A Quantum Interpretation of Separating Conjunction for Local Reasoning of Quantum Programs Based on Separation Logic

XUAN-BACH LE, Nanyang Technological University, Singapore
SHANG-WEI LIN, Nanyang Technological University, Singapore
JUN SUN, Singapore Management University, Singapore
DAVID SANAN, Nanyang Technological University, Singapore

It is well-known that quantum programs are not only complicated to design but also challenging to verify because the quantum states can have exponential size and require sophisticated mathematics to encode and manipulate. To tackle the state-space explosion problem for quantum reasoning, we propose a Hoare-style inference framework that supports local reasoning for quantum programs. By providing a quantum interpretation of the separating conjunction, we are able to infuse separation logic into our framework and apply local reasoning using a quantum frame rule that is similar to the classical frame rule. For evaluation, we apply our framework to verify various quantum programs including Deutsch–Jozsa's algorithm and Grover's algorithm.

CCS Concepts: • **Software and its engineering** → **General programming languages**; • **Social and professional topics** → *History of programming languages.*

Additional Key Words and Phrases: Quantum Computing, Verification, Formal Semantics

ACM Reference Format:
Xuan-Bach Le, Shang-Wei Lin, Jun Sun, and David Sanan. 2022. A Quantum Interpretation of Separating Conjunction for Local Reasoning of Quantum Programs Based on Separation Logic. *Proc. ACM Program. Lang.* 6, POPL, Article 36 (January 2022), 27 pages. https://doi.org/10.1145/3498697

1 INTRODUCTION

Since quantum technology has progressed rapidly in the last decade, academia and industry started to build quantum computers [Google 2018a; IBM 2020]. Accordingly, there is an increasing number of quantum programming languages and associated compilers developed recently, e.g. Quipper [Green et al. 2013], Scaffold [JavadiAbhari et al. 2015], QWire [Paykin et al. 2017], Microsoft's LIQUi|⟩ [Wecker et al. 2014] and Q# [Svore et al. 2018], IBM's OpenQASM [Cross et al. 2017], Google's Cirq [Google 2018b], ProjectQ [Steiger et al. 2018], Chisel-Q [Liu and Kubiatowicz 2013], Q|SI⟩ [Liu et al. 2017], and Silq [Bichsel et al. 2020]. It can be expected that quantum programs will take over certain computation tasks from traditional programs in the near future, and thus ensuring the correctness of quantum program becomes relevant.

Unlike classical bit which can only be either 0 or 1 at a time, a quantum bit (qubit) can be in the *superposition* where both states 0 and 1 coexist simultaneously. In general, the superposition

Authors' addresses: Xuan-Bach Le, Nanyang Technological University, Singapore, bach.le@ntu.edu.sg; Shang-Wei Lin, Nanyang Technological University, Singapore, shang-wei.lin@ntu.edu.sg; Jun Sun, Singapore Management University, Singapore, junsun@smu.edu.sg; David Sanan, Nanyang Technological University, Singapore, sanan@ntu.edu.sg.

2475-1421/2022/1-ART36
https://doi.org/10.1145/3498697

$$\frac{\{P\}c\{Q\} \quad \text{free}(F) \cap \text{mod}(c) = \emptyset}{\{P \circledast F\}c\{Q \circledast F\}} \text{ QFrame}$$

Fig. 1. The quantum frame rule QFrame

of n qubits can contain up to 2^n classical states, therefore having the potential to speed up the computation exponentially. Evidently, many computation problems can be solved efficiently using quantum computation, such as Deutsch-Jozsa's algorithm [Deutsch and Jozsa 1992] for solving quantum black box, Grover's algorithm [Grover 1996] for searching element in an unordered list, or Shor's algorithm [Shor 1994] for integer factorization.

Nevertheless, quantum algorithms/programs are challenging to design and verify due to the exponential size of the superposition and certain physical limits on handling the superposition as enforced by quantum mechanics. One of the pioneer works that explores verification for quantum programs was proposed by Ying [Ying 2012] in which he extended Hoare logic [Hoare 1969] to reason about quantum programs. Subsequently, several verification frameworks based on this comprehensive foundation have been fruitfully established [Li and Ying 2017; Liu et al. 2019; Ying 2019]. These efforts are however less than ideal for the following reasons. First, these works employ a global reasoning approach over the entire quantum system, which can cause scalability issue due to the exponential size of the superposition. Second, these frameworks lacks of the direct support for classical variables and classical controls. Even though classical states can be modelled using quantum states [Feng and Ying 2021], such construct is tedious and unnecessarily complicates the reasoning of the classical states.

To address the scalability issue, our approach infuses separation logic (SL) in its core to support local reasoning for quantum computation. Fig. 1 presents our quantum frame rule QFrame that is identical to the classical frame rule [Reynolds 2002]. That said, the quantum separating conjunction $\circledast$ has a different interpretation within the context of quantum computation. The semantics of the predicate $P \circledast Q$ means that the quantum state can be tensor-factorized into two separate quantum states that satisfy P and Q respectively. For convenience, we name $\circledast$ the *tensor conjunction* hereafter. Informally, the tensor-factorization can be viewed as a generalization of the heap disjoint union in multi-dimensional spaces. Similar to the classical frame rule, the side condition of quantum frame rule QFrame requires that the program c does not modify any free variable in the frame predicate F.

Another feature of our framework is the direct supports for classical variables and classical controls. This is achieved by treating the computation as interactions between a classical computer C and a quantum computer Q, where C reads the measures made by Q and fetches classical inputs (in binary) to Q. In other words, quantum computation is viewed as a sub-routine of the classical program. As a result, we can express classical states using classical predicates and reuse the classical reasoning framework such as Hoare rules for variable assignment and conditional statements.

Our paper is structured as follows.

- § 2 We review necessary background knowledge about quantum computing.
- § 3 We provide an example to illustrate local reasoning in our framework.
- § 4 **Contribution 1**: We propose an intermediate quantum language in §4.1 and an assertion language for verification in §4.2. Our quantum language supports both classical statements and quantum statements including dynamic allocation and deallocation of qubits.
- § 5 **Contribution 2**: We propose an inference system for quantum reasoning in §5.1 and §5.2, and entailment system in §5.3.

§ 6 **Contribution 3**: We apply our framework to verify the n-qubit Hadamard transformation §6.1, Deutsch's algorithm §6.2 and also its generalization Deutsch–Jozsa's algorithm §6.3 [Deutsch and Jozsa 1992], and Grover's algorithm §6.4 [Grover 1996].
§7 **Contribution 4**: We define the operational semantics and use it to justify the soundness of our reasoning framework. We then discuss the main limitations of our framework.
§8 We discuss the related works and compare them with our framework.
§9 We conclude our work and discuss the future works.

2 PRELIMINARIES

We assume that our readers are familiar with basic concepts in linear algebra, e.g. Hilbert spaces, tensor products $\otimes$, complex conjugate $\overline{z}$. For a complex scalar $c \in \mathbb{C}$, we write $|c|$ for its *modulus* $\sqrt{c\overline{c}}$. Given a matrix $\mathcal{M}$, we write $\mathcal{M}^T$ for its transpose, and $\mathcal{M}^\dagger$ for its conjugate transpose. We may write the sum $\sum_{i=i_0}^{n}$ as $\sum_i$ if the range of i can be implied from the context.

2.1 Dirac Notation for Matrix Operators

A vector v is a column matrix, i.e. $v = \begin{pmatrix} c_1 \dots c_n \end{pmatrix}^T$. We use the *ket* $|\cdot\rangle$ to express vectors, and the *bra* $\langle\cdot|$ for their conjugate transposes. We let $\mathbb{B} \triangleq \{|0\rangle, |1\rangle\}$ be the standard binary basis where $|0\rangle = \begin{pmatrix} 1 & 0 \end{pmatrix}^T$ and $|1\rangle = \begin{pmatrix} 0 & 1 \end{pmatrix}^T$. Then we can express a vector $v = \begin{pmatrix} a_1 & a_2 \end{pmatrix}^T$ as $|v\rangle = a_1|0\rangle + a_2|1\rangle$ and $\langle v| = \overline{a_1}\langle 0| + \overline{a_2}\langle 1|$. We let $\mathbb{V}_{\mathcal{B}}^n$ be the Hilbert space spanning from the basis $\mathbb{B}^n$ where $\mathbb{B}^0 \triangleq \{1\}$ and $\mathbb{B}^{n+1} \triangleq \mathbb{B}^n \otimes \mathbb{B}$. Hereafter, we use the symbols $|e\rangle, |e'\rangle, |e''\rangle, \dots$ to represent basis vectors.

