QPMC: A Model Checker for Quantum Programs and Protocols

Yuan Feng[1], Ernst Moritz Hahn[2], Andrea Turrini[2], and Lijun Zhang[2]

[1] Centre for Quantum Computation and Intelligent Systems,
University of Technology Sydney, Australia
[2] State Key Laboratory of Computer Science, Institute of Software,
Chinese Academy of Sciences

Abstract. We present QPMC *(Quantum Program/Protocol Model Checker)*, an extension of the probabilistic model checker ISCASMC to automatically verify quantum programs and quantum protocols. QPMC distinguishes itself from the previous quantum model checkers proposed in the literature in that it works for general quantum programs and protocols, not only those using Clifford operations. A command-line version of QPMC is available at `http://iscasmc.ios.ac.cn/tool/`. Details of the case studies can be found in Appendix B.

1 Introduction and Motivation

Although commercial quantum computers are still in their infancy, rapid progress has been made in building reliable and scalable components for quantum computers. In particular, quantum cryptographic systems are already commercially available by companies such as Id Quantique, Cerberis, MagiQ Technologies, SmartQuantum, and NEC. One of the greatest advantages of quantum cryptography over its classical counterpart is that the security and the ability to detect the presence of eavesdropping are provable, based on the principles of quantum mechanics. In practice, however, as quantum mechanics is counter-intuitive, quantum cryptographic protocol designers will inevitably make more mistakes than classical protocol designers, especially when more and more complicated quantum protocols can be implemented by future physical technology. Therefore, it is indispensable to develop methodologies and techniques for the verification of quantum systems.

In the last decade, researchers started to explore the possibility of applying model checking, one of the dominant techniques for verification which has a large number of successful industrial applications, to the verification of quantum programs as well as quantum protocols. The main obstacle is that the set of all quantum states, traditionally regarded as the underlying state space of the model to be checked, is a continuum. Hence, the techniques of classical model checking, which normally work only for a finite state space, cannot be applied directly. Gay et al. [11] provided a solution to this problem by restricting the state space to a set of finitely describable states called *stabiliser states*, and restricting the quantum operations applied on them to the class of Clifford group. By doing this, they were able to obtain an efficient model checker [12] for quantum protocols, employing purely classical algorithms.

There is one limitation of the approach by Gay et al.: since only quantum protocols expressible in stabiliser formalism are considered, so that the state space can be encoded in a classical way, their model checker does not work for general protocols. To deal with this problem, one of the authors of the current paper and his colleagues proposed a novel notion of super-operator weighted Markov chain in which the state space is taken classical (and usually can be finite), while all quantum effects are encoded in the super-operators labelling the transitions [10]. This model is especially suited for verification of *classical properties* for which only the measurement outcomes as well as the probabilities of obtaining them are relevant, and the quantum effects caused by superposition, entanglement, etc., are merely employed to hopefully increase the efficiency or security of the protocol. Typical examples include super-dense coding [6], quantum coin-flipping protocol [4], and quantum key distribution protocols [3, 4].

The distinct advantage of super-operator weighted Markov chains, for model checking purpose, is twofold: (1) It provides a way to check *once for all* in that once a property is checked to hold, it holds for all input quantum states. This is especially important for the verification of quantum programs. For example, for the reachability problem we calculate the accumulated *super-operator*, say $\mathcal{E}$, along all valid paths. As a result, the reachability *probability* when the program is executed on the input quantum state ρ is simply the trace $\mathrm{tr}(\mathcal{E}(\rho))$ of $\mathcal{E}(\rho)$; (2) As the state space is usually finite, techniques from classical model checking can be adapted to verification of quantum systems.

The contribution of this paper is the development of a software tool that implements the techniques and algorithms proposed in [10]. The implementation is based on ISCASMC [13], a web-based model checker for probabilistic systems.

Other related works. Besides the model checker proposed by Gay et al. [12], recently Ardeshir-Larijani et al. developed equivalence checkers for deterministic quantum protocols [1] as well as concurrent quantum protocols that behave *functionally* [2]. As for [12], these tools work only within the stabiliser formalism, and the generalisation to general quantum protocols seems difficult.

2 The QMC Model and the Logic QCTL

In this section, we recall the notion of quantum Markov chains that serves as the semantic model of quantum programs and protocols. We assume the readers are familiar with the basic notions of quantum information theory. We give a brief introduction in Appendix A for their convenience. For more details, we refer to [10, 16].

Let $\mathcal{S}(\mathcal{H})$ be the set of *super-operators* over a Hilbert space $\mathcal{H}$. Here a super-operator is a completely positive linear operator from $\mathcal{L}(\mathcal{H})$ to itself, where $\mathcal{L}(\mathcal{H})$ is the set of linear operators on $\mathcal{H}$. In particular, we denote by $\mathcal{I}_\mathcal{H}$ and $0_\mathcal{H}$ the identity and null super-operators in $\mathcal{S}(\mathcal{H})$, respectively. For any $\mathcal{E}, \mathcal{F} \in \mathcal{S}(\mathcal{H})$, let $\mathcal{E} \lesssim \mathcal{F}$ if for any quantum state ρ in $\mathcal{H}$, $\mathrm{tr}(\mathcal{E}(\rho)) \leq \mathrm{tr}(\mathcal{F}(\rho))$. Note that the trace tr of a (unnormalised) quantum state denotes the probability that the (normalised) state is reached [17]. Intuitively, $\mathcal{E} \lesssim \mathcal{F}$ means that the success probability of performing $\mathcal{E}$ is always not greater than that of performing $\mathcal{F}$, whatever the initial state is. Let $\eqsim$ be $\lesssim \cap \gtrsim$.

We denote by $\mathcal{S}^\mathcal{I}(\mathcal{H})$ the set of trace-nonincreasing super-operators over $\mathcal{H}$; that is, $\mathcal{S}^\mathcal{I}(\mathcal{H}) = \{\, \mathcal{E} \in \mathcal{S}(\mathcal{H}) \mid 0_\mathcal{H} \lesssim \mathcal{E} \lesssim \mathcal{I}_\mathcal{H} \,\}$. Observe that $\mathcal{E} \in \mathcal{S}^\mathcal{I}(\mathcal{H})$ if and only if for any quantum state ρ, $\mathrm{tr}(\mathcal{E}(\rho)) \in [0, 1]$. It is natural to regard the set $\mathcal{S}^\mathcal{I}(\mathcal{H})$ as

the quantum correspondence of $[0,1]$, the domain of traditional probabilities. This is exactly the key to the notion of quantum Markov chains defined in [10].

