ω-regular property, and this is needed to justify the correctness of the automata-based approach to model checking.

Example 3. *As an example of the parity construction, consider again the QMC shown in Figure 1 and the LTL formula $(\neg\ell_2)\,\mathbf{U}\,\ell_3$.*

On the left part of Figure 3 we present the parity automaton $\mathfrak{A}$ corresponding to the formula $(\neg\ell_2)\,\mathbf{U}\,\ell_3$. The automaton has 3 states, namely $\bar{a}$, $\bot$, and $\top$; the value assigned by the priority function pri to each state is given as the number near the state: this means that all states have priority 1 except for $\mathsf{pri}(\top) = 0$. The transition relation is given by the depicted arrows, where we encode the sets of atomic propositions by a Boolean formula: for instance, the formula ℓ_3 is satisfied by all subsets A of $AP = \{\ell_0, \ell_1, \ell_2, \ell_3\}$, such that $\ell_3 \in A$; similarly, the formula $\ell_2 \wedge \neg\ell_3$ is satisfied by all subsets A of AP, such that $\ell_2 \in A$ but $\ell_3 \notin A$.

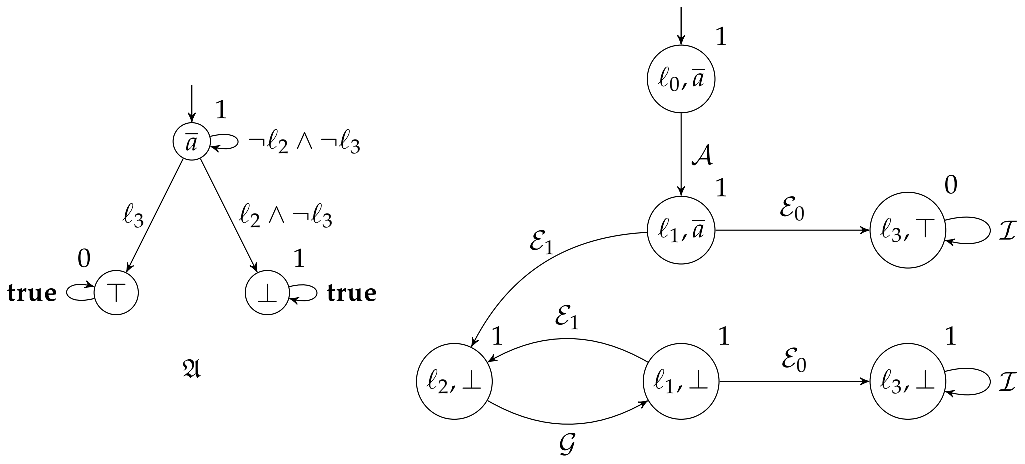

Figure 3. Parity construction for the QMC from Figure 1 and the LTL formula $(\neg\ell_2)\,\mathbf{U}\,\ell_3$.

5.5. Computing PQMC Values

As we have seen in Section 5.3, it is not enough to compute the BSCCs by looking only at the classical states of the product PQMC. The solution adopted by [49] is to encode the behavior of $\mathfrak{M}$ into a single super-operator acting on the *extended* Hilbert space $\mathcal{H}_c \otimes \mathcal{H}$ representing both the classical states and the super-operators decorating the transitions. This is obtained by taking the tensor product of the classical state space and the quantum one; then the notion of BSCC *subspaces* for super-operators, as defined in [73], is used in the computation of the PQMC values. The intuition underlying this extended construction is to keep track of the classical states reached step by step in the $\mathcal{H}_c$ part of $\mathcal{H}_c \otimes \mathcal{H}$, and to accumulate the effects of the super-operators decorating the taken transitions on the $\mathcal{H}$ part of $\mathcal{H}_c \otimes \mathcal{H}$. This allows us to overcome the problem of the BSCC decomposition of the underlying graph of the given PQMC: as we have seen in Section 5.3, by performing a transition we might change the underlying graph, since some transition might be disabled as effect of the composition of super-operators; this does not happen in the classical and probabilistic setting, where the underlying graph is immutable.

Let $\mathcal{E}$ be a super-operator acting on $\mathcal{H}$; given a subspace X of $\mathcal{H}$, we call X:

- *invariant* for $\mathcal{E}$ if $\mathcal{E}(X) \subseteq X$;
- a *BSCC* of $\mathcal{E}$ if $\mathcal{R}(|\psi\rangle\langle\psi|) = X$ holds for each pure state $|\psi\rangle \in X$, where $\mathcal{R}(\rho) = \bigvee_{i=0}^{\infty} \operatorname{supp}(\mathcal{E}^i(\rho))$ is the *reachable subspace* of $\mathcal{E}$ starting in $\rho \in \mathcal{D}(\mathcal{H})$;
- *transient* if $\lim_{k\to\infty} \operatorname{tr}(P_X \mathcal{E}^k(\rho)) = 0$ for each $\rho \in \mathcal{D}(\mathcal{H})$, where P_X is the projection onto X.