Let $\mathcal{V}_1, \mathcal{V}_2$ be vector spaces with the bases B_1, B_2 respectively. Then the tensor space $\mathcal{V}_1 \otimes \mathcal{V}_2$ is the vector space spanning from the basis $B_1 \otimes B_2$. We say a vector $|v\rangle \in \mathcal{V}_1 \otimes \mathcal{V}_2$ is *tensor-factorizable* if it can be expressed as the tensor product $|v\rangle = |v_1\rangle \otimes |v_2\rangle$ where $|v_1\rangle \in \mathcal{V}_1, |v_2\rangle \in \mathcal{V}_2$. As a result, local reasoning can be applied to reason about the individual states expressed by $|v_1\rangle$ and $|v_2\rangle$. On the other hand, not every vector in the tensor space $\mathcal{V}_1 \otimes \mathcal{V}_2$ is tensor-factorizable. For example, it can be shown through proof of contradiction that the vector $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$ representing the Bell state is not tensor-factorizable. This phenomenon is usually referred as the *quantum entanglement* in which the states of the quantum sub-systems cannot be expressed independently of each other.

Let $v_1 = \sum_i a_i|e_i\rangle \in \mathcal{V}_1, |v_2\rangle = \sum_j b_j|e'_j\rangle \in \mathcal{V}_2$. Their outer product is $|v_1\rangle\langle v_2| \triangleq \sum_{i,j} a_i\overline{b_j}|e_i\rangle\langle e'_j|$. If $\mathcal{V}_1 = \mathcal{V}_2$ then their inner product is the complex $\langle v_2|v_1\rangle \triangleq \sum_i a_i\overline{b_i}$. Note that $\langle\cdot|\cdot\rangle$ is conjugate symmetric, i.e. $\langle v_1|v_2\rangle = \overline{\langle v_2|v_1\rangle}$, and linear in the second argument, i.e. $\langle v|av_1 + bv_2\rangle = a\langle v|v_1\rangle + b\langle v|v_2\rangle$. Furthermore, $\langle v|v\rangle$ is real and positive for every $|v\rangle \neq 0$, and the *norm* of $|v\rangle$ is defined as $\|v\| \triangleq \sqrt{\langle v|v\rangle}$. We say $|v\rangle$ is *normalized* if $\|v\| = 1$.

Unless stated otherwise, we assume that our vectors belong to the Hilbert spaces $\mathbb{V}_{\mathcal{B}}^n$. For convenience, we may write the tensor product $|v_1\rangle \otimes |v_2\rangle$ as $|v_1\rangle|v_2\rangle$ or even $|v_1v_2\rangle$. Also, we write $|v^n\rangle$ to express the tensor $|v\rangle \dots |v\rangle$ in which $|v\rangle$ appears n times. If $n = 0$ then we let $|v^0\rangle = 1$.

2.2 A Minimalist's View of Quantum Computing

Complex Hilbert spaces are used to model quantum computation. In particular, the *pure state* of a **qubit** is expressed by a normalized vector $a_0|0\rangle + a_1|1\rangle$ such that $|a_0|^2 + |a_1|^2 = 1$. This indicates that the qubit is in the *superposition* where both classical states $|0\rangle$ and $|1\rangle$ exist simultaneously and the complex numbers a_0, a_1 are their amplitudes respectively. In general, the superposition of n qubits is expressed by the vector $|v\rangle = \sum_i a_i|e_i\rangle \in \mathbb{V}_{\mathcal{B}}^n$ such that $\|v\| = 1$, where each basis vector $|e_i\rangle \in \mathbb{B}^n$ corresponds to a classical state in the superposition.

Quantum gates/operators transform the quantum superpositions from one to another. A quantum operator $\mathcal{U} : \mathbb{V}^n_{\mathcal{B}} \mapsto \mathbb{V}^n_{\mathcal{B}}$ for n qubits is a $2^n \times 2^n$ *unitary matrix*, i.e. $\mathcal{U}\mathcal{U}^\dagger = \mathcal{U}^\dagger\mathcal{U} = \mathcal{I}_n$ where $\mathcal{I}_n$ denotes the $2^n \times 2^n$ identity matrix. We can express $\mathcal{U}$ as sum of outer products as follows:

$$\mathcal{U} \triangleq \sum_i |v_i\rangle\langle e_i| \text{ where } |e_i\rangle \in \mathbb{B}^n \text{ and } |v_i\rangle = \mathcal{U}(|e_i\rangle) \tag{1}$$

Let $|v\rangle = \sum_i a_i|e_i\rangle \in \mathbb{V}^n_{\mathcal{B}}$. Then $\mathcal{U}$ transforms $|v\rangle$ into:

$$\mathcal{U}(|v\rangle) \triangleq \mathcal{U}|v\rangle = \left(\sum_i |v_i\rangle\langle e_i|\right)\left(\sum_i a_i|e_i\rangle\right) = \sum_i a_i|v_i\rangle \tag{2}$$

Quantum operators can be combined together via the tensor product. Let $\mathcal{U}' = \sum_j |v'_j\rangle\langle e'_j|$ be a quantum operator for m qubits. Then $\mathcal{U} \otimes \mathcal{U}'$ is a quantum operator for $n + m$ qubits such that:

$$\mathcal{U} \otimes \mathcal{U}' \triangleq \left(\sum_i |v_i\rangle\langle e_i|\right) \otimes \left(\sum_j |v'_j\rangle\langle e'_j|\right) = \sum_{i,j} |v_i v'_j\rangle\langle e_i e'_j| \tag{3}$$

Quantum measurement manipulates the state of the qubits to obtain information about them, e.g. their classical states. The measurement of n qubits is defined using a set of linear mappings[1] $\{\mathcal{M}_i\}_{i=1}^k$ (not necessarily unitary) where each mapping $\mathcal{M}_i$ computes a measured outcome and $\sum_{i=1}^k \mathcal{M}_i^\dagger \mathcal{M}_i = \mathcal{I}_n$. Let $|v\rangle = \sum_j a_j|e_j\rangle \in \mathbb{V}^n_{\mathcal{B}}$ be the state space of n qubits to be measured. Then there is a probability of ρ_i that the measurement would yield the state $|v_i\rangle$ such that:

$$\rho_i = \langle v|\mathcal{M}_i^\dagger \mathcal{M}|v\rangle = \|\mathcal{M}_i(|v\rangle)\|^2 \quad \text{and} \quad |v_i\rangle = \frac{\mathcal{M}_i(|v\rangle)}{\sqrt{\rho_i}} = \frac{\mathcal{M}_i(|v\rangle)}{\|\mathcal{M}_i(|v\rangle)\|} \tag{4}$$

Note that the requirement $\sum_{i=1}^k \mathcal{M}_i^\dagger \mathcal{M}_i = \mathcal{I}_n$ ensures that the sum of all outcome probabilities is 1, i.e. $\sum_{i=1}^k \rho_i = 1$. Similar to quantum operators, measurements can be combined via the tensor product. Let $\{\mathcal{M}'_j\}_{j=1}^h$ be a measurement for m qubits. Then $\{\mathcal{M}_i\}_{i=1}^h \otimes \{\mathcal{M}'_j\}_{j=1}^k \triangleq \{\mathcal{M}_i \otimes \mathcal{M}'_j\}_{(i,j)=(1,1)}^{(k,h)}$ is a measurement for $n + m$ qubits. In particular, the measurement $\{\mathcal{M}_i\}_{i=1}^h \otimes \{\mathcal{I}_m\} = \{\mathcal{M}_i \otimes \mathcal{I}_m\}_{i=1}^h$ is used when we only measure a subset of qubits in the quantum system.

More often, a qubit is measured for its classical states $|0\rangle$ and $|1\rangle$. The corresponding measurement is $\{\mathcal{M}_0, \mathcal{M}_1\}$ where $\mathcal{M}_i = |i\rangle\langle i|$. This type of measurement is *projective* as $\mathcal{M}_i^2 = \mathcal{M}_i$ for $i \in \{0, 1\}$. The measurement would collapse a superposition $a_0|0\rangle + a_1|1\rangle$ into $|0\rangle$ with probability $|a_0|^2$, and into $|1\rangle$ with probability $|a_1|^2$. In general, we can measure n qubits via the projective measurement $\{|e_i\rangle\langle e_i|\}_i$ where each $|e_i\rangle$ is a basis vector in $\mathbb{B}^n$. This measurement would collapse a superposition $\sum_i a_i|e_i\rangle$ into each classical state $|e_i\rangle$ with probability $|a_i|^2$.