Definition 1 (Quantum Markov Chain, [10]). *A super-operator weighted Markov chain, also referred to as quantum Markov chain (QMC) for simplicity, over a Hilbert space $\mathcal{H}$ is a tuple $(S, \mathbf{Q}, AP, L)$, where*

(1) S is a countable (typically finite) set of classical states*;*
(2) $\mathbf{Q}\colon S \times S \to \mathcal{S}^{\mathcal{I}}(\mathcal{H})$ is called the transition matrix *where for each $s \in S$, the super-operator $\sum_{t\in S} \mathbf{Q}(s,t)$ is trace-preserving;*
(3) AP is a finite set of atomic propositions*; and*
(4) $L\colon S \to 2^{AP}$ is a labelling function*.*

From the above definition, a QMC is simply a discrete time Markov chain (DTMC) with all traditional probabilities replaced by *quantum probabilities* from $\mathcal{S}^{\mathcal{I}}(\mathcal{H})$. The properties are expressed using the quantum computation tree logic (QCTL) proposed in [10], which is a natural extension of PCTL. The syntax of QCTL is as follows:

$$\Phi ::= \ a \mid \neg\Phi \mid \Phi \wedge \Phi \mid \mathbb{Q}_{\sim\mathcal{E}}[\phi]$$
$$\phi ::= \mathbf{X}\Phi \mid \Phi\ \mathbf{U}^{\leq k}\ \Phi \mid \Phi\ \mathbf{U}\ \Phi$$

where $a \in AP$ is an atomic proposition, $\sim \in \{\lesssim, \gtrsim, \eqsim\}$, $\mathcal{E} \in \mathcal{S}^{\mathcal{I}}(\mathcal{H})$, and $k \in \mathbb{N}$. We call Φ a *state formula* and ϕ a *path formula*. We use the following abbreviations: $\Phi_1 \vee \Phi_2 \equiv \neg(\neg\Phi_1 \wedge \neg\Phi_2)$, $\mathtt{tt} \equiv a \vee \neg a$, $\mathbf{F}\Phi \equiv \mathtt{tt}\ \mathbf{U}\ \Phi$, and $\mathbf{F}^{\leq k}\Phi \equiv \mathtt{tt}\ \mathbf{U}^{\leq k}\ \Phi$.

Note the essential difference between QCTL and the traditional PCTL:

- in PCTL we have the probabilistic operator formula $\mathbb{P}_{\sim p}[\phi]$ with $\sim \in \{\leq, \geq\}$, which asserts that the probability of paths from a certain state satisfying the path formula ϕ is constrained by $\sim p$ where $0 \leq p \leq 1$,
- in QCTL, $\mathbb{P}_{\sim p}[\phi]$ is replaced by $\mathbb{Q}_{\sim\mathcal{E}}[\phi]$, which asserts that the accumulated super-operators corresponding to paths from a certain state satisfying the formula ϕ is constrained by $\sim \mathcal{E}$ where $0_{\mathcal{H}} \lesssim \mathcal{E} \lesssim \mathcal{I}_{\mathcal{H}}$.

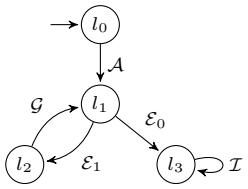
Note that $\mathbb{P}_{\sim p}[\phi]$ is a special case of $\mathbb{Q}_{\sim\mathcal{E}}[\phi]$ by taking $\mathcal{E} = p\mathcal{I}_{\mathcal{H}}$.

Example 1. A simple quantum loop program goes as follows:

$$\begin{aligned} &l_0 : q := \mathcal{A}(q)\\ &l_1 : \mathbf{while}\ M[q]\ \mathbf{do}\\ &l_2 : \qquad q := \mathcal{G}(q)\\ &l_3 : \mathbf{end}\end{aligned}$$

where for $E_0 = |0\rangle\langle 0| + \frac{1}{\sqrt{2}}|1\rangle\langle 1|$ and $E_1 = \frac{1}{\sqrt{2}}|0\rangle\langle 1|$, $\mathcal{A} = \{E_0, E_1\}$ is the $\frac{1}{2}$-amplitude damping channel, $M = \lambda_0|0\rangle\langle 0| + \lambda_1|1\rangle\langle 1|$, and $\mathcal{G}$ is the Hadamard super-operator. In this program, we first apply $\mathcal{A}$ on the quantum system q for the initialisation. At line l_1, the two-outcome projective measurement M is applied. If the outcome λ_0 is observed, then the program terminates at line l_3; otherwise it proceeds to l_2 where the super-operator $\mathcal{G}$ is performed, and then the program returns to line l_1 and another iteration continues.

The QMC for this program, depicted on the right, is constructed as follows. Let $S = AP = \{ l_i \mid 0 \leq i \leq 3 \}$, $L(l_i) = \{l_i\}$ for each i, and $\mathbf{Q}$ be defined as $\mathbf{Q}(l_0, l_1) = \mathcal{A}$, $\mathbf{Q}(l_1, l_3) = \mathcal{E}_0 = \{|0\rangle\langle 0|\}$, $\mathbf{Q}(l_1, l_2) = \mathcal{E}_1 = \{|1\rangle\langle 1|\}$, $\mathbf{Q}(l_2, l_1) = \mathcal{G}$, and $\mathbf{Q}(l_3, l_3) = \mathcal{I}$.



The QCTL formula $\mathbb{Q}_{\gtrsim \mathcal{E}}[\mathbf{F}\ l_3]$ asserts the probability that the loop program terminates is lower bounded by $\mathcal{E}$, that is, for any initial quantum state ρ, the termination probability is not less than $\mathrm{tr}(\mathcal{E}(\rho))$. In particular, the property that it always terminates for any input can be described as $\mathbb{Q}_{\gtrsim \mathcal{I}}[\mathbf{F}\ l_3]$.