Intuitively, we have that the probability of remaining forever in a transient subspace X tends to 0, as happens for the transient states in a classical Markov chain. Similarly, it is not possible to leave a BSCC X, and eventually all elements of X are visited again and again, as happens for states in the BSCCs of a classical Markov chain.

We are now ready to present how to combine together the classical part and the quantum part of a PQMC. Let $\mathfrak{M} = (S, \mathbf{Q}, \mathsf{pri})$ be a PQMC on $\mathcal{H}$; for each $s, t \in S$, let $\mathcal{E}^{s,t} = \{ E_j^{s,t} \mid j \in J^{s,t} \}$ be the set of Kraus operators with $J^{s,t}$ as set of indexes for the super-operator $\mathcal{E}^{s,t} = \mathbf{Q}(s,t)$. Following [72], let $\mathcal{E}_{\mathfrak{M}} = \{ |t\rangle\langle s| \otimes E_j^{s,t} \mid s,t \in S, j \in J^{s,t} \}$ be the super-operator acting on the extended Hilbert space $\mathcal{H}_c \otimes \mathcal{H}$, where $\mathcal{H}_c$ is the $|S|$-dimensional Hilbert space with orthonormal basis $\{ |s\rangle \mid s \in S \}$.

For each $s \in S$, let $O^s = \{ (t,j) \mid t \in S, j \in J^{s,t} \}$ and $M^s = \{ E_j^{s,t} \mid (t,j) \in O^s \}$. To see how $\mathcal{E}_{\mathfrak{M}}$ encodes the behavior of $\mathfrak{M}$, consider the following steps, which exactly capture the intended meaning of $\mathfrak{M}$.

1. A projective measurement $M = \{ |s\rangle\langle s| \mid s \in S \}$ is performed on the classical system $\mathcal{H}_c$ to determine the current classical state.
2. If the measurement outcome of M is s, then the quantum measurement M^s is performed on the quantum system $\mathcal{H}$.
3. If the observed outcome of M^s is (t,i), then the classical state is set to be $|t\rangle\langle t|$.

The super-operator $\mathcal{E}_{\mathfrak{M}}$ enjoys a couple of nice properties: starting from a product initial state, the classical and quantum systems remain separable (disentangled) along the whole execution (cf. [49] (Lemma 11)); moreover, each fixed point state σ of $\mathcal{E}_{\mathfrak{M}}$, i.e., $\mathcal{E}_{\mathfrak{M}}(\sigma) = \sigma$, has the form $\sigma = \sum_{s\in S} |s\rangle\langle s| \otimes \sigma_s$. According to [73], we find that any BSCC of $\mathcal{E}_{\mathfrak{M}}$ can be spanned by pure states of the form $|s\rangle|\eta\rangle$ where $s \in S$ and $|\eta\rangle \in \mathcal{H}$.

We are now ready to collect all BSCCs of $\mathcal{E}_{\mathfrak{M}}$ and classify them according to their acceptance condition. Given a BSCC B of $\mathcal{E}_{\mathfrak{M}}$, let

$$C(B) = \{ s \in S \mid |s\rangle|\eta\rangle \in B \text{ for some } |\eta\rangle \in \mathcal{H} \}$$

be the set of classical states supported in B. Note that it is possible to have $C(B) \cap C(B') \neq \emptyset$ for two different BSCCs B and B'; this happens when there are two states $|s\rangle|\eta\rangle \in B$ and $|s'\rangle|\eta'\rangle \in B'$ with $s = s'$. However, this is not a real problem: as shown by [49] (Lemma 12), two BSCCs are orthogonal as long as they differ on at least one classical state.

Given the set of all BSCCs of $\mathcal{E}_{\mathfrak{M}}$, we can organize it with respect to the minimal priority assigned to the classical states in each BSCC. For a priority $k \in \mathbb{N}$, let $\mathcal{BSCC}_{=k}$ be the span of all BSCCs of $\mathcal{E}_{\mathfrak{M}}$ whose minimal priority is k, that is,

$$\mathcal{BSCC}_{=k} = \bigvee \{ B \text{ is a BSCC of } \mathcal{E}_{\mathfrak{M}} \mid \min\{ \mathsf{pri}(s) \mid s \in C(B) \} = k \}.$$