Interpretation of quantum computation. Quantum states are classified into *pure states* — the superposition of classical states — and *mixed states* — probabilistic ensembles of pure states (e.g. due to measurements). Vectors can express the former but not the latter. In general, the *density operator* is used to express mixed states. Let $S = \{(\rho_i, |s_i\rangle)\}_{i=1}^n$ be a mixed state where ρ_i is the probability associated with the pure state $|s_i\rangle$. The density operator of S is the following matrix:

$$\mathcal{S} = \sum_{i=1}^n \rho_i |s_i\rangle\langle s_i| \tag{5}$$

[1]The conventional definition of measurement starts with $\{\mathcal{F}_i\}_{i=1}^k$ where $\mathcal{F}_i$ is self-adjoint positive semidefinite and $\sum_{i=1}^k \mathcal{F}_i = \mathcal{I}_n$. The former condition is equivalent to $\mathcal{F}_i = \mathcal{M}_i^\dagger \mathcal{M}_i$ for some $\mathcal{M}_i$, which then reduces to our definition of measurement.

$$
\begin{array}{ll}
1 & \{\ q^*[0,99] \mapsto |0^{100}\rangle\ \} \\
2 & \mathcal{H}(q^*[0]); \mathcal{H}(q^*[1]); \mathcal{H}(q^*[99]); \\
3 & \{\ q^*[0] \mapsto |+\rangle \circledast q^*[1] \mapsto |+\rangle \circledast \mathbf{true}\ \}
\end{array}
$$

Fig. 2. A quantum program with Hadamard operator $\mathcal{H}$.

$$
\begin{array}{ll}
1 & \{\ q^*[0,99] \mapsto |0^{100}\rangle\ \} \\
2 & \{\ q^*[0] \mapsto |0\rangle \circledast q^*[1] \mapsto |0\rangle \circledast q^*[99] \mapsto |0\rangle \circledast \mathbf{true}\ \} \\
3 & \Longleftrightarrow \{\ q^*[0] \mapsto |0\rangle\ \}\ \mathcal{H}(q^*[0]);\ \{\ q^*[0] \mapsto |+\rangle\ \} \\
4 & \Longleftrightarrow \{\ q^*[1] \mapsto |0\rangle\ \}\ \mathcal{H}(q^*[1]);\ \{\ q^*[1] \mapsto |+\rangle\ \} \\
5 & \Longleftrightarrow \{\ q^*[99] \mapsto |0\rangle\ \}\ \mathcal{H}(q^*[99]);\ \{\ q^*[99] \mapsto |+\rangle\ \} \\
6 & \{\ q^*[0] \mapsto |+\rangle \circledast q^*[1] \mapsto |+\rangle \circledast q^*[99] \mapsto |+\rangle \circledast \mathbf{true}\ \} \\
7 & \{\ q^*[0] \mapsto |+\rangle \circledast q^*[1] \mapsto |+\rangle \circledast \mathbf{true}\ \}
\end{array}
$$

Fig. 3. Proof of our quantum program.

In our framework, the mixed state is split into multiple pure states, each is associated with the respective probability (c.f. §7.1). As a result, it is sufficient to express quantum states using vectors which can be tensor-factorized for local reasoning. Furthermore, the vector representation is more state-space efficient and human-readable than the density operator. When a measurement occurs, the pure state goes to one of the outcomes non-deterministically, and the associated probability is updated accordingly. Our interpretation of quantum computation is similar to the probability based transition system for quantum computation [Selinger and Valiron 2005], which is consistent with the density operator model [Selinger and Valiron 2008].

3 AN ILLUSTRATIVE EXAMPLE

We use the example in Fig 2 to demonstrate the strength of the quantum frame rule QFrame (Fig. 1) for local reasoning. The precondition specifies that initially the quantum system contains 100 qubits stored in the array segment $q^*[0,99]$ from index 0 to 99. Besides, their combined state is $|0^{100}\rangle$ which is the tensor of 100 vectors $|0\rangle$. Equivalently, each qubit in the system has the classical state $|0\rangle$. The program applies the quantum Hadamard operator $\mathcal{H}$ to the three qubits $\{q^*[0], q^*[1], q^*[99]\}$ sequentially. Here the operator $\mathcal{H}$ transforms the classical state $|0\rangle$ into the superposition $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and the classical state $|1\rangle$ into the superposition $\frac{|0\rangle-|1\rangle}{\sqrt{2}}$. Thus its specs can be expressed by the following matrix written in Dirac notation:

$$\mathcal{H} \triangleq |+\rangle\langle 0| + |-\rangle\langle 1| \quad \text{where} \quad |+\rangle \triangleq \frac{|0\rangle + |1\rangle}{\sqrt{2}} \text{ and } |-\rangle \triangleq \frac{|0\rangle - |1\rangle}{\sqrt{2}} \tag{6}$$

In the postcondition, we would like to prove that the states of $q^*[0]$ and $q^*[1]$ are both $|+\rangle$ and are separable. Since the semantics for $\circledast$ in our framework is classical rather than intuitionistic, the extra predicate **true** in the postcondition is necessary to avoid 'dropping resources on the floor', i.e. we forbid $P \circledast Q$ to imply P. We utilize the following proof notations to highlight the use of the frame rule QFrame:

$$\Longleftrightarrow \{\ P\ \}\ c\ \{\ Q\ \} \qquad \text{or equivalently} \qquad \begin{array}{l} \Longrightarrow \{\ P\ \} \\ \quad c \\ \Longleftarrow \{\ Q\ \} \end{array}$$

$$
\begin{array}{llll}
(\mathit{Expr}) & e & ::= & \mathbb{Z} \mid x, y, \ldots \mid f_m(e, \ldots, e) \\
(\mathit{Bexpr}) & b & ::= & \textbf{true} \mid P_m(e, \ldots, e) \mid b \&\& b \mid !b \\
(\mathit{Aexpr}) & e^* & ::= & [] \mid q^*[e, e], \ldots \mid e^* e^* \\
(\mathit{Stmt}) & c & ::= & \textbf{skip} \mid x := e \mid \textbf{if } b \textbf{ do } c \textbf{ else } c \mid \textbf{while } b \textbf{ do } c \mid c\ ;c \mid \\
 & & & q^* := \textbf{qbit}(e) \mid \mathcal{G}(e^*) \mid x := \textbf{measure}(e^*) \mid \textbf{dispose}(q^*)
\end{array}
$$

Fig. 4. Syntax of quantum programs.

Both notations indicate that $\{P\}c\{Q\}$ is the local proof for c and the latter is useful for long proofs. The frame predicate F can be deduced from the assertions before P or after Q.

Fig. 3 presents the complete proof for our program. Since the postcondition only assesses the first two qubits, the key idea of our approach is to reason about the states of these two qubits locally rather than over the global state of 100 qubits as in the existing approaches (e.g. [Ying 2012]). Starting from the precondition, the combined state is split into the states of individual qubits via the following entailment rule (c.f. §5.3):

$$
\frac{|v\rangle \in \mathbb{V}_{\mathcal{B}}^{|e^*|} \qquad |v'\rangle \in \mathbb{V}_{\mathcal{B}}^{|e'^*|}}{e^* e'^* \mapsto |v\rangle \otimes |v'\rangle \ \dashv\vdash \ \left(e^* \mapsto \frac{|v\rangle}{\||v\rangle\|}\right) \circledast \left(e'^* \mapsto \frac{|v'\rangle}{\||v'\rangle\|}\right)} \ {}^{\circledast}\!\mapsto \tag{7}
$$

Here we write $\Phi_1 \vdash \Phi_2$ to indicate that Φ_1 proves Φ_2, and if Φ_2 also proves Φ_1 then we write $\Phi_1 \dashv\vdash \Phi_2$. We use the mapping $e^* \mapsto |v\rangle$ to express that the the qubits in e^* has the state $|v\rangle$. Also, $e^* e'^*$ represents the concatenation of the two array segments e^* and e'^*. The above rule allows us to separate and combine the states of e^* and e'^*, given that the combined state of $e^* e'^*$ can be tensor-factorized into the states of e^* and e^*. On the right hand side, the states of e^* and e'^* are divided by their norms for normalization.

As our program only modifies the three qubits $q^*[0]$, $q^*[1]$ and $q^*[99]$, other qubits can be abstracted with the predicate **true** by applying the following rules (c.f. §5.3):

$$
\frac{}{P \vdash \textbf{true}}\ {}^{\top} \qquad \frac{}{\overline{P} \circledast \overline{Q} \dashv\vdash \overline{P} \wedge \overline{Q}}\ {}^{\overline{\circledast}} \qquad \frac{P \vdash Q}{R \circledast P \vdash R \circledast Q}\ {}^{\circledast\vdash} \tag{8}
$$

We $\overline{\text{overline}}$ a predicate $\overline{P}$ to indicate that P is *pure*, i.e. it does not contain symbols related to the quantum states such as the tensor conjunction $\circledast$ and the quantum mapping $\mapsto$. Starting from the precondition, we use the fact that $\circledast$ is commutative and associative to separate the three qubits $q^*[0]$, $q^*[1]$ and $q^*[99]$ from the rest. We then apply $\top$–rule to turn the irrelevant qubits into **true**, $\overline{\circledast}$–rule to conjunct the **true** predicates together, and finally ${}^{\circledast}\!\vdash$–rule to combine **true** with the three qubits in the program. In lines 3–5, we apply the frame rule QFrame to reason about the state transformations of the individual qubits in $\{q^*[0], q^*[1], q^*[99]\}$. Finally, the postcondition is obtained by abstracting $q^*[99]$ with the predicate **true**.