3 The Tool QPMC

The QPMC model checker is the extension to QMCs of the web-based model checker IscasMc [13]. In order to support quantum operations, IscasMc has been enriched with the data structures for matrices and super-operators as well as the algorithms to manipulate them. Correspondingly, we have extended the PRISM [15] language used for modelling classical MCs with new keywords and operations specific for QMCs.

Implementation Aspect. IscasMc is written in Java with a few optional parts (which are not used for QMCs) being written in C. While the syntax of models is very close to the one of PRISM, the code of IscasMc is not based on the former.

The integration of QMCs into IscasMc has been possible because the underlying algorithms integrated into our model checker work with generic values rather than, for instance, restricting to computations with IEEE 754 double values. Thus, by defining the way of how mathematical operations are to be performed on super-operators, how they can be compared and how they can be displayed to the user, we are able to use algorithms already implemented in IscasMc. Thus, we can for instance use a variant of the well-known value iteration algorithms. Because QMCs do not behave exactly as DTMCs, some care has to be taken. Multiplication of super-operators is not commutative which needs to be taken into account in the value iteration. Also, the precomputation of states which reach target states with probability 1 has to be adapted.

IscasMc is split into several packages, to allow a clear separation between, for instance, the user interface, operations on mathematical objects, syntax trees of models and properties, etc. This way, extensions of one part are possible without interfering with the other modules. This allows for instance to quickly integrate additional operations on super-operators if they turn out to be useful for end users.

AT: added the following sentences

QPMC is available for evaluation at `http://iscasmc.ios.ac.cn/tool/` where it is possible to download the lastest stable version, together with a brief summary on the required dependencies and on the usage. QPMC is also accessible via the web interface at `http://iscasmc.ios.ac.cn/` where the considered quantum models are already available for the user "qpmc" (with password "qpmc").

Modeling Language. We extend the well-known PRISM [15] guarded-command based language to model QMCs. Fig. 1 depicts the source code in our language that describes the quantum loop program in Example 1:

– The keyword **qmc** specifies the model type.

```
qmc // model type

const vector |p>_2 = (|0>_2 + |1>_2)/sqrt(2);
const matrix E0 = |0>_2 <0|_2 + |1>_2 <1|_2/sqrt(2);
const matrix E1 = |0>_2 <1|_2/sqrt(2);
const superoperator(2) ampdamp = << E0, E1 >>;

module loop
  s : [0..3] init 0;
  [] (s=0) -> ampdamp: (s'=1);
  [] (s=1) -> << M1 >> : (s'=2) + << M0 >> : (s'=3);
  [] (s=2) -> << HD >> : (s'=1);
  [] (s=3) -> (s'=3);
endmodule
```

Fig. 1. Source code for the QMC in Example 1

– In addition to the constants definable in PRISM, one can specify constants of types **vector**, **matrix**, and **superoperator**. Notably, QPMC supports the use of *bra-ket* notation which is standard for describing quantum states in quantum mechanics. Specifically, `|v>_n` denotes a vector in $\mathcal{H}_n$, the n-dimensional Hilbert space. To ease notations, we have predefined the computational basis `|0>_n`, ..., `|n-1>_n` for $\mathcal{H}_n$; that is, for each $0 \leq i < n$,

$$|i\rangle_n = (0, \cdots, 0, 1, 0, \cdots, 0)^T$$

where 1 appears at the $(i+1)$-th entry. These vectors can be used for free. The operations such as *inner product*, *outer product*, and *tensor product* over bra-ket vectors are denoted in the normal way. For example, `<0|1>_2` stands for $\langle 0|1\rangle$, `|0>_2 <1|_2` for $|0\rangle\langle 1|$ in $\mathcal{H}_2$, and `|0>_2 |1>_2` means $|01\rangle$ in $\mathcal{H}_2 \otimes \mathcal{H}_2$. We use Kraus operators collected in a pair of double angle brackets to represent a super-operator. For example, the following statement

```
const superoperator(2) ampdamp = << E0, E1 >>;
```

defines a super-operator named ampdamp in the Hilbert space $\mathcal{H}_2$ with the Kraus operators $\{E0, E1\}$. For convenience of the users, we predefined some useful matrices such as the n-dimensional *identity* matrix **ID**(n), the *Pauli* matrices **PX**, **PY**, **PZ**, the *Hadamard* matrix **HD**, the *control-not* matrix **CN**, the *Toffoli* matrix **TF**, the *Fredkin* matrix **FK**, the *swap* matrix **SW**, and the *phase-shift* matrix **PS**(θ). We also predefined the *measurement operators* with respect to the computational basis in $\mathcal{H}_2$: $\mathbf{M0} = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ and $\mathbf{M1} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$.

– The main behavior of the QMC is described in the **module** environment. It has a state variable s, and several guarded commands representing the system transitions. As in PRISM, each guarded command has a precondition, and a sum of updates. The only difference is that each update is associated with a super-operator.

Properties. To help reasoning, besides the logical operators presented in QCTL, we also support an extended operator Q=?[ϕ], where ϕ is a path formula, to calculate (the

matrix representation[1] of) the super-operator of satisfying ϕ. We further provide a function qeval(Q=?[ϕ], ρ) to compute the density operator obtained from applying the resultant super-operator on a given density operator ρ, and

$$\text{qprob(Q=?[}\phi\text{], }\rho) = \text{tr}(\text{qeval(Q=?[}\phi\text{], }\rho))$$

to calculate the probability of satisfying ϕ, starting from the quantum state ρ.

Quantum loop program. For the quantum program in Example 1, we check the following properties

```
Q>=1 [ F (s=3) ]
qeval(Q=?[F (s=3)], |p>_2 <p|_2)
qeval(Q=?[F (s=3)], ID(2)/2)
```

where the first one checks if $l_0 \models \mathbb{Q}_{\gtrsim\mathcal{I}}[\mathbf{F}\ l_3]$ and the last two show the output states when the inputs of the program are the pure state $|+\rangle\langle+|$ where $|+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and the maximally mixed state $I/2$, respectively. QPMC returns **true** for the first property and $\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ for the last two, as expected.

It is worth noting that the termination properties we checked here cannot be verified by the previous tools in [1, 2, 12] for the following two reasons: (1) the loop program employs an amplitude damping operation which does not belong to the Clifford group; (2) the program is an open system which takes an arbitrary quantum state as its input, and we are checking the termination for *any* input state.