Similarly, let $\mathcal{BSCC}_{<k}$ and $\mathcal{BSCC}_{>k}$ be the spans of all BSCCs with the minimal priority being less than and larger than k, respectively. Since $\mathcal{BSCC}_{=k}$, $\mathcal{BSCC}_{<k}$, and $\mathcal{BSCC}_{>k}$ have different sets of classical states, they are pairwise orthogonal by [49] (Lemma 12). This implies, by [73], that $\mathcal{H}_c \otimes \mathcal{H}$ can be decomposed *uniquely* into

$$\mathcal{H}_c \otimes \mathcal{H} = T \oplus \mathcal{BSCC}_{=k} \oplus \mathcal{BSCC}_{<k} \oplus \mathcal{BSCC}_{>k},$$

where T is the maximum transient subspace of $\mathcal{E}_{\mathfrak{M}}$.

An important result (cf. [49] (Lemma 13)) is that the BSCCs in $\mathcal{BSCC}_{=k}$, $\mathcal{BSCC}_{<k}$, and $\mathcal{BSCC}_{>k}$ do not interfere with each other: let $\mathrm{tr}(R_t^k(\rho))$ denote the probability that k is the lowest priority occurring infinitely often from the initial state $|t\rangle\langle t| \otimes \rho$. Then, we have that:

- if $\mathrm{supp}(|t\rangle\langle t| \otimes \rho) \subseteq \mathcal{BSCC}_{=k}$, then $\mathrm{tr}(R_t^k(\rho)) = \mathrm{tr}(\rho)$;
- if $\mathrm{supp}(|t\rangle\langle t| \otimes \rho) \subseteq \mathcal{BSCC}_{<k}$, then $\mathrm{tr}(R_t^k(\rho)) = 0$; and
- if $\mathrm{supp}(|t\rangle\langle t| \otimes \rho) \subseteq \mathcal{BSCC}_{>k}$, then $\mathrm{tr}(R_t^k(\rho)) = 0$,

that is, once we enter $\mathcal{BSCC}_{=k}$ with probability $\mathrm{tr}(\rho)$, we will see k as the minimum priority almost surely; instead, if we enter $\mathcal{BSCC}_{<k}$ or $\mathcal{BSCC}_{>k}$, we will see k as minimum priority with probability 0. In practice, $\mathcal{BSCC}_{=k}$ acts like the standard BSCC with minimum priority k in classical Markov chains.

It now remains to collect all $\mathcal{BSCC}_{=k}$ with even priority k to obtain the accepting BSCCs, and combine together their super-operators. This is formalized as follows.

Theorem 1 ([49] (Corollary 15))**.** *Let $\mathfrak{M} = (S, \mathbf{Q}, \mathsf{pri})$ be a PQMC. Then, for any $s \in S$,*

$$\mathrm{val}_s^{\mathfrak{M}} \eqsim \mathrm{tr}_c \circ \mathcal{P}_{even} \circ \mathcal{E}_{\mathfrak{M}}^{\infty} \circ \mathcal{E}_s$$

where tr_c is the partial trace super-operator, such that $\mathrm{tr}_c(|s\rangle\langle t| \otimes \rho) = \langle t\rangle s \cdot \rho$, $\mathcal{P}_{even} = \sum_{\{k \in \mathsf{pri}(S) | k \text{ is even}\}} \mathcal{P}_{=k}$ where $\mathcal{P}_{=k} = \{P_{=k}\}$ is the projection super-operator onto $\mathcal{BSCC}_{=k}$, $\mathcal{E}_{\mathfrak{M}}^{\infty} = \lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \mathcal{E}_{\mathfrak{M}}^{n}$, and $\mathcal{E}_s(\rho) = |s\rangle\langle s| \otimes \rho$.

Similarly to the QCTL case (cf. Section 3), the value $\mathrm{val}_s^{\mathfrak{M}}$ is not computed exactly, but only a *trace equivalent* super-operator is obtained. This is in line with the statement of [44] (Theorem 3.2), saying that the super-operator valued measure for a QMC (hence, for a PQMC) is well-defined only up to the trace equivalence $\eqsim$, already at the level of the measure for the measurable space $(\mathrm{Path}^{\mathfrak{M}}, \Sigma)$. This means that we are not able to compute the *exact* super-operator corresponding to a measurable set of paths, but only a super-operator in the same $\eqsim$-equivalence class. This suffices for our purposes, since LTL and ω-regular model checking for LQMCs (as well as the QCTL model checking) only look for the *probability* of satisfying the given property, which is preserved by $\eqsim$.