4 LANGUAGES FOR QUANTUM PROGRAMMING AND REASONING

We first explain our lightweight quantum language in §4.1 then the assertion language for reasoning in §4.2. Notation-wise, we use the characters $\{x, y, \ldots\}$ for classical variables, the characters $\{q^*, p^*, \ldots\}$ for quantum variables which are essentially arrays of qubits, and the characters $\{\mathcal{H}, \mathcal{X}, \ldots\}$ for quantum gates/operators. We also assume a pre-defined set of arithmetic functions f_m (e.g. $+, \times, \ldots$) and a pre-defined set of Boolean predicates P_m (e.g. $\leq, =, \ldots$).

$$
\begin{array}{llll}
(\textit{Pure expression}) & e_p & ::= & \mathbb{C} \mid x, y, \ldots \mid f_m(e_p, \ldots, e_p) \\
(\textit{Pure formula}) & \overline{P} & ::= & |e^*| \leq e_p \mid P_m(e_p, \ldots, e_p) \mid \overline{P} \wedge \overline{P} \mid \neg \overline{P} \mid \exists x.\ \overline{P} \\
(\textit{State expression}) & |s\rangle & ::= & e_p \mid |e_p\rangle \mid |s\rangle + |s\rangle \mid |s\rangle \otimes |s\rangle \\
(\textit{Quantum formula}) & F & ::= & \overline{P} \mid e^* \mapsto |s\rangle \mid e_p \cdot F \mid F \circledast F \mid F \vee F \mid F \wedge F \mid \exists x.\ F \mid \forall x.\ F
\end{array}
$$

Fig. 5. The assertion language.

4.1 Quantum Programming Language

Fig. 4 presents the description of our quantum language. We use standard constructs for arithmetic expression e and Boolean expression b. The construct e^* represents an array of qubits which can be [] — the empty array — or the sub-array $q^*[e_1, e_2]$ — where e_1, e_2 are the starting and ending indices inclusively — or $e_1^* e_2^*$ — the concatenation of the two array segments e_1^* and e_2^*. We also assume that our arrays except [] are non-empty by default.

The construct c captures the syntax of our quantum programs. It supports standard statements such as skip, assignment for classical variables, if-else and while. As a result, classical programs can be expressed directly by our language. For quantum computation, our language supports the following statements:

+ **Allocation**: $q^* := \mathbf{qbit}(n)$ allocates n new qubits to the fresh array segment $q^*[0, n-1]$ and initiates the state of each qubit to $|0\rangle$. For example, the statement $q^* := \mathbf{qbit}(3)$ allocates three qubits to $q^*[0, 2]$ and sets the initial state of each qubit to $|0\rangle$.

+ **Transformation**: $\mathcal{G}(e^*)$ applies the quantum operator $\mathcal{G}$ to transform the state of the qubits in the array segment e^*. For the transformation to be feasible, the dimension of the operator $\mathcal{G}$ needs to be compatible with the state dimension of e^*.

+ **Measurement**: $v := \mathbf{measure}(e^*)$ measures the state of the qubits in the array segment e^* and assigns the result to the classical variable v in decimal base. For example, measuring the quantum state $(|00\rangle + |11\rangle)/\sqrt{2}$ would either return the classical value 0 (i.e. $|00\rangle$) or 3 (i.e. $|11\rangle$) with equal probability of 0.5. After the measurement, the qubits in e^* become separable from the remaining qubits in the system.

+ **Deallocation**: $\mathbf{dispose}(q^*)$ removes all qubits in the array q^*. For the deallocation to be safe from interference, we require that the disposing qubits are separable from the remaining quantum system. Since our framework focuses on the reasoning, the above condition is enforced at the semantics layer by ensuring that quantum states can be tensor-factorizable before the disposal. In practice, it can be done (incompletely) through static type systems (e.g. [Bichsel et al. 2020]).

4.2 Assertion Language

Fig. 5 presents our assertion language for quantum reasoning. The classical states are expressed by *pure predicate* $\overline{P}$ which is a first-order formula over pure expressions e_p — arithmetic expression with complex constants. Note that $\overline{P}$ can contain the length predicate $|e^*| \leq e_p$ in which $|e^*|$ represents the length of the array segment e^*. This length predicate is useful to keep track of the number of qubits in the arrays during the allocations and deallocations.

Quantum states are expressed by the vector $|s\rangle$ in Dirac notation to improve readability. In detail, $|s\rangle$ can be a complex scalar e_p, a basis vector $|e_p\rangle \in \mathbb{B}$ where e_p is evaluated to either 0 or 1, the vector sum $|s_1\rangle + |s_2\rangle$, or the tensor product $|s_1\rangle \otimes |s_2\rangle$. Note that the scalar multiplication can be interpreted using the tensor product, i.e. $e_p|s\rangle$ is equivalent to $e_p \otimes |s\rangle$.

The quantum predicate F is a first-order formula over both classical and quantum states. Classical states are expressed by pure predicates $\overline{P}$. Furthermore, F contains the following predicates to express quantum states and probabilities:

$$\frac{\{P\}c\{Q\} \qquad \mathsf{free}(F) \cap \mathsf{mod}(c) = \emptyset}{\{P \circledast F\}c\{Q \circledast F\}}\ \mathsf{QFrame}$$

$$\frac{\{P\}c\{Q\} \qquad \mathsf{free}(e) \cap \mathsf{mod}(c) = \emptyset}{\{e \cdot P\}c\{e \cdot Q\}}\ \mathsf{PFrame} \qquad \frac{\{P\}c\{Q\} \qquad \mathsf{free}(\overline{F}) \cap \mathsf{mod}(c) = \emptyset}{\{P \wedge \overline{F}\}c\{Q \wedge \overline{F}\}}\ \overline{\mathsf{Frame}}$$

Fig. 6. The quantum frame rule QFrame and its derived rules

(1) **Quantum mapping** $e^* \mapsto |s\rangle$ which specifies that the vector $|s\rangle$ represents the combined quantum state of the qubits in the array e^*. This notation is generalized from the SL mapping $x \mapsto v$ which indicates the value v is stored at the memory address x. This generalization is necessary to express the entangled states of multiple qubits.
(2) **Fractional conjunction** $e_p \cdot F$ which specifies that the program state satisfies F and is associated with a probability of e_p. We use this predicate to capture the probabilities of the measurement outcomes. The notation $e_p \cdot F$ is inspired from the predicate for disjoint fractional permissions [Le and Hobor 2018], where e_p represents the permission associated with the resource specified by F.
(3) **Tensor conjunction** $F_1 \circledast F_2$ which specifies that the quantum state can be tensor-factorized into two quantum states satisfying F_1 and F_2 respectively. Semantics-wise, this indicates that the two quantum states are separable and thus can be reasoned locally using the frame rule.

Notations. We adopt the following notations hereafter. We write $|\mathbf{emp}\rangle$ for the empty quantum heap that maps the empty array to the scalar 1, i.e. $[] \mapsto 1$. We write $\circledast_{i=m}^{n} F(i)$ to express the predicate $F(m) \circledast F(m+1) \ldots \circledast F(n)$. If $m > n$ then we let $\circledast_{i=m}^{n} F(i)$ equal to 1.

5 INFERENCE FRAMEWORK

We explain the key inference rules in our framework, starting from the rule QFrame and its variants in §5.1, followed by the core quantum rules in §5.2, to the entailment rules in §5.3.