Superdense coding protocol. Another protocol we analysed is the superdense coding protocol [6] that permits to use a single qubit to faithfully transmit two classical bits, under the hypothesis that a maximally entangled state is already shared between the sender and the receiver. As for the quantum loop example, QPMC establishes the correctness of the protocol by returning **true** for the property `Q>=1 [ F (succ)]`. That is, the success state, where the original message has been transmitted from Alice to Bob faithfully, will be reached for sure.

Quantum key distribution protocol. The third protocol we considered is (a simplified version of) the quantum key distribution protocol [4] that allows Alice and Bob to create and share a private key between them, in a provably secure way, without interacting with a trusted third party for the exchange. QPMC returns **true** for both the properties `Q<=0 [F (fail)]` and `Q=0.5 [F (succ)]`, meaning that BB84 never fails, and with probability exactly 0.5 (the best success probability one can achieve), it successfully terminates at a shared key.

4 Conclusion and Future Work

Based on the theoretical work in [10], we have presented QPMC, a model checker aiming at verification of quantum programs and quantum protocols. Compared with the

[1] The matrix representation of a super-operator $\{ E_i \mid i \in I \}$ in an n-dimensional Hilbert space $\mathcal{H}$ is an n^2 by n^2 matrix $\sum_{i\in I} E_i \otimes E_i^*$, where the complex conjugate is taken according to some orthonormal basis of $\mathcal{H}$.

existing model checkers for the same purpose in the literature, our tool is able to verify more general programs and protocols which are beyond the stabiliser formalism.

For further studies, we are going to use qCCS, a quantum extension of CCS developed by one of the authors and his colleagues [7–9, 19], as our modelling language. This will make the protocol description more intuitive and more readable.

References

1. E. Ardeshir-Larijani, S. Gay, and R. Nagarajan. Equivalence checking of quantum protocols. In *TACAS*, volume 7795 of *LNCS*, pages 478–492. 2013.
2. E. Ardeshir-Larijani, S. Gay, and R. Nagarajan. Verification of concurrent quantum protocols by equivalence checking. In *TACAS*, volume 8413 of *LNCS*, pages 500–514. 2014.
3. C. H. Bennett. Quantum cryptography using any two nonorthogonal states. *Physical Review Letters*, 68:3121, 1992.
4. C. H. Bennett and G. Brassard. Quantum cryptography: Public-key distribution and coin tossing. In *Proceedings of the IEEE International Conference on Computer, Systems and Signal Processing*, pages 175–179, 1984.
5. C. H. Bennett, G. Brassard, C. Crepeau, R. Jozsa, A. Peres, and W. Wootters. Teleporting an unknown quantum state via dual classical and EPR channels. *Physical Review Letters*, 70:1895–1899, 1993.
6. C. H. Bennett and S. J. Wiesner. Communication via one- and two-particle operators on Einstein-Podolsky-Rosen states. *Physical Review Letters*, 69(20):2881–2884, 1992.
7. Y. Feng, R. Duan, Z. Ji, and M. Ying. Probabilistic bisimulations for quantum processes. *Information and Computation*, 205(11):1608–1639, 2007.
8. Y. Feng, R. Duan, and M. Ying. Bisimulations for quantum processes. In M. Sagiv, editor, *Proceedings of the 38th ACM Symposium on Principles of Programming Languages (POPL'11)*, pages 523–534, 2011.
9. Y. Feng, R. Duan, and M. Ying. Bisimulation for Quantum Processes. *ACM Transactions on Programming Languages and Systems*, 34(4):1–43, Dec. 2012.
10. Y. Feng, N. Yu, and M. Ying. Model checking quantum Markov chains. *Journal of Computer and System Sciences*, 79(7):1181–1198, 2013.
11. S. Gay, R. Nagarajan, and N. Papanikolaou. Probabilistic model-checking of quantum protocols. In *Proceedings of the 2nd International Workshop on Developments in Computational Models*, 2006.
12. S. Gay, R. Nagarajan, and N. Papanikolaou. QMC: A model checker for quantum systems. In *CAV 08*, pages 543–547. Springer, 2008.
13. E. M. Hahn, Y. Li, S. Schewe, A. Turrini, and L. Zhang. ISCASMC: A web-based probabilistic model checker. In *FM 2014: Formal Methods*, pages 312–317. 2014.
14. K. Kraus. *States, Effects and Operations: Fundamental Notions of Quantum Theory*. Springer, Berlin, 1983.
15. M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, volume 6806 of *LNCS*, pages 585–591, 2011.
16. M. Nielsen and I. Chuang. *Quantum computation and quantum information*. Cambridge university press, 2000.
17. P. Selinger. Towards a quantum programming language. *Mathematical Structures in Computer Science*, 14(4):527–586, 2004.
18. J. von Neumann. *Mathematical Foundations of Quantum Mechanics*. Princeton University Press, Princeton, NJ, 1955.
19. M. Ying, Y. Feng, R. Duan, and Z. Ji. An algebra of quantum processes. *ACM Transactions on Computational Logic (TOCL)*, 10(3):1–36, 2009.

A Basic Notions from Quantum Information Theory

For convenience of the reader, we briefly recall some basic notions from linear algebra and quantum information theory which are needed in this paper.

A.1 Basic Linear Algebra

A *Hilbert space* $\mathcal{H}$ is a vector space which is complete with respect to an inner product

$$\langle \cdot | \cdot \rangle \colon \mathcal{H} \times \mathcal{H} \to \mathbb{C}$$

such that

(1) $\langle \psi | \psi \rangle \geq 0$ for any $|\psi\rangle \in \mathcal{H}$, with equality if and only if $|\psi\rangle = 0$;
(2) $\langle \phi | \psi \rangle = \langle \psi | \phi \rangle^*$;
(3) $\langle \phi | \sum_i c_i |\psi_i\rangle = \sum_i c_i \langle \phi | \psi_i \rangle$,

where $\mathbb{C}$ is the set of complex numbers, and for each $c \in \mathbb{C}$, c^* stands for the complex conjugate of c. For any vector $|\psi\rangle \in \mathcal{H}$, its length $\||\psi\rangle\|$ is defined to be $\sqrt{\langle \psi | \psi \rangle}$, and it is said to be *normalised* if $\||\psi\rangle\| = 1$. Two vectors $|\psi\rangle$ and $|\phi\rangle$ are *orthogonal* if $\langle \psi | \phi \rangle = 0$. An *orthonormal basis* of a Hilbert space $\mathcal{H}$ is a basis $\{|i\rangle\}$ where each $|i\rangle$ is normalised and any pair of them is orthogonal.