As an application of the above construction, consider the right part of Figure 3, where we show the product of the QMC shown in Figure 1 and the PA $\mathfrak{A}$ encoding the formula $(\neg\ell_2)\,\mathbf{U}\,\ell_3$. It is easy to observe that, for this specific case, we have only one accepting BSCC, composed only by the state $(\ell_3, \top)$, and that the super-operator corresponding to the paths leading to it is $\mathcal{E}_0\mathcal{A}$, the same as we computed in Example 1, as expected.

As another application of the above construction, consider again the PQMC depicted on the right of Figure 2 where the super-operators decorating the transitions are $\mathcal{E}_0 = \{|0\rangle\langle 0|\}$ and $\mathcal{E}_1 = \{|1\rangle\langle 1|\}$. Then, the super-operator $\mathcal{E}_{\mathfrak{M}}$ encoding $\mathfrak{M}$ is

$$\begin{aligned}\mathcal{E}_{\mathfrak{M}} = \quad & \{|s_1\rangle\langle s_0|\} \otimes \mathcal{E}_0 + \{|s_0\rangle\langle s_0|\} \otimes \mathcal{E}_1 \\ & + \{|s_2\rangle\langle s_1|\} \otimes \mathcal{E}_0 + \{|s_1\rangle\langle s_1|\} \otimes \mathcal{E}_1 \\ & + \{|s_1\rangle\langle s_2|\} \otimes \mathcal{E}_0 + \{|s_2\rangle\langle s_2|\} \otimes \mathcal{E}_1.\end{aligned}$$

In practice, for each pair of states s and t for which $\mathbf{Q}(s,t) \not\eqsim 0_{\mathcal{H}}$, we add $\{|t\rangle\langle s|\} \otimes \mathbf{Q}(s,t)$, which can be intuitively interpreted as: we transition from s to t while performing $\mathbf{Q}(s,t)$. We can now decompose $\mathcal{E}_{\mathfrak{M}}$ into the BSCCs and the maximal transient space, obtaining that the maximal transient space of $\mathcal{E}_{\mathfrak{M}}$ is $T = \mathrm{span}\{|s_0\rangle|0\rangle\}$ and that the BSCCs are

$$B_1 = \operatorname{span}\{|s_0\rangle\,|1\rangle\}, \qquad B_2 = \operatorname{span}\{|s_1\rangle\,|1\rangle\},$$
$$B_3 = \operatorname{span}\{|s_1\rangle\,|0\rangle, |s_2\rangle\,|0\rangle\}, \qquad B_4 = \operatorname{span}\{|s_2\rangle\,|1\rangle\}.$$

Since the priorities range over $\{0,1\}$, we obtain only $\mathcal{BSCC}_{=0} = \bigvee\{B_1, B_2, B_3\}$ with even minimum priority, with corresponding projection super-operator (in Kraus notation)

$$\mathcal{P}_{=0} = \{|s_0\rangle\langle s_0| \otimes |1\rangle\langle 1| + |s_1\rangle\langle s_1| \otimes I + |s_2\rangle\langle s_2| \otimes |0\rangle\langle 0|\},$$

which is also the projection super-operator $\mathcal{P}_{even}$ occurring in Theorem 1.

Regarding the super-operator $\mathcal{E}^{\infty}_{\mathfrak{M}}$, we first compute its components. For any $n \geq 1$, we have $\mathcal{E}^{2n-1}_{\mathfrak{M}} \approx \mathcal{F}_0 \otimes \mathcal{E}_0 + \mathcal{F} \otimes \mathcal{E}_1$ and $\mathcal{E}^{2n}_{\mathfrak{M}} \approx \mathcal{F}_1 \otimes \mathcal{E}_0 + \mathcal{F} \otimes \mathcal{E}_1$, where

$$\begin{aligned}
\mathcal{F}_0 &= \{|s_1\rangle\langle s_0|, |s_2\rangle\langle s_1|, |s_1\rangle\langle s_2|\} \\
\mathcal{F}_1 &= \{|s_2\rangle\langle s_0|, |s_1\rangle\langle s_1|, |s_2\rangle\langle s_2|\} \\
\mathcal{F} &= \{|s_0\rangle\langle s_0|, |s_1\rangle\langle s_1|, |s_2\rangle\langle s_2|\}.
\end{aligned}$$

Thus, $\mathcal{E}^{\infty}_{\mathfrak{M}} \approx \frac{\mathcal{F}_0+\mathcal{F}_1}{2} \otimes \mathcal{E}_0 + \mathcal{F} \otimes \mathcal{E}_1$, and

$$\mathcal{P}_0 \circ \mathcal{E}^{\infty}_{\mathfrak{M}} \approx \frac{\mathcal{F}_0 + \mathcal{F}_1}{2} \otimes \mathcal{E}_0 + (\mathcal{P}_{s_0} + \mathcal{P}_{s_1}) \otimes \mathcal{E}_1.$$