5.1 Quantum Frame Rule

Fig. 6 features the quantum frame rule QFrame and its derived rules PFrame and $\overline{\mathsf{Frame}}$. Similar to its SL counterpart, QFrame allows us to attach the the frame predicate F into the pre/post-conditions of the local proof $\{P\}c\{Q\}$. The side condition $\mathsf{free}(F) \cap \mathsf{mod}(c) = \emptyset$ specifies that the program c does not modify any free variables in F. Here $\mathsf{free}(F)$ contains free classical variables (e.g. $x, y, \ldots$) and quantum variables (e.g. $q^*, p^*, \ldots$) in F. On the other hand, $\mathsf{mod}(c)$ contains classical variables modified by c and quantum variables in the allocation/deallocation statements of c, i.e.:

$$\begin{gathered}\mathsf{mod}(\mathcal{G}(e^*)) \triangleq \emptyset \qquad \mathsf{mod}(v := e) = \mathsf{mod}(v := \mathbf{measure}(e^*)) \triangleq \{v\} \\ \mathsf{mod}(q^* := \mathbf{qbit}[e](e')) = \mathsf{mod}(\mathbf{dispose}(q^*)) \triangleq \{q^*\}\end{gathered} \tag{9}$$

The two rules PFrame and $\overline{\mathsf{Frame}}$ are derived from QFrame (c.f. §7.4). The *probabilistic frame rule* PFrame reduces the proof $\{e \cdot P\}c\{e \cdot Q\}$ to $\{P\}c\{Q\}$. The *pure frame rule* $\overline{\mathsf{QFrame}}$ is applicable when the frame predicate $\overline{F}$ is pure and so $\circledast$ can be replaced by $\wedge$. Both derived rules require that the program c does not modify any free variable in the frame components, i.e. the probability expression e in PFrame and the pure predicate $\overline{F}$ in $\overline{\mathsf{Frame}}$.

$$\frac{}{\{|\mathbf{emp}\rangle \wedge e = n > 0\} q^* := \mathbf{qbit}(e) \{q^*[0, n-1] \mapsto |0^n\rangle \wedge |q^*| = n\}} \text{ Qubit}$$

$$\frac{}{\{q^*[0, n-1] \mapsto |v\rangle \wedge |q^*| = n\} \mathbf{dispose}(q^*) \{|\mathbf{emp}\rangle\}} \text{ Dis}$$

$$\frac{\mathcal{G} : \mathbb{V}_{\mathcal{B}}^{|e^*|} \mapsto \mathbb{V}_{\mathcal{B}}^{|e^*|} \qquad |e_i\rangle \in \mathbb{B}^{|e^*|}}{\{e^* e'^* \mapsto \sum_{i,j} a_{i,j} |e_i\rangle |e'_j\rangle\} \mathcal{G}(e^*) \{e^* e'^* \mapsto \sum_{i,j} a_{i,j} \mathcal{G}(|e_i\rangle) |e'_j\rangle\}} \text{ Trans}$$

$$\frac{\begin{array}{c} |e_i\rangle \in \mathbb{B}^{|e^*|}, |e'_j\rangle \in \mathbb{B}^{|e'^*|} \qquad v \notin \text{free}(\Psi) \qquad \rho_i \triangleq \sum_j |a_{i,j}|^2 \\ \Psi \triangleq e^* e'^* \mapsto \sum_{i,j} a_{i,j} |e_i\rangle |e'_j\rangle \qquad \Psi_i \triangleq e^* \mapsto |e_i\rangle \circledast e'^* \mapsto \sum_j \frac{a_{i,j}}{\sqrt{\rho_i}} |e'_j\rangle \end{array}}{\{\Psi \wedge (\bigwedge_i \overline{\Phi_i}[v/e_i])\} v := \mathbf{measure}(e^*) \{\bigvee_i (\rho_i \cdot \Psi_i \wedge \overline{\Phi_i})\}} \text{ Ms}$$

(a) Quantum inference rules

$$\frac{}{\{P\}\mathbf{skip}\{P\}} \text{ Sk} \qquad \frac{}{\{P[v/e]\} v := e \{P\}} \text{ Assign} \qquad \frac{\{P\} c_1 \{R\} \quad \{R\} c_2 \{Q\}}{\{P\} c_1; c_2 \{Q\}} \text{ Seq}$$

$$\frac{\{P \wedge b\} c_1 \{Q\} \quad \{P \wedge \neg b\} c_2 \{Q\}}{\{P\} \mathbf{if}\ b\ \mathbf{do}\ c_1\ \mathbf{else}\ c_2 \{Q\}} \text{ If} \qquad \frac{\{I \wedge b\} c \{I\}}{\{I\} \mathbf{while}\ b\ \mathbf{do}\ c \{I \wedge \neg b\}} \text{ While}$$

$$\frac{}{\{\mathbf{false}\} c \{P\}} \text{ Bot} \qquad \frac{P' \vdash P \quad \{P\} c \{Q\} \quad Q \vdash Q'}{\{P'\} c \{Q'\}} \text{ Cons}$$

$$\frac{\{P\} c \{Q\} \quad \{P'\} c \{Q'\}}{\{P \vee P'\} c \{Q \vee Q'\}} \text{ Disj} \qquad \frac{\{P\} c \{Q\} \quad \{P'\} c \{Q'\}}{\{P \wedge P'\} c \{Q \wedge Q'\}} \text{ Conj}$$

(b) Classical Hoare rules

Fig. 7. Inference system for reasoning

5.2 Quantum Core Rules

Fig. 7 presents the core reasoning rules in our framework. The rules for classical statements in Fig. 7b are standard and self-explanatory. We now explain the quantum rules in Fig. 7a.

[Qubit] reasons about the allocation $q^* := \mathbf{qbit}(e)$. The precondition of the Hoare triple specifies the empty quantum heap $|\mathbf{emp}\rangle$ and the pure expression e is evaluated to some integer $n > 0$. The postcondition specifies that n new qubits are allocated to q^* and their states are initiated to $|0\rangle$. Besides, the length predicate $|q^*| = n$ allows us to keep track of the number of qubits in q^*.

[Dis] reasons about the deallocation $\mathbf{dispose}(q^*)$. In the Hoare triple, the precondition captures the combined state of the qubits in q^*. As the qubits in q^* are disposed, the postcondition becomes the empty quantum heap $|\mathbf{emp}\rangle$.

[Trans] reasons about the quantum transformation $\mathcal{G}(e^*)$, i.e. the application of the quantum operator $\mathcal{G}$ to the qubits in e^*. The precondition of the Hoare triple specifies the combined state of $e^* e'^*$. This includes the cases where the state of e^* is entangled with the state of e'^* and therefore their states must be expressed together. In the postcondition, the combined state of $e^* e'^*$ is transformed by applying $\mathcal{G}$ to the basis vectors of e^*.

[Ms] reasons about the measurement $v := \mathbf{measure}(e^*)$ where the qubits in e^* are measured and the outcome is assigned to v in decimal base. The precondition of the Hoare triple is the conjunctive

predicate $\Psi \wedge (\bigwedge_i \overline{\Phi}_i[v/e_i])$ where Ψ is the mapsto predicate that expresses the combined state of $e^*e'^*$ and $\overline{\Phi}_i[v/e_i]$ are pure predicates in which v is replaced by the measurement outcomes e_i. Similar to the rule Trans, the combined state $e^*e'^*$ in Ψ is necessary when the states of e^* and e'^* are entangled. The side condition $v \notin \text{free}(\Psi)$ — that v is not a free variable in Ψ — helps to prevent v from affecting the evaluation of the quantum state in Ψ after the measurement. Our postcondition is the disjunctive predicate $\bigvee_i(\rho_i \cdot \Psi_i \wedge \overline{\Phi_i})$ which represents the probabilistic ensemble of the quantum states after the measurement. Each predicate $\rho_i \cdot \Psi_i \wedge \overline{\Phi_i}$ specifies an outcome e_i where ρ_i is the respective probability, Ψ_i expresses the result quantum state where the state of e^* is $|e_i\rangle$ and is separable from the state of e'^*, and $\overline{\Phi(i)}$ expresses the classical state where $v = e_i$.

Example of Ms **rule**. Assume our 2-qubit system $\{q^*[0], q^*[1]\}$ is in the entangled Bell's state $\frac{|00\rangle+|11\rangle}{\sqrt{2}}$ and the first qubit is measured as $v := \textbf{measure}(q^*[0])$. We let the precondition be the following conjunctive predicate:

$$\left(q^*[0,1] \mapsto \frac{1}{\sqrt{2}}|00\rangle + \frac{1}{\sqrt{2}}|11\rangle\right) \wedge (0 = 0 \wedge 1 = 1) \tag{10}$$

According to the measurement rule, the probabilities associated with the measured states $|0\rangle$ and $|1\rangle$ are computed to be $\rho_1 = \rho_2 = \frac{1}{2}$. Our postcondition is the disjunctive predicate $F_1 \vee F_2$ where each F_i corresponds to an outcome of the measurement, i.e.:

$$F_1 \triangleq \tfrac{1}{2} \cdot \left(q^*[0] \mapsto |0\rangle \circledast q^*[1] \mapsto |0\rangle\right) \wedge v = 0 \qquad F_2 \triangleq \tfrac{1}{2} \cdot \left(q^*[0] \mapsto |1\rangle \circledast q^*[1] \mapsto |1\rangle\right) \wedge v = 1 \tag{11}$$

The predicates F_1 and F_2 capture the states of $q^*[0], q^*[1]$ and the value of v after the measurement. In both cases, the states of the two qubits become separable.