Let $\mathcal{L}(\mathcal{H})$ be the set of linear operators on $\mathcal{H}$. For any $A \in \mathcal{L}(\mathcal{H})$, A is *Hermitian* if $A^\dagger = A$ where $A^\dagger$ is the adjoint operator of A such that $\langle \psi | A^\dagger | \phi \rangle = \langle \phi | A | \psi \rangle^*$ for any $|\psi\rangle, |\phi\rangle \in \mathcal{H}$. The fundamental *spectral theorem* states that a set of normalised eigenvectors of a Hermitian operator in $\mathcal{L}(\mathcal{H})$ constitutes an orthonormal basis for $\mathcal{H}$. That is, there exists a so-called spectral decomposition for each Hermitian A such that

$$A = \sum_i \lambda_i |i\rangle\langle i| = \sum_{\lambda_i \in spec(A)} \lambda_i E_i$$

where the set $\{|i\rangle\}$ constitutes an orthonormal basis of $\mathcal{H}$, $spec(A)$ denotes the set of eigenvalues of A, and E_i is the projector to the corresponding eigenspace of λ_i. A linear operator $A \in \mathcal{L}(\mathcal{H})$ is *unitary* if $A^\dagger A = A A^\dagger = \mathcal{I}_\mathcal{H}$ where $\mathcal{I}_\mathcal{H}$ is the identity operator on $\mathcal{H}$. The *trace* of A is defined as $\mathrm{tr}(A) = \sum_i \langle i | A | i \rangle$ for some given orthonormal basis $\{|i\rangle\}$ of $\mathcal{H}$. It is worth noting that the trace function is actually independent of the chosen orthonormal basis. It is also easy to check that the trace function is linear and $\mathrm{tr}(AB) = \mathrm{tr}(BA)$ for any operators $A, B \in \mathcal{L}(\mathcal{H})$.

Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be two Hilbert spaces. Their *tensor product* $\mathcal{H}_1 \otimes \mathcal{H}_2$ is defined as a vector space consisting of linear combinations of the vectors $|\psi_1 \psi_2\rangle = |\psi_1\rangle|\psi_2\rangle = |\psi_1\rangle \otimes |\psi_2\rangle$ with $|\psi_1\rangle \in \mathcal{H}_1$ and $|\psi_2\rangle \in \mathcal{H}_2$. Here the tensor product operator $\otimes$ maps two component vectors into a new vector such that

$$\left(\sum_i \lambda_i |\psi_i\rangle \right) \otimes \left(\sum_j \mu_j |\phi_j\rangle \right) = \sum_{i,j} \lambda_i \mu_j |\psi_i\rangle \otimes |\phi_j\rangle.$$

Then $\mathcal{H}_1 \otimes \mathcal{H}_2$ is also a Hilbert space where the inner product is defined as the following: for any $|\psi_1\rangle, |\phi_1\rangle \in \mathcal{H}_1$ and $|\psi_2\rangle, |\phi_2\rangle \in \mathcal{H}_2$,

$$\langle \psi_1 \otimes \psi_2 | \phi_1 \otimes \phi_2 \rangle = \langle \psi_1 | \phi_1 \rangle_{\mathcal{H}_1} \langle \psi_2 | \phi_2 \rangle_{\mathcal{H}_2}$$

where $\langle \cdot | \cdot \rangle_{\mathcal{H}_k}$ is the inner product of $\mathcal{H}_k$, $k = 1, 2$. For any $A_1 \in \mathcal{L}(\mathcal{H}_1)$ and $A_2 \in \mathcal{L}(\mathcal{H}_2)$, $A_1 \otimes A_2$ is defined as a linear operator in $\mathcal{L}(\mathcal{H}_1 \otimes \mathcal{H}_2)$ such that for each $|\psi_1\rangle \in \mathcal{H}_1$ and $|\psi_2\rangle \in \mathcal{H}_2$,

$$(A_1 \otimes A_2)|\psi_1\psi_2\rangle = A_1|\psi_1\rangle \otimes A_2|\psi_2\rangle.$$

The *partial trace* of $A \in \mathcal{L}(\mathcal{H}_1 \otimes \mathcal{H}_2)$ with respected to $\mathcal{H}_k$, $k = 1, 2$, is defined as $\mathrm{tr}_{\mathcal{H}_k}(A) = \sum_i \langle i|A|i\rangle$ where $\{|i\rangle\}$ is an orthonormal basis of $\mathcal{H}_k$. As for the trace function, also the partial trace functions are independent of the chosen orthonormal basis.

Traditionally, a linear operator $\mathcal{E}$ on $\mathcal{L}(\mathcal{H})$ is called a *super-operator* on $\mathcal{H}$. A super-operator is said to be *completely positive* if it maps positive operators in $\mathcal{L}(\mathcal{H})$ to positive operators in $\mathcal{L}(\mathcal{H})$, and for any auxiliary Hilbert space $\mathcal{H}'$, the trivially extended operator $\mathcal{I}_{\mathcal{H}'} \otimes \mathcal{E}$ also maps positive operators in $\mathcal{L}(\mathcal{H}' \otimes \mathcal{H})$ to positive operators in $\mathcal{L}(\mathcal{H}' \otimes \mathcal{H})$. Here $\mathcal{I}_{\mathcal{H}'}$ is the identity operator on $\mathcal{L}(\mathcal{H}')$. We always assume complete positivity for super-operators in this paper. The elegant and powerful *Kraus representation theorem* [14] of completely positive super-operators states that a super-operator $\mathcal{E}$ is completely positive if and only if there is some set of operators $\{ E_i \mid i \in I \}$ with appropriate dimension such that

$$\mathcal{E}(A) = \sum_{i \in I} E_i A E_i^\dagger$$

for any $A \in \mathcal{L}(\mathcal{H})$. The operators E_i are called Kraus operators of $\mathcal{E}$. We abuse the notation slightly by denoting $\mathcal{E} = \{ E_i \mid i \in I \}$. It is worth noting that the set of Kraus operators is not unique and we can always take one such that the number of Kraus operators does not exceed d^2 where d is the dimension of the Hilbert space. A super-operator is said to be *trace-preserving* if $\mathrm{tr}(\mathcal{E}(A)) = \mathrm{tr}(A)$ for any positive $A \in \mathcal{L}(\mathcal{H})$; equivalently, its Kraus operators E_i satisfy $\sum_i E_i^\dagger E_i = I$.