Lastly, we have $\mathcal{E}_s = \{|s\rangle \otimes I_{\mathcal{H}}\}$ and $\mathrm{tr}_c = \{\,\langle s_i| \otimes I_{\mathcal{H}} \mid i = 0,1,2\,\}$. By Theorem 1, it follows that

$$\mathrm{val}^{\mathfrak{M}}_s \approx \mathrm{tr}_c \circ \mathcal{P}_0 \circ \mathcal{E}^{\infty}_{\mathfrak{M}} \circ \mathcal{E}_s = \begin{cases} \mathcal{E}_0 + \mathcal{E}_1 \approx \mathcal{I}_{\mathcal{H}} & \text{if } s = s_0 \text{ or } s = s_1; \\ \mathcal{E}_0 & \text{if } s = s_2. \end{cases}$$

These values coincide with the informal discussion we have given in Section 5.3.

5.6. Complexity of the LTL and ω-Regular Model Checking Problem

A detailed algorithm providing the necessary computational steps implementing the construction given above can be found in [49]. Its complexity for computing the matrix representation of $\mathrm{tr}_c \circ \mathcal{P}_{even} \circ \mathcal{E}^{\infty}_{\mathfrak{M}} \circ \mathcal{E}_s$ is polynomial in the size of the given PQMC and in the dimension of its Hilbert space.

If we want to express the size of the product PQMC $\mathfrak{M} \otimes \mathfrak{A}_{\varphi}$ in terms of the original LQMC $\mathfrak{M}$ and of the LTL formula φ, then $\mathfrak{M} \otimes \mathfrak{A}_{\varphi}$ is linear in the size of $\mathfrak{M}$ and double exponential in the size of φ. The double exponential blowup is common in model checking LTL properties: the first exponential blowup occurs in the translation from φ to an equivalent nondeterministic Büchi automaton; the second blowup is caused by its determinization as a Rabin, a Streett, a Muller, or a parity automaton. These blowups cannot be avoided (cf. [26,27,63]).

6. Conclusions

In this paper we have presented three logics and relative model checking algorithms for quantum Markov chains given in literature: quantum CTL, quantum LTL and ω-regular properties, and fidelity CTL. The former two kinds of properties focus on evaluating the probability of certain events; the latter, instead, looks for how well the super-operator modelled by the LQMC preserves the quantum states it is applied on.

The algorithms for the quantum counterparts of CTL and LTL follow the steps of the algorithms already developed for these logics in the setting of classical and probabilistic model checking. The quantum nature of the system to be checked, however, prevents the naïve adaptation of these algorithms, in particular for LTL and ω-regular properties: the usual BSCC decomposition of the underlying graph fails, so a similar decomposition needs to be performed in an extended Hilbert space representing both the classical and the quantum states of the product PQMC. The algorithm for the computation of the states satisfying the fidelity operator, instead, is newly designed, since fidelity is a concept strictly related to quantum systems that has no counterparts in classical and probabilistic systems.