5.3 Entailment Reasoning

Fig. 8 presents our core entailment rules to manipulate the quantum predicates. We use the notation $P \vdash Q$ to indicate that P proves Q, and the notation $P \dashv\vdash Q$ if the other direction also holds. We now briefly explain these rules.

Fig. 8a consists of proof rules for $\circledast$. The three rules $\{\circledast^E, \circledast^C, \circledast^A\}$ indicate that the empty quantum heap $|\textbf{emp}\rangle$ is the identity element and $\circledast$ is commutative and associative. The $\circledast^{\vdash}$–rule allows us to attach the predicate R into $P \vdash Q$ as $P \circledast R \vdash Q \circledast R$. The $\overline{\circledast}$–rule replaces $\circledast$ with $\wedge$ if both predicates are pure. The $\overline{\circledast^{\wedge}}$–rule replaces $\overline{P} \wedge (Q \circledast R)$ with $(\overline{P} \wedge Q) \circledast R$ and vice versa. The two rules $\{\circledast^{\wedge}, \circledast^{\vee}\}$ help to distribute $\circledast$ into conjunctive and disjunctive predicates.

The two rules in Fig. 8b are for the mapsto predicate. The $\overset{\circledast}{\mapsto}$–rule allows us to split/join the quantum states. In detail, if the combined state of $e^*e'^*$ can be tensor-factorized into $|v\rangle \otimes |v'\rangle$ then the states of e^* and e'^* are separable and are expressed by the normalized vectors $|v\rangle/\|v\|$ and $|v'\rangle/\|v'\|$ respectively. The other $\overset{C}{\mapsto}$–rule helps to rearrange the positions of the qubits. We can use it to infer the combined state of e'^*e^* from the combined state of $e^*e'^*$ — i.e. the positions of e^* and e'^* are swapped — by swapping the basis vectors of e^* and e'^* in the state vector of $e^*e'^*$.

The rules in Fig. 8c are for probability reasoning. The $\mathbb{P}^{\vdash}$–rule reduces the proof $p \cdot P \vdash p \cdot Q$ to $P \vdash Q$. The $\mathbb{P}^I$–rule rewrite $1 \cdot P$ with P and vice versa. The $\mathbb{P}^{\perp}$–rule helps to eliminate the cases where the probability expression p is invalid. The $\overline{\mathbb{P}}$–rule rewrites $p \cdot (\overline{P} \wedge Q)$ with $\overline{P} \wedge p \cdot Q$ and vice versa. The $\mathbb{P}^J$–rule collapses nested probabilities by multiplying them together. The $\mathbb{P}^{\circledast}$–rule distributes the probabilities over $\circledast$. The two rules $\{\mathbb{P}^{\wedge}, \mathbb{P}^{\vee}\}$ distribute the probability over $\{\wedge, \vee\}$.

Fig. 8d highlights several useful classical rules. The $\top$–rule asserts that any P proves **true** whereas the $\perp$–rule asserts that **false** proves any P. The $\overset{S}{=}$–rule substitutes the term/expression t' for t in P — given that they are semantically equivalent with respect to the context of P.

$$\frac{}{P \circledast |\mathbf{emp}\rangle \dashv\vdash P}\ \circledast^E \qquad \frac{}{P \circledast Q \vdash Q \circledast P}\ \circledast^C \qquad \frac{}{(P \circledast Q) \circledast R \dashv\vdash P \circledast (Q \circledast R)}\ \circledast^A$$

$$\frac{P \vdash Q}{P \circledast R \vdash Q \circledast R}\ \circledast^{\vdash} \qquad \frac{}{\overline{P} \circledast \overline{Q} \dashv\vdash \overline{P} \wedge \overline{Q}}\ \overline{\circledast} \qquad \frac{}{\overline{P} \wedge (Q \circledast R) \dashv\vdash (\overline{P} \wedge Q) \circledast R}\ \overline{\circledast^{\wedge}}$$

$$\frac{}{P \circledast (Q \wedge R) \vdash (P \circledast Q) \wedge (P \circledast R)}\ \circledast^{\wedge} \qquad \frac{}{P \circledast (Q \vee R) \dashv\vdash (P \circledast Q) \vee (P \circledast R)}\ \circledast^{\vee}$$

(a) Rules for the tensor conjunction

$$\frac{|v\rangle \in \mathbb{V}_{\mathcal{B}}^{|e^*|} \qquad |v'\rangle \in \mathbb{V}_{\mathcal{B}}^{|e'^*|}}{e^*e'^* \mapsto |v\rangle \otimes |v'\rangle \dashv\vdash e^* \mapsto \frac{|v\rangle}{\||v\rangle\|} \circledast e'^* \mapsto \frac{|v'\rangle}{\||v'\rangle\|}}\ \overset{\circledast}{\mapsto}$$

$$\frac{|e_i\rangle \in \mathbb{V}_{\mathcal{B}}^{|e^*|} \qquad |e'_j\rangle \in \mathbb{V}_{\mathcal{B}}^{|e'^*|}}{e^*e'^* \mapsto \sum_{i,j} a_{i,j}|e_i\rangle|e'_j\rangle \vdash e'^*e^* \mapsto \sum_{i,j} a_{i,j}|e'_j\rangle|e_i\rangle}\ \overset{C}{\mapsto}$$

(b) Rules for the quantum mapping

$$\frac{P \vdash Q}{p \cdot P \vdash p \cdot Q}\ \mathbb{P}^{\vdash} \qquad \frac{}{1 \cdot P \dashv\vdash P}\ \mathbb{P}^{I} \qquad \frac{}{(p \le 0 \vee p > 1) \wedge p \cdot P \vdash \mathbf{false}}\ \mathbb{P}^{\perp}$$

$$\frac{}{p \cdot (\overline{P} \wedge Q) \dashv\vdash \overline{P} \wedge p \cdot Q}\ \overline{\mathbb{P}} \qquad \frac{}{p \cdot (p' \cdot P) \dashv\vdash (pp') \cdot P \wedge 0 < p, p' \le 1}\ \mathbb{P}^{J}$$

$$\frac{}{p \cdot (P \wedge Q) \dashv\vdash (p \cdot P) \wedge (p \cdot Q)}\ \mathbb{P}^{\wedge} \qquad \frac{}{p \cdot (P \vee Q) \dashv\vdash (p \cdot P) \vee (p \cdot Q)}\ \mathbb{P}^{\vee}$$

(c) Rules for the fractional conjunction

$$\frac{}{P \vdash \mathbf{true}}\ \top \qquad \frac{}{\mathbf{false} \vdash P}\ \perp \qquad \frac{\models P \Rightarrow t = t'}{P \vdash P[t/t']}\ \overset{S}{=}$$

(d) Useful rules in classical logic

Fig. 8. Inference rules for rewriting predicates

Reasoning about the probabilities of pure predicates. We introduce the following notation to reason about the probabilities of pure predicates, particularly the probabilities of the measurements. Let $\overline{Q}$ be a pure formula. We use the notation $\mathbb{P}(\overline{Q})$ to express the probability that $\overline{Q}$ holds. Also, we let F be a predicate in the closure of $\{\mapsto, \wedge, \vee, \circledast\}$, i.e.:

$$F ::= e^* \mapsto |v\rangle \mid F \circledast F \mid F \wedge F \mid F \vee F \tag{12}$$

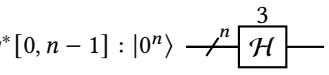
Here F represents the pure states with probability 1 and thus $p \cdot F$ represents the pure states with probability p. Ideally, we would like to interpret $\mathbb{P}(\overline{Q})$ using the following entailment rule:

$$\frac{P \wedge \overline{Q} \vdash p \cdot F \wedge p > 0}{P \vdash \mathbb{P}(\overline{Q}) = p}\ \mathbb{P} \qquad \text{(first attempt)} \tag{13}$$

That is, if the quantum pure states in which $\overline{Q}$ holds have probability $p > 0$ then $\mathbb{P}(\overline{Q}) = p$. However, this is semantically incorrect since $\overline{Q}$ can hold in multiple quantum pure states and thus the probability should be greater than p in general. Changing '=' to '≥' — i.e. $\mathbb{P}(\overline{Q}) \ge p$ — fixes

```
1   i := 0;
2   while i < n do
3     H(q*[i]);
4     i := i + 1;
```