A.2 Basic Quantum Mechanics

According to von Neumann's formalism of quantum mechanics [18], an isolated physical system is associated with a Hilbert space which is called the *state space* of the system. A *pure state* of a quantum system is a normalised vector in its state space, and a *mixed state* is represented by a density operator on the state space. Here a density operator ρ on the Hilbert space $\mathcal{H}$ is a positive linear operator such that $\mathrm{tr}(\rho) = 1$. Another equivalent representation of density operator is a probabilistic ensemble of pure states. In particular, given an ensemble $\{(p_i, |\psi_i\rangle)\}$ where $p_i \geq 0$, $\sum_i p_i = 1$, and $|\psi_i\rangle$ are pure states, then $\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$ is a density operator. Conversely, each density operator can be generated by an ensemble of pure states in this way. Let $\mathcal{D}(\mathcal{H})$ denote the set of density operators on $\mathcal{H}$.

The state space of a composite system (for example, a quantum system consisting of many qubits) is the tensor product of the state spaces of its components. For a mixed state ρ on $\mathcal{H}_1 \otimes \mathcal{H}_2$, partial traces of ρ have explicit physical meanings: the density operators $\mathrm{tr}_{\mathcal{H}_1}(\rho)$ and $\mathrm{tr}_{\mathcal{H}_2}(\rho)$ are exactly the reduced quantum states of ρ on the second and the first component system, respectively. Note that in general, the state of a composite

system cannot be decomposed into the tensor product of the reduced states on its component systems. A well-known example is the 2-qubit state $|\Psi_0\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. This kind of state is called *entangled state*. To see the strangeness of entanglement, suppose a measurement $M = \lambda_0|0\rangle\langle 0| + \lambda_1|1\rangle\langle 1|$ is applied on the first qubit of $|\Psi_0\rangle$ (see the following for the definition of quantum measurements). Then after the measurement, the second qubit will definitely collapse into state $|0\rangle$ or $|1\rangle$, depending on whether the outcome λ_0 or λ_1 is observed. In other words, the measurement on the first qubit changes the state of the second qubit in some way. This is an outstanding feature of quantum mechanics which has no counterpart in classical world, and is the key to many quantum information processing tasks such as teleportation [5] and superdense coding [6].

The evolution of a closed quantum system is described by a unitary operator on its state space: if the states of the system at times t_1 and t_2 are ρ_1 and ρ_2, respectively, then $\rho_2 = U\rho_1 U^\dagger$ for some unitary operator U which depends only on t_1 and t_2. In general, the dynamics which can occur in a physical system is described by a trace-preserving super-operator on its state space.

A quantum *measurement* is described by a collection $\{M_m\}$ of measurement operators, where the indices m refer to the measurement outcomes. It is required that the measurement operators satisfy the completeness equation $\sum_m M_m^\dagger M_m = I_{\mathcal{H}}$. If the system is in state ρ, then the probability that the measurement result m occurs is given by $p(m) = \mathrm{tr}(M_m \rho M_m^\dagger)$, and the state of the post-measurement system is $\frac{M_m \rho M_m^\dagger}{p(m)}$.

A particular case of measurement is the *projective measurement* which is usually represented by a Hermitian operator. Let M be a Hermitian operator and

$$M = \sum_{m \in spec(M)} mE_m \tag{1}$$

its spectral decomposition. Obviously, the projectors $\{ E_m \mid m \in spec(M) \}$ form a quantum measurement. If the state of a quantum system is ρ, then the probability that result m occurs when measuring M on the system is $p(m) = \mathrm{tr}(E_m \rho)$, and the post-measurement state of the system is $\frac{E_m \rho E_m}{p(m)}$. Note that for each outcome m, the map $\mathcal{E}_m(\rho) = E_m \rho E_m$ is a super-operator by Kraus Theorem.

B The Case Studies in Detail

Below we present in detail the BB84 quantum key distribution and the superdense coding protocols from [10] to show the utility of our tool.

B.1 Quantum Key-Distribution

BB84, the first quantum key distribution protocol developed by Bennett and Brassard in 1984 [4], provides a provably secure way to create a private key between two parties, say, Alice and Bob. The basic BB84 protocol goes as follows:

(1) Alice randomly creates two strings of bits $\tilde{B}_a$ and $\tilde{K}_a$, each with size n.
(2) Alice prepares a string of qubits $\tilde{q}$, with size n, such that the i-th qubit of $\tilde{q}$ is $|x_y\rangle$ where x and y are the i-th bits of $\tilde{B}_a$ and $\tilde{K}_a$, respectively, and $|0_0\rangle = |0\rangle$, $|0_1\rangle = |1\rangle$, $|1_0\rangle = |+\rangle = \frac{|0\rangle+|1\rangle}{\sqrt{2}}$, and $|1_1\rangle = |-\rangle = \frac{|0\rangle-|1\rangle}{\sqrt{2}}$.

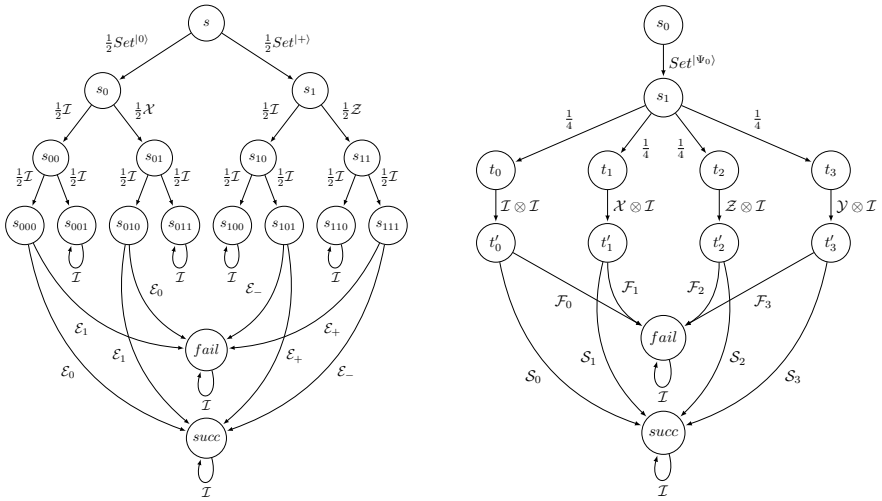


Fig. 2. QMCs for the BB84 protocol when $n = 1$ (left) and the superdense coding protocol (right).