Funding: This work was supported in part by the Guangdong Science and Technology Department (Grant No. 2018B010107004) and by the National Natural Science Foundation of China (Grants Nos. 61836005, 62102407, 62172019).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Dijkstra, E.W. The Humble Programmer. *Commun. ACM* **1972**, *15*, 859–866. [CrossRef]
2. Clarke, E.M.; Emerson, E.A. Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic. In *Logic of Programs*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1981; Volume 131, pp. 52–71.
3. Queille, J.; Sifakis, J. Specification and verification of concurrent systems in CESAR. In *Symposium on Programming*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1982; Volume 137, pp. 337–351.
4. Burch, J.R.; Clarke, E.M.; McMillan, K.L.; Dill, D.L. Sequential Circuit Verification Using Symbolic Model Checking. In Proceedings of the 27th ACM/IEEE Design Automation Conference, Orlando, FL, USA, 24–28 June 1990; pp. 46–51.
5. Clarke, E.M.; Filkorn, T.; Jha, S. Exploiting Symmetry In Temporal Logic Model Checking. In *CAV*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1993; Volume 697, pp. 450–462.
6. Gupta, A. Formal Hardware Verification Methods: A Survey. *Formal Methods Syst. Des.* **1992**, *1*, 151–238. [CrossRef]
7. Kropf, T. *Introduction to Formal Hardware Verification*; Springer: Berlin/Heidelberg, Germany, 1999.
8. Yoeli, M. *Formal Verification of Hardware Design*; IEEE Computer Society Press: Piscataway, NJ, USA, 1990.
9. Liggesmeyer, P.; Rothfelder, M.; Rettelbach, M.; Ackermann, T. Qualitätssicherung Software-basierter technischer Systeme—Problembereiche und Lösungsansätze. *Inform. Spektrum* **1998**, *21*, 249–258. [CrossRef]
10. Peled, D.A. *Software Reliability Methods*; Texts in Computer Science; Springer: Berlin/Heidelberg, Germany, 2001.
11. Rushby, J.M.; von Henke, F.W. Formal Verification of Algorithms for Critical Systems. *IEEE Trans. Softw. Eng.* **1993**, *19*, 13–23. [CrossRef]
12. Tretmans, J.; Wijbrans, K.; Chaudron, M.R.V. Software Engineering with Formal Methods: The Development of a Storm Surge Barrier Control System Revisiting Seven Myths of Formal Methods. *Formal Methods Syst. Des.* **2001**, *19*, 195–215. [CrossRef]
13. Wijbrans, K.; Buve, F.; Rijkers, R.; Geurts, W. Software Engineering with Formal Methods: Experiences with the Development of a Storm Surge Barrier Control System. In *FM*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5014, pp. 419–424.
14. Holzmann, G.J. *Design and Validation of Computer Protocols*; Prentice-Hall, Inc.: Hoboken, NJ, USA, 1990.
15. Holzmann, G.J. Design and Validation of Protocols: A Tutorial. *Comput. Netw. ISDN Syst.* **1993**, *25*, 981–1017. [CrossRef]
16. Holzmann, G.J. The Theory and Practice of A Formal Method: NewCoRe. In *IFIP Congress (1)*; IFIP Transactions; North-Holland: Amsterdam, The Netherlands, 1994; Volume A-51, pp. 35–44.
17. Clarke, E.M.; Grumberg, O.; Hiraishi, H.; Jha, S.; Long, D.E.; McMillan, K.L.; Ness, L.A. Verification of the Futurebus+ Cache Coherence Protocol. *Formal Methods Syst. Des.* **1995**, *6*, 217–232. [CrossRef]
18. Lowe, G. Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR. *Softw. Concepts Tools* **1996**, *17*, 93–102.
19. Chen, T.; Diciolla, M.; Kwiatkowska, M.Z.; Mereacre, A. Quantitative verification of implantable cardiac pacemakers over hybrid heart models. *Inf. Comput.* **2014**, *236*, 87–101. [CrossRef]
20. Chan, W.; Anderson, R.J.; Beame, P.; Burns, S.; Modugno, F.; Notkin, D.; Reese, J.D. Model Checking Large Software Specifications. *IEEE Trans. Softw. Eng.* **1998**, *24*, 498–520. [CrossRef]
21. Staunstrup, J.; Andersen, H.R.; Hulgaard, H.; Lind-Nielsen, J.; Larsen, K.G.; Behrmann, G.; Kristoffersen, K.J.; Skou, A.; Leerberg, H.; Theilgaard, N.B. Practical Verification of Embedded Software. *Computer* **2000**, *33*, 68–75. [CrossRef]
22. Havelund, K.; Lowry, M.R.; Penix, J. Formal Analysis of a Space-Craft Controller Using SPIN. *IEEE Trans. Softw. Eng.* **2001**, *27*, 749–765. [CrossRef]
23. Holzmann, G.J.; Najm, E.; Serhrouchni, A. SPIN Model Checking: An Introduction. *Int. J. Softw. Tools Technol. Transf.* **2000**, *2*, 321–327. [CrossRef]
24. Bozzano, M.; Cimatti, A.; Katoen, J.; Katsaros, P.; Mokos, K.; Nguyen, V.Y.; Noll, T.; Postma, B.; Roveri, M. Spacecraft early design validation using formal methods. *Reliab. Eng. Syst. Saf.* **2014**, *132*, 20–35. [CrossRef]
25. Hoque, K.A.; Mohamed, O.A.; Savaria, Y. Towards an accurate reliability, availability and maintainability analysis approach for satellite systems based on probabilistic model checking. In Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 9–13 March 2015; pp. 1635–1640.
26. Clarke, E.M.; Henzinger, T.A.; Veith, H.; Bloem, R. (Eds.) *Handbook of Model Checking*; Springer: Berlin/Heidelberg, Germany, 2018.

