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, Sydney, Australia
[2] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
andrea.turrini@gmail.com

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/qmc/.

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. The security of quantum cryptographic protocols is mathematically provable, based on the principles of quantum mechanics, without imposing any restrictions on the computational capacity of an attacker. In practice, however, security analysis of quantum cryptographic protocols is notoriously difficult; for example, the manual proof of BB84 in [15] contains about 50 pages. It is hard to imagine such an analysis being carried out for more sophisticated quantum protocols. Thus, techniques for automated or semi-automated verification of these protocols will be indispensable.

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. [10] 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 [11] for quantum protocols, employing purely classical algorithms.

N. Bjørner and F. de Boer (Eds.): FM 2015, LNCS 9109, pp. 265–272, 2015.
DOI: 10.1007/978-3-319-19249-9_17

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 [9]. 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 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 verified, 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 [9]. The implementation is based on IsCASMC [12], a web-based model checker for probabilistic systems.

Other related works. Besides the model checker proposed by Gay et al. [11], 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 [11], 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 [9,16].

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 [9].

Definition 1 (Quantum Markov Chain [9]). *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 [9], 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 $0_{\mathcal{H}} \lesssim \mathcal{E} \lesssim \mathcal{I}_{\mathcal{H}}$.

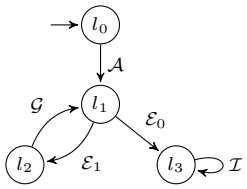
Note that $\mathbb{P}_{\sim p}[\phi]$ is a special case of $\mathbb{Q}_{\sim\mathcal{E}}[\phi]$ by taking $\mathcal{E} = p\mathcal{I}_{\mathcal{H}}$.

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 [12]. 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 [14] 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, being restricted 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.

Complex numbers, matrices and super-operators are stored using IEEE 754 doubles and manipulated using standard operations. QPMC could be extended to instead use representations of numbers with higher or infinite precision (using symbolic representations) e.g. for stiff models. Doing so would not affect the rest of the implementation.

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.