(3) Alice sends the qubit string $\tilde{q}$ to Bob.
(4) Bob randomly generates a string of bits $\tilde{B}_b$ with size n.
(5) Bob measures each qubit received from Alice according to the basis determined by the bits he generated: if the i-th bit of $\tilde{B}_b$ is k then he measures the i-th qubit of $\tilde{q}$ with $\{|k_0\rangle, |k_1\rangle\}$. Let the measurement results be $\tilde{K}_b$, which is also a string of bits with size n.
(6) Bob sends his choice of measurement bases $\tilde{B}_b$ back to Alice, and upon receiving the information, Alice sends her bases $\tilde{B}_a$ to Bob.
(7) Alice and Bob determine at which positions the bit strings $\tilde{B}_a$ and $\tilde{B}_b$ are equal. They discard the bits in $\tilde{K}_a$ and $\tilde{K}_b$ where the corresponding bits of $\tilde{B}_a$ and $\tilde{B}_b$ do not match.

After the execution of the basic BB84 protocol above, the remaining bits of $\tilde{K}_a$ and $\tilde{K}_b$ should be the same, provided that the communication channels used are perfect, and no eavesdropper exists.

The QMC for the basic BB84 protocol in the simplest case of $n = 1$ is depicted in Fig. 2 (left), where $Set^{|\psi\rangle}$ is the 1-qubit super-operator which sets the target qubit to $|\psi\rangle$, $\mathcal{X}$ and $\mathcal{Z}$ are respectively the Pauli-X and Pauli-Z super-operators, and $\mathcal{E}_i = \{|i\rangle\langle i|\}$, $i = 0, 1, +, -$. We use the subscripts for the s-states to denote the choices of the basis B_a of Alice, the key K_a generated by Alice, and the basis B_b guessed by Bob. For example, in s_0, $B_a = 0$; in s_{01}, $B_a = 0$ and $K_a = 1$; and in s_{101}, $B_a = B_b = 1$ and $K_a = 0$. Let $AP = S \cup \{abort\}$ and $L(s) = \{abort\}$ if $s \in \{s_{001}, s_{011}, s_{100}, s_{110}\}$, meaning that at these states Alice's and Bob's bases differ, so the protocol will be aborted without generating any key. For other states s, we let $L(s) = \{s\}$ naturally.

We use the states $succ$ and $fail$ to denote the successful and unsuccessful termination of BB84 protocol, respectively. We take the state s_{101} as an example to illustrate

```
qmc

const vector |p>_2 = (|0>_2 + |1>_2)/sqrt(2); //|+> state
const vector |m>_2 = (|0>_2 - |1>_2)/sqrt(2); //|-> state
const superoperator(2) setplus = <<|p>_2 <0|_2,|p>_2 <1|_2>>; // Set the target
    qubit to |+>
const superoperator(2) set0 = <<|0>_2 <0|_2,|0>_2 <1|_2>>; // Set the target qubit
    to |0>
const superoperator(2) measurep = <<|p>_2 <p|_2>>; //measure in {|+>, |->} and get
    result +
const superoperator(2) measurem = <<|m>_2 <m|_2>>; //measure in {|+>, |->} and get
    result -

module BB84

  s : [0..16] init 0;
//Alice prepares her state equiprobably from |0>, |1>, |+>, |->
  [] (s=0) -> 0.5*set0 : (s'=1) + 0.5*setplus : (s'=2);
  [] (s=1) -> 0.5 : (s'=3)            //Alice's state = |0>
            + 0.5*<< PX >> : (s'=4); //Alice's state = |1>
  [] (s=2) -> 0.5 : (s'=5)            //Alice's state = |+>
            + 0.5*<< PZ >> : (s'=6);  //Alice's sate = |->
//Bob guesses his measurement basis
  [] (s>2) & (s<7) -> 0.5 : (s'=2*s+1)  //Bob guesses basis {|0>, |1>}
                    + 0.5 : (s'=2*s+2); //Bob guesses basis {|+>, |->}
//Bob measures his received state
  [] (s=7) -> << M0 >> : (s'=16)        //Outcome= 0. Succeeded
            + << M1 >> : (s'=15);         //Outcome = 1. Failed
  [] (s=9) -> << M0 >> : (s'=15)          //Outcome = 0. Failed
            + << M1 >> : (s'=16);       //Outcome = 1. Succeeded
  [] (s=12) -> measurep : (s'=16)       //Outcome = +. Succeeded
            +  measurem : (s'=15);        //Outcome = -. Failed
  [] (s=14) -> measurep : (s'=15)         //Outcome = +. Failed
            +  measurem : (s'=16);      //Outcome = -. Succeeded
  [] (s=8)|(s=10)|(s=11)|(s=13) -> (s'=s); //Bob guesses the wrong basis. Aborted.
  [] (s=15) -> (s'=15);                     //Protocol failed
  [] (s=16) -> (s'=16);                   //Protocol succeeded

endmodule

formula succ = (s=16);
formula fail = (s=15);
formula abort = (s=8)|(s=10)|(s=11)|(s=13);
```

Fig. 3. Source code for the QMC of Quantum key-distribution

the basic idea. As the bases of Alice and Bob are both $\{|+\rangle, |-\rangle\}$ at s_{101}, they will regard the key bit as the final key generated by the protocol. Thus if the outcome of Bob's measurement is 0, which corresponds to the super-operator $\mathcal{E}_+$, then the protocol succeeds since Alice and Bob indeed share the same key bit 0; otherwise the protocol fails as they end up with different bits: Alice with 0 while Bob with 1. That explains why we have $\mathbf{Q}(s_{101}, succ) = \mathcal{E}_+$ while $\mathbf{Q}(s_{101}, fail) = \mathcal{E}_-$.