27. Baier, C.; Katoen, J. *Principles of Model Checking*; MIT Press: Cambridge, MA, USA, 2008.
28. Bérard, B.; Bidoit, M.; Finkel, A.; Laroussinie, F.; Petit, A.; Petrucci, L.; Schnoebelen, P.; McKenzie, P. *Systems and Software Verification, Model-Checking Techniques and Tools*; Springer: Berlin/Heidelberg, Germany, 2001.
29. Clarke, E.M.; Grumberg, O.; Peled, D.A. *Model Checking*; MIT Press: Cambridge, MA, USA, 2001.
30. Huth, M.; Ryan, M.D. *Logic in Computer Science—Modelling and Reasoning about Systems*; Cambridge University Press: Cambridge, UK, 2000.
31. Schneider, K. *Verification of Reactive Systems—Formal Methods and Algorithms*; Texts in Theoretical Computer Science; An EATCS Series; Springer: Berlin/Heidelberg, Germany, 2004.
32. Ben-Ari, M. *Principles of the SPIN Model Checker*; Springer: Berlin/Heidelberg, Germany, 2008.
33. McMillan, K.L. *Symbolic Model Checking*; Kluwer: Alphen aan den Rijn, The Netherlands, 1993.
34. Bennett, C.H.; Brassard, G. Quantum cryptography: Public-key distribution and coin tossing. In Proceedings of the IEEE International Conference on Computer, Systems and Signal Processing, Bangalore, India, 9–12 December 1984; pp. 175–179.
35. Bennett, C.H.; Brassard, G. Quantum cryptography: Public key distribution and coin tossing. *Theor. Comput. Sci.* **2014**, *560*, 7–11. [CrossRef]
36. Bennett, C.H. Quantum cryptography using any two nonorthogonal states. *Phys. Rev. Lett.* **1992**, *68*, 3121. [CrossRef]
37. Bennett, C.H.; Wiesner, S.J. Communication Via One- and Two-particle Operators on Einstein-Podolsky-Rosen States. *Phys. Rev. Lett.* **1992**, *69*, 2881–2884. [CrossRef] [PubMed]
38. Shor, P.W. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In Proceedings of the 35th Annual Symposium on Foundations of Computer Science, Santa Fe, NM, USA, 20–22 November 1994; pp. 124–134.
39. Gay, S.J.; Nagarajan, R.; Papanikolaou, N. Probabilistic model-checking of quantum protocols. In Proceedings of the 2nd International Workshop on Developments in Computational Models, Venice, Italy, 16 July 2006.
40. Papanikolaou, N. Model Checking Quantum Protocols. Ph.D. Thesis, University of Warwick, Coventry, UK, 2009.
41. Gay, S.J.; Nagarajan, R.; Papanikolaou, N. QMC: A Model Checker for Quantum Systems. In *CAV*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5123, pp. 543–547.
42. Hung, W.N.N.; Song, X.; Yang, G.; Yang, J.; Perkowski, M.A. Quantum logic synthesis by symbolic reachability analysis. In Proceedings of the 41st Annual Design Automation Conference, San Diego, CA, USA, 7–11 June 2004; pp. 838–841.
43. Hung, W.N.N.; Song, X.; Yang, G.; Yang, J.; Perkowski, M.A. Optimal synthesis of multiple output Boolean functions using a set of quantum gates by symbolic reachability analysis. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2006**, *25*, 1652–1663. [CrossRef]
44. Feng, Y.; Yu, N.; Ying, M. Model checking quantum Markov chains. *J. Comput. Syst. Sci.* **2013**, *79*, 1181–1198. [CrossRef]
45. Pnueli, A. The Temporal Logic of Programs. In Proceedings of the 18th Annual Symposium on Foundations of Computer Science, Providence, RI, USA, 31 October–2 November 1977; pp. 46–57.
46. Emerson, E.A.; Halpern, J.Y. "Sometimes" and "Not Never" revisited: On branching versus linear time temporal logic. *J. ACM* **1986**, *33*, 151–178. [CrossRef]
47. Hansson, H.; Jonsson, B. A Logic for Reasoning about Time and Reliability. *Formal Aspects Comput.* **1994**, *6*, 512–535. [CrossRef]
48. Bianco, A.; de Alfaro, L. Model Checking of Probabalistic and Nondeterministic Systems. In *FSTTCS*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1995; Volume 1026, pp. 499–513.
49. Feng, Y.; Hahn, E.M.; Turrini, A.; Ying, S. Model Checking Omega-regular Properties for Quantum Markov Chains. In *CONCUR*; Schloss Dagstuhl—Leibniz-Zentrum für Informatik: Wadern, Germany, 2017; Volume 85, pp. 35:1–35:16.
50. Xu, M.; Fu, J.; Mei, J.; Deng, Y. An Algebraic Method to Fidelity-based Model Checking over Quantum Markov Chains. *arXiv* **2021**, arXiv:2101.04971.
51. Duan, Z.; Niu, L. Some properties of quantum fidelity in infinite-dimensional quantum systems. *Int. J. Quantum Inf.* **2018**, *16*, 1850028. [CrossRef]
52. Uhlmann, A. On "Partial" Fidelities. *Rep. Math. Phys.* **2000**, *45*, 407–418. [CrossRef]
53. Burrell, A.H.; Szwer, D.J.; Webster, S.C.; Lucas, D.M. Scalable simultaneous multiqubit readout with 99.99% single-shot fidelity. *Phys. Rev. A* **2010**, *81*, 040302. [CrossRef]
54. Myerson, A.H.; Szwer, D.J.; Webster, S.C.; Allcock, D.T.C.; Curtis, M.J.; Imreh, G.; Sherman, J.A.; Stacey, D.N.; Steane, A.M.; Lucas, D.M. High-Fidelity Readout of Trapped-Ion Qubits. *Phys. Rev. Lett.* **2008**, *100*, 200502. [CrossRef]
55. Ying, M.; Feng, Y. *Model Checking Quantum Systems: Principles and Algorithms*; Cambridge University Press: Cambridge, UK, 2021.
56. Nielsen, M.A.; Chuang, I.L. *Quantum Computation and Quantum Information*; Cambridge University Press: Cambridge, UK, 2000.
57. Selinger, P. A Brief Survey of Quantum Programming Languages. In *FLOPS*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2004; Volume 2998, pp. 1–6.
58. Feng, Y.; Hahn, E.M.; Turrini, A.; Zhang, L. QPMC: A Model Checker for Quantum Programs and Protocols. In *FM*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2015; Volume 9109, pp. 265–272.
59. de Alfaro, L. Formal Verification of Probabilistic Systems. Ph.D. Thesis, Stanford University, Stanford, CA, USA, 1997.
60. Courcoubetis, C.; Yannakakis, M. The Complexity of Probabilistic Verification. *J. ACM* **1995**, *42*, 857–907. [CrossRef]
61. Bustan, D.; Rubin, S.; Vardi, M.Y. Verifying omega-Regular Properties of Markov Chains. In CAV; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2004; Volume 3114, pp. 189–201.