QPMC is available at http://iscasmc.ios.ac.cn/tool/qmc/ where it is possible to download the latest stable version, together with a brief summary on the required dependencies and on the usage.

```
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

Modeling Language. We extend the well-known PRISM [14] 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.
- 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 $0 \leq i < n$,

$$|\texttt{i}\texttt{>\_n} = (0, \cdots, 0, 1, 0, \cdots, 0)^T$$

  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 $|01\rangle = |0\rangle \otimes |1\rangle$ 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 listed below:
  - the n-dimensional *identity* matrix **ID**(n) = $diag(1, \ldots, 1)$;
  - the *Pauli* matrices **PX** = $\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$, **PY** = $\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}$, **PZ** = $\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$, the *Hadamard* matrix **HD** = $\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$, and the *phase-shift* matrix **PS**(θ) = $\begin{pmatrix} 1 & 0 \\ 0 & e^{i\theta} \end{pmatrix}$. We also predefined the *measurement operators* with respect to the computational basis in $\mathcal{H}_2$: **M0** = $\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$ and **M1** = $\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$;

  - the *control-not* matrix **CN** = **ID**(2) $\oplus$ **PX**, and the *swap* matrix **SW** = **ID**(1) $\oplus$ **PX** $\oplus$ **ID**(1),
  - the *Toffoli* matrix **TF** = **ID**(4) $\oplus$ **CN**, and the *Fredkin* matrix **FK** = **ID**(4) $\oplus$ **SW**.

– 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 instead of a probability. We always omit the identity super-operators along the updates.

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

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

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, 11] 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

[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 $\mathcal{H}$. See [9] for more details.

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 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. For simplicity, we only consider the basic version of BB84 where the channels used are perfect, and no eavesdropper exists. Then the correctness of the protocol can be described by the properties **Q**<=0 [**F** (fail)] and **Q**=0.5 [**F** (succ)], meaning that BB84 never fails (i.e., it always generates identical keys between Alice and Bob), and with probability exactly 0.5 (the best success probability one can achieve), it successfully terminates at a shared key. QPMC returns **true** for both the properties.

Performance of the tool. Each experiment has been performed on a MacBook Pro with a 2.9 GHz Intel Core i7 processor with 8 GB 1600 MHz DDR3 RAM, taking an overall time of less than 1 second per model.

4 Conclusion and Future Work

Based on the theoretical work in [9], we have presented QPMC, a model checker aiming at verification of quantum programs and quantum protocols. Compared with the 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 [7,8,19], as our modelling language. This will make the protocol description more intuitive and more readable. We also plan to consider analysis of LTL properties. Classical decision algorithms check acceptance of bottom strongly connected components and then compute reachability probabilities for transient states. We would like to see how this technique can be extended to QMCs.

Acknowledgments. This work is supported by Australian Research Council (Grant No. DP130102764), the National Natural Science Foundation of China (Grant Nos. 61428208, 61472473 and 61361136002), the Chinese Academy of Sciences Fellowship for International Young Scientists (Grant Nos. 2013Y1GB0006 and 2015VTC029), AMSS-UTS Joint Research Laboratory for Quantum Computation, Chinese Academy of Sciences, and the CAS/SAFEA International Partnership Program for Creative Research Team.

References

1. Ardeshir-Larijani, E., Gay, S., Nagarajan, R.: Equivalence checking of quantum protocols. In: Piterman, N., Smolka, S.A. (eds.) TACAS 2013. LNCS, vol. 7795, pp. 478–492. Springer, Heidelberg (2013)

2. Ardeshir-Larijani, E., Gay, S., Nagarajan, R.: Verification of concurrent quantum protocols by equivalence checking. In: Ábrahám, E., Havelund, K. (eds.) TACAS 2014. LNCS, vol. 8413, pp. 500–514. Springer, Heidelberg (2014)
3. Bennett, C.H.: Quantum cryptography using any two nonorthogonal states. Physical Review Letters 68, 3121 (1992)
4. 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, pp. 175–179 (1984)
5. Bennett, C.H., Brassard, G., Crepeau, C., Jozsa, R., Peres, A., Wootters, W.: Teleporting an unknown quantum state via dual classical and EPR channels. Physical Review Letters 70, 1895–1899 (1993)
6. Bennett, C.H., Wiesner, S.J.: Communication via one- and two-particle operators on Einstein-Podolsky-Rosen states. Physical Review Letters 69(20), 2881–2884 (1992)
7. Feng, Y., Duan, R., Ji, Z., Ying, M.: Probabilistic bisimulations for quantum processes. Information and Computation 205(11), 1608–1639 (2007)
8. Feng, Y., Duan, R., Ying, M.: Bisimulation for Quantum Processes. ACM Transactions on Programming Languages and Systems 34(4), 1–43 (2012)
9. Feng, Y., Yu, N., Ying, M.: Model checking quantum Markov chains. Journal of Computer and System Sciences 79(7), 1181–1198 (2013)
10. Gay, S., Nagarajan, R., Papanikolaou, N.: Probabilistic model-checking of quantum protocols. In: Proceedings of the 2nd International Workshop on Developments in Computational Models (2006)
11. Gay, S.J., Nagarajan, R., Papanikolaou, N.: QMC: A model checker for quantum systems. In: Gupta, A., Malik, S. (eds.) CAV 2008. LNCS, vol. 5123, pp. 543–547. Springer, Heidelberg (2008)
12. Hahn, E.M., Li, Y., Schewe, S., Turrini, A., Zhang, L.: ISCASMC: A web-based probabilistic model checker. In: Jones, C., Pihlajasaari, P., Sun, J. (eds.) FM 2014. LNCS, vol. 8442, pp. 312–317. Springer, Heidelberg (2014)
13. Kraus, K.: States, Effects and Operations: Fundamental Notions of Quantum Theory. Springer, Berlin (1983)
14. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011)
15. Mayers, D.: Unconditional security in quantum cryptography. Journal of the ACM 48(3), 351–406 (2001)
16. Nielsen, M., Chuang, I.: Quantum computation and quantum information. Cambridge university press (2000)
17. Selinger, P.: Towards a quantum programming language. Mathematical Structures in Computer Science 14(4), 527–586 (2004)
18. von Neumann, J.: Mathematical Foundations of Quantum Mechanics. Princeton University Press, Princeton (1955)
19. Ying, M., Feng, Y., Duan, R., Ji, Z.: An algebra of quantum processes. ACM Transactions on Computational Logic (TOCL) 10(3), 1–36 (2009)