The correctness of basic BB84 protocol can be stated as

$$s \models \mathbb{Q}_{\lesssim 0_{\mathcal{H}}}[\mathbf{F}\, fail] \wedge \mathbb{Q}_{\gtrsim \mathcal{I}_{\mathcal{H}}}[\mathbf{F}(succ \vee abort)] \wedge \mathbb{Q}_{\approx \frac{1}{2}\mathcal{I}_{\mathcal{H}}}[\mathbf{F}\, succ] \wedge \mathbb{Q}_{\approx \frac{1}{2}\mathcal{I}_{\mathcal{H}}}[\mathbf{F}\, abort],$$

which asserts that the protocol never (with probability 0) fails and always completes with either success or abort, each one occurring with probability one half. As there is

only a half chance for Bob to correctly guess Alice's basis, the probability of successfully establishing a key cannot exceed $\frac{1}{2}$.

Verification result. The source code for the superdense coding protocol is listed in Fig. 3, and the properties we need to check are

```
Q<=0 [F (fail)]
Q>=1 [F(succ | abort)]
Q=0.5 [F (succ)] & Q=0.5 [F (abort)]
```

QPMC returns **true** for all the properties, as well as their conjunction, as expected.

B.2 Superdense Coding Protocol

Superdense coding is a well-known protocol presented by Bennett and Wiesner [6], in which two bits of classical information can be faithfully transmitted by sending only one qubit, provided that a maximally entangled state is shared *a priori* between the sender and the receiver. The superdense coding protocol goes as follows:

(1) Alice and Bob prepare a maximally entangled state $|\Psi_0\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$ on q_1 and q_2. Alice holds q_1 while Bob holds q_2;
(2) Depending on which message among the four possibilities $\{0, 1, 2, 3\}$ Alice wishes to transmit, she applies one of the four Pauli super-operators $\mathcal{I}$, $\mathcal{X}$, $\mathcal{Z}$, and $\mathcal{Y}$ on her qubit q_1, and sends it to Bob;
(3) Upon receiving q_1, Bob performs a Bell-basis measurement $M = \sum_{i=0}^{3} i|\Psi_i\rangle\langle\Psi_i|$ on q_1 and q_2, where

$$|\Psi_1\rangle = \frac{|01\rangle + |10\rangle}{\sqrt{2}},$$
$$|\Psi_2\rangle = \frac{|00\rangle - |11\rangle}{\sqrt{2}},$$
$$|\Psi_3\rangle = \frac{|01\rangle - |10\rangle}{\sqrt{2}}.$$

Bob interprets the measurement outcome as the message sent by Alice.

The QMC which describes the behaviour of superdense coding protocol is depicted in Fig. 2 (right) where $Set^{|\Psi_0\rangle}$ is the 2-qubit super-operator which sets the target qubits q_1, q_2 to $|\Psi_0\rangle$, $\mathcal{S}_i = \{|\Psi_i\rangle\langle\Psi_i|\}$, and $\mathcal{F}_i = \{\, |\Psi_j\rangle\langle\Psi_j| \mid j \neq i \,\}$. For simplicity, we assume that Alice chooses the message she wishes to send equiprobably. The atomic proposition set AP is the same as S, and $L(s) = \{s\}$ for each $s \in S$. Then the correctness of superdense coding protocol can be stated as

$$s_0 \models \mathbb{Q}_{\gtrsim \mathcal{I}}[\mathbf{F}\ succ].$$

Verification result. The source code for the superdense coding protocol is listed in Fig. 4, and the property we need to check is

```
Q>=1 [F (succ)]
```

QPMC returns **true** as expected.

```
qmc

const vector |b00>_2 = (|0>_2|0>_2 + |1>_2|1>_2)/sqrt(2);
const vector |b01>_2 = (|0>_2|1>_2 + |1>_2|0>_2)/sqrt(2);
const vector |b10>_2 = (|0>_2|0>_2 - |1>_2|1>_2)/sqrt(2);
const vector |b11>_2 = (|0>_2|1>_2 - |1>_2|0>_2)/sqrt(2);

const matrix B00 = |b00>_2 <b00|_2;
const matrix B01 = |b01>_2 <b01|_2;
const matrix B10 = |b10>_2 <b10|_2;
const matrix B11 = |b11>_2 <b11|_2;

module sdc

s : [0..11] init 0;

//Preparation of Bell state (|00> + |11>)/sqrt(2)

  [] (s=0) -> << |b00>_2 <b00|_2, |b00>_2 <b01|_2, |b00>_2 <b10|_2,
                                                    |b00>_2 <b11|_2 >> : (s'=1);

//Alice chooses her message from 0,1,2,3 randomly
  [] (s=1) -> 0.25 : (s'=2) + 0.25 : (s'=3)
            + 0.25 : (s'=4) + 0.25 : (s'=5);

//Alice applies the corresponding super-operator on her qubit depending on the
    message she chose
  [] (s=2) ->  (s'=6);                              //Message = 0

  [] (s=3) -> <<kron(PX, ID(2))>> : (s'=7);         //Message = 1

  [] (s=4) -> <<kron(PZ, ID(2))>> : (s'=8);         //Message = 2

  [] (s=5) -> <<kron(PY, ID(2))>> : (s'=9);         //Message = 3

//Bob performs the measurement in the basis of Bell states
  [] (s=6) -> <<B00>> : (s'=11)          //Outcome = 0. Succeeded
            + <<B01, B10, B11>> : (s'=10); //Outcome <> 0. Failed

  [] (s=7) -> <<B01>> : (s'=11)          //Outcome = 1. Succeeded
            + <<B00, B10, B11>> : (s'=10); //Outcome <> 1. Failed

  [] (s=8) -> <<B10>> : (s'=11)          //Outcome = 2. Succeeded
            + <<B00, B01, B11>> : (s'=10); //Outcome <> 2. Failed

  [] (s=9) -> <<B11>> : (s'=11)          //Outcome = 3. Succeeded
            + <<B00, B01, B10>> : (s'=10); //Outcome <> 3. Failed

  [] (s=10) -> (s'=10);                             //Protocol failed

  [] (s=11) -> (s'=11);                          //Protocol succeeded

endmodule

formula succ = (s=11);
```

Fig. 4. Source code for the QMC of Superdense coding