62. Vardi, M.Y.; Wolper, P. An Automata-Theoretic Approach to Automatic Program Verification (Preliminary Report). In Proceedings of the First Symposium on Logic in Computer Science, Cambridge, MA, USA, 16–18 June 1986; pp. 332–344.
63. Grädel, E.; Thomas, W.; Wilke, T. (Eds.) *Automata, Logics, and Infinite Games: A Guide to Current Research*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2002; Volume 2500.
64. Mostowski, A.W. Regular expressions for infinite trees and a standard form of automata. In *Symposium on Computation Theory*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1984; Volume 208, pp. 157–168.
65. Emerson, E.A.; Jutla, C.S. Tree Automata, Mu-Calculus and Determinacy (Extended Abstract). In Proceedings of the 32nd Annual Symposium of Foundations of Computer Science, San Juan, PR, USA, 1–4 October 1991; pp. 368–377.
66. Safra, S. On the Complexity of omega-Automata. In Proceedings of the 29th Annual Symposium on Foundations of Computer Science, White Plains, NY, USA, 24–26 October 1988; pp. 319–327.
67. Piterman, N. From Nondeterministic Büchi and Streett Automata to Deterministic Parity Automata. *Log. Methods Comput. Sci.* **2007**, *3*, 1–21. [CrossRef]
68. Schewe, S. Tighter Bounds for the Determinisation of Büchi Automata. In *FoSSaCS*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2009; Volume 5504, pp. 167–181.
69. Kähler, D.; Wilke, T. Complementation, Disambiguation, and Determinization of Büchi Automata Unified. In *ICALP (1)*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2008; Volume 5125, pp. 724–735.
70. Fogarty, S.; Kupferman, O.; Vardi, M.Y.; Wilke, T. Profile trees for Büchi word automata, with application to determinization. *Inf. Comput.* **2015**, *245*, 136–151. [CrossRef]
71. Löding, C.; Pirogov, A. Determinization of Büchi Automata: Unifying the Approaches of Safra and Muller-Schupp. In *ICALP*; LIPIcs; Schloss Dagstuhl—Leibniz-Zentrum für Informatik: Wadern, Germany, 2019; Volume 132, pp. 120:1–120:13.
72. Li, L.; Feng, Y. Quantum Markov chains: Description of hybrid systems, decidability of equivalence, and model checking linear-time properties. *Inf. Comput.* **2015**, *244*, 229–244. [CrossRef]
73. Ying, S.; Feng, Y.; Yu, N.; Ying, M. Reachability Probabilities of Quantum Markov Chains. In *CONCUR*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2013; Volume 8052, pp. 334–348.