(a) The operator's code



(b) The operator's circuit design

Fig. 9. Hadamard operator for n qubits $\mathcal{H}_n$

the above issue but also weakens the reasoning quite significantly. To compromise, one can use a slightly stronger rule which infers that $\mathbb{P}(\overline{Q})$ is a multiple of p:

$$\frac{P \wedge \overline{Q} \vdash p \cdot F \wedge p > 0}{\text{exist an integer } n \geq 1.\ P \vdash \mathbb{P}(\overline{Q}) = np \leq 1}\ \mathbb{P} \tag{14}$$

In particular, if $p > \frac{1}{2}$ then $n = 1$ and thus $\mathbb{P}(\overline{Q}) = p$. Also the above rule automatically gives us the lower bound $\mathbb{P}(\overline{Q}) \geq p$. As a technicality, the value p needs to be unique for the predicate $\mathbb{P}(\overline{Q}) = p$ to be well-defined. This can be justified by the following lemma:

Lemma 5.1. *Suppose the two predicates $p \cdot F$ and $p' \cdot F'$ where $p, p' \in (0, 1]$ are satisfied by the same program state. Then $p = p'$.*

For example, the following formula captures the measurement of the state $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$:

$$P \triangleq \left(x = 0 \wedge \frac{1}{2} \cdot q^*[0] \mapsto |0\rangle\right) \vee \left(x = 1 \wedge \frac{1}{2} \cdot q^*[0] \mapsto |1\rangle\right) \tag{15}$$

As P entails the implication $x = 1 \longrightarrow \frac{1}{2} \cdot q^*[0] \mapsto |0\rangle$, it follows that $\mathbb{P}(x = 1) = \frac{n}{2}$ for some integer $n \geq 1$ and so $\mathbb{P}(x = 1) \geq \frac{1}{2}$.

6 CASE STUDIES

We apply our framework to verify the following programs: n-qubit Hadamard transformation §6.1, Deutsch's algorithm §6.2 together with its generalization Deutsch–Jozsa's algorithm §6.3 [Deutsch and Jozsa 1992], and finally Grover's algorithm [Grover 1996] §6.4.

The two well-known Deutsch–Jozsa's algorithm and Grover's algorithm showcase the superior computation power of quantum computing over classical computing. Despite being small programs, they require sophisticated mathematics to encode the superposition and need significant efforts to be formally verified. In particular, it takes more than 3000 line of code [Liu et al. 2019, §5.2] for the mechanization of the Grover's algorithm in Isabelle/HOL using Ying's framework [Ying 2012].

6.1 Hadamard Transformation for Multiple Qbits

Fig. 9 presents the implementation of the Hadamard operator $\mathcal{H}_n$ for n qubits using the Hadamard operators $\mathcal{H}$ for a single qubit. The code of $\mathcal{H}_n$ in Fig. 9a is a loop where $\mathcal{H}$ is applied to each qubit in the array q^*. At the circuit level in Fig. 9b, this is equivalent to applying n Hadamard gates to the qubits in q^*. This example illustrates the use of loop invariant in our framework. Also, the operator $\mathcal{H}_n$ is utilized in both Deutsch–Jozsa's algorithm §.6.3 and Grover's algorithm §.6.4.

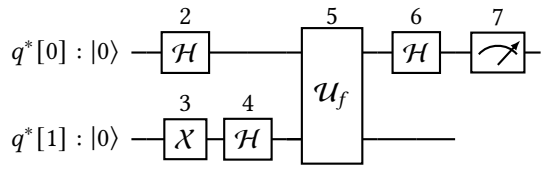
In Fig. 10, we prove that $\mathcal{H}_n$ transforms the state of the each qubit in q^* from $|0\rangle$ into $|+\rangle$. Accordingly, we let $\circledast_{k=0}^{n-1}\ q^*[k] \mapsto |0\rangle$ be our precondition and $\circledast_{k=0}^{n-1}\ q^*[k] \mapsto |+\rangle$ be our postcondition.

1 $\{\ \circledast_{k=0}^{n-1} q^*[k] \mapsto |0\rangle\ \}$
2 $i := 0;$
3 $\{\ i = 0\ \wedge \left(\circledast_{k=0}^{n-1} q^*[k] \mapsto |0\rangle\right)\ \}$
4 $\{\ I : i \le n \wedge \left(\circledast_{k=0}^{i-1} q^*[k] \mapsto |+\rangle\right) \circledast \left(\circledast_{k=i}^{n-1} q^*[k] \mapsto |0\rangle\right)\ \}(\overset{\circledast}{\mapsto}, \overline{\circledast^{\wedge}})$
5 **while** $i < n$ **do**
6 $\{\ i < n \wedge \left(\circledast_{k=1}^{i-1} q^*[k] \mapsto |+\rangle\right) \circledast \left(\circledast_{k=i}^{n-1} q^*[k] \mapsto |0\rangle\right)\ \}(\overline{\circledast^{\wedge}})$
7 $\iff \{\ q^*[i] \mapsto |0\rangle\ \}\ \mathcal{H}(q^*[i]);\ \{\ q^*[i] \mapsto |+\rangle\ \}$
8 $\{\ i < n \wedge \left(\circledast_{k=0}^{i} q^*[k] \mapsto |+\rangle\right) \circledast \left(\circledast_{k=i+1}^{n-1} q^*[k] \mapsto |0\rangle\right)\ \}(\overset{\circledast}{\mapsto}, \overline{\circledast^{\wedge}})$
9 $i := i + 1;$
10 $\{\ I : i \le n \wedge \left(\circledast_{k=0}^{i-1} q^*[k] \mapsto |+\rangle\right) \circledast \left(\circledast_{k=i}^{n-1} q^*[k] \mapsto |0\rangle\right)\ \}$
11 $\{\ I \wedge i \ge n\ \}$
12 $\{\ \circledast_{k=0}^{n-1} q^*[k] \mapsto |+\rangle\ \}(\circledast^{I})$

Fig. 10. Proof of the Hadamard operator for n qubits $\mathcal{H}_n$

1 $q^* := \mathbf{qbit}(2);$
2 $\mathcal{H}(q^*[0]);$
3 $\mathcal{X}(q^*[1]);$
4 $\mathcal{H}(q^*[1]);$
5 $\mathcal{U}_f(q^*);$
6 $\mathcal{H}(q^*[0]);$
7 $v := \mathbf{measure}(q^*[0]);$
8 $\mathbf{dispose}(q^*);$

(a) The algorithm's code



(b) The algorithm's circuit design

Fig. 11. Deutsch's algorithm

First, we apply the rule Assign to derive the postcondition for the assignment $i := 0$. Using the entailment rules in Fig. 8, we arrive at the invariant I in line 4. Here I partitions the qubits in q^* into the transformed part $q^*[0, i-1]$ and the residue part $q^*[i, n-1]$. Besides, the condition $i \le n$ in I allows us to derive $i = n$ after the loop. Using the rule While, we enter the loop with the predicate $I \wedge i < n$. In line 7, we apply the frame rule QFrame and the rule Trans to reason about the state transformation of the qubit $q^*[i]$. We then apply the rule Assign for the assignment $i := i + 1$ and recover the invariant I. Using the rule While, we exit the loop with the predicate $I \wedge i \ge n$ from which we can derive the postcondition $\circledast_{k=0}^{n-1} q^*[k] \mapsto |+\rangle$.

6.2 Deutsch's Algorithm

Fig. 11 presents Deutsch's algorithm [Deutsch and Jozsa 1992] with its code on the left (Fig.11a) and its quantum circuit on the right (Fig. 11b). The number above each gate on the right matches

1 $\{\ |\mathbf{emp}\rangle\ \}$
2 $q^* := \mathbf{qbit}(2);$
3 $\{\ q^*[0] \mapsto |0\rangle \circledast q^*[1] \mapsto |1\rangle \wedge |q^*| = 2\ \}$
4 $\iff \{\ q^*[0] \mapsto |0\rangle\ \} \mathcal{H}(q^*[0]); \{\ q^*[0] \mapsto |+\rangle\ \}$
5 $\implies \{\ q^*[1] \mapsto |0\rangle\ \}$
6 $\mathcal{X}(q^*[1]);$
7 $\{\ q^*[1] \mapsto |1\rangle\ \}$
8 $\mathcal{H}(q^*[1]);$
9 $\impliedby \{\ q^*[1] \mapsto |-\rangle\ \}$
10 $\{\ q^*[0] \mapsto |+\rangle \circledast q^*[1] \mapsto |-\rangle \wedge |q^*| = 2\ \}$
11 $\{\ q^*[0,1] \mapsto \frac{1}{2}|0\rangle|0\rangle - \frac{1}{2}|0\rangle|1\rangle + \frac{1}{2}|1\rangle|0\rangle - \frac{1}{2}|1\rangle|1\rangle \wedge |q^*| = 2\ \}$
12 $\implies \{\ q^*[0,1] \mapsto \frac{1}{2}|0\rangle|0\rangle - \frac{1}{2}|0\rangle|1\rangle + \frac{1}{2}|1\rangle|0\rangle - \frac{1}{2}|1\rangle|1\rangle\ \}$
13 $\mathcal{U}_f(q^*[0,1]);$
14 $\{\ q^*[0,1] \mapsto \frac{1}{2}|0\rangle|f(0)\rangle - \frac{1}{2}|0\rangle|1 \oplus f(0)\rangle + \frac{1}{2}|1\rangle|f(1)\rangle - \frac{1}{2}|1\rangle|1 \oplus f(1)\rangle\ \}$
15 $\impliedby \{\ q^*[0] \mapsto \frac{1}{\sqrt{2}}|0\rangle + \frac{(-1)^{f(0)\oplus f(1)}}{\sqrt{2}}|1\rangle \circledast q^*[1] \mapsto (-1)^{f(0)}|-\rangle\ \}$
16 $\implies \{\ q^*[0] \mapsto \frac{1}{\sqrt{2}}|0\rangle + \frac{(-1)^{f(0)\oplus f(1)}}{\sqrt{2}}|1\rangle\ \}$
17 $\mathcal{H}(q^*[0]);$
18 $\{\ q^*[0] \mapsto \frac{1+(-1)^{f(0)\oplus f(1)}}{2}|0\rangle + \frac{1-(-1)^{f(0)\oplus f(1)}}{2}|1\rangle\ \}$
19 $\{\ (f(0) \oplus f(1) = 0 \wedge q^*[0] \mapsto |0\rangle) \vee (f(0) \oplus f(1) = 1 \wedge q^*[0] \mapsto |1\rangle)\ \}$
20 $v := \mathbf{measure}(q^*[0]);$
21 $\impliedby \{\ (v = 0 \wedge f_= \wedge q^*[0] \mapsto |0\rangle) \vee (v = 1 \wedge f_\neq \wedge q^*[0] \mapsto |1\rangle)\ \}$
22 $\{\ (v = 0 \wedge f_= \wedge |q^*| = 2 \wedge q^*[0,1] \mapsto \ldots) \vee (v = 1 \wedge f_\neq \wedge |q^*| = 2 \wedge q^*[0,1] \mapsto \ldots)\ \}$
23 $\implies \{\ |q^*| = 2 \wedge q^*[0,1] \mapsto \ldots\ \}$
24 $\mathbf{dispose}(q^*);$
25 $\impliedby \{\ |\mathbf{emp}\rangle\ \}$
26 $\{\ (v = 0 \wedge f_= \wedge |\mathbf{emp}\rangle) \vee (v = 1 \wedge f_\neq \wedge |\mathbf{emp}\rangle)\ \}$
27 $\{\ (f_= \leftrightarrow v = 0) \wedge (f_\neq \leftrightarrow v = 1) \wedge |\mathbf{emp}\rangle\ \}$

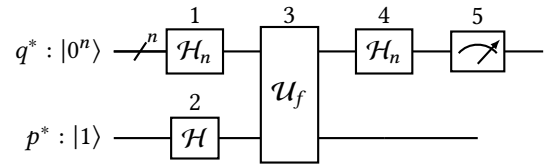
Fig. 12. Proof of Deutsch's algorithm

the number of the corresponding statement on the left. Deutsch's algorithm is a special case of Deutsch–Jozsa's algorithm for n qubits [Deutsch and Jozsa 1992]. We will explore the latter in §6.3.

The problem is as follows. Let $f : \{0,1\}^n \mapsto \{0,1\}$ be a Boolean function whose domain consists of bit strings of length n. We say f is *constant* if it is 0 on all outputs or 1 on all outputs, and is *balanced* if exactly half of its outputs is 0 and the other half is 1. Knowing that f is either constant or balanced, we want to determine its type efficiently. Note that a classical computer needs to compute the entire domain of f in the worst case to obtain the answer. Thanks to the quantum superposition, quantum computer only needs to compute f once for the result, and thus achieves an exponentially computational speedup. The function f is encoded by an oracle quantum gate $\mathcal{U}_f : \mathbb{V}_{\mathcal{B}}^{n+1} \mapsto \mathbb{V}_{\mathcal{B}}^{n+1}$ that maps each basic vector $|e_i\rangle|e'_j\rangle$ — where $|e_i\rangle \in \mathbb{B}^n$ and $|e'_j\rangle \in \mathbb{B}$ — to the basis vector $|e_i\rangle|e'_j \oplus f(e_i)\rangle$. Here $\oplus$ is the Boolean XOR operator and the last ancillary qubit in $\mathcal{U}_f$ stores the result of the computation. Accordingly, $\mathcal{U}_f$ can be expressed in the following Dirac

1 $\mathcal{H}_n(q^*)$;
2 $\mathcal{H}(p^*)$;
3 $\mathcal{U}_f(q^*p^*)$;
4 $\mathcal{H}_n(q^*)$;
5 $v := \textbf{measure}(q^*)$;

(a) The algorithm's code



(b) The algorithm's circuit design

Fig. 13. Deutsch–Jozsa's algorithm

notation:

$$\mathcal{U}_f \triangleq \sum_{i,j} |e_i\rangle|e'_j \oplus f(e_i)\rangle\langle e_i|\langle e'_j| \text{ where } |e_i\rangle \in \mathbb{B}^n \text{ and } |e'_j\rangle \in \mathbb{B} \tag{16}$$

Deutsch's algorithm solves the special case when $n = 1$ and thus every $f : \{0,1\} \mapsto \{0,1\}$ is either constant or balanced. In particular, the operator $\mathcal{U}_f$ for two qubits is expressed as:

$$\mathcal{U}_f = |0\rangle|f(0)\rangle\langle 0|\langle 0| + |0\rangle|1 \oplus f(0)\rangle\langle 0|\langle 1| + |1\rangle|f(1)\rangle\langle 1|\langle 0| + |1\rangle|1 \oplus f(1)\rangle\langle 1|\langle 1| \tag{17}$$

We write $f_=$ to indicate that f is constant and $f_\neq$ to indicate that f is balanced. The two predicates can be defined as:

$$f_= \triangleq f(0) \oplus f(1) = 0 \qquad f_\neq \triangleq f(0) \oplus f(1) = 1 \tag{18}$$

We explain the code in Fig. 11a. Initially, q^* is allocated with two qubits. The first qubit is transformed by the Hadamard operator $\mathcal{H}$ whereas the other qubit is transformed by the Pauli-X operator $\mathcal{X} \triangleq |0\rangle\langle 1| + |1\rangle\langle 0|$ followed by $\mathcal{H}$. The operator $\mathcal{X}$ — which is similar to the classical NOT gate — allows us to flip the state of $q^*[1]$ from $|0\rangle$ to $|1\rangle$. After that, $\mathcal{U}_f$ is applied to both qubits and $\mathcal{H}$ is applied to the first qubit. Then the state of the first qubit is measured and the result is assigned to v. Finally, the qubits in q^* are deallocated. Deutsch's algorithm ensures that the value of v implies the type of f, i.e. $v = 0$ iff f is constant and $v = 1$ iff f is balanced. The qubit allocation and deallocation are not present in the original algorithm but are amended there to illustrate the rules for qubit allocation/deallocation.

We now navigate through the proof in Fig. 12. The rule Qubit is applied to derive the state of $q^*[0,1]$ after the allocation. In lines 4–9, the states of the qubits in q^* are first reasoned locally using the frame rule and then combined in line 10. We apply the $\overset{\circledast}{\mapsto}$–rule to obtain the combined state of $q^*[0,1]$ and then use the rule Trans to derive the postcondition of the transformation $\mathcal{U}_f(q^*[0,1])$. In line 15, we tensor-factorize the combined state of $q^*[0,1]$ and apply the $\overset{\circledast}{\mapsto}$–rule to split the states of $q^*[0]$ and $q^*[1]$. In lines 16-21, the state of $q^*[0]$ is reasoned locally. After applying $\mathcal{H}$, the state of $q^*[0]$ is $|0\rangle$ iff f is constant and $|1\rangle$ iff f is balanced. As a result, measuring $q^*[0]$ is sufficient to determine the type of f. Line 21 captures the postcondition of the measurement where $v = 0$ when f is constant and $v = 1$ otherwise.

The remaining proof reasons about the deallocation of q^*. To derive the postcondition, we first apply the rule Disj to split the disjunction in line 22, then the frame rule to frame out the classical variable v, followed by the rule Dis to deallocate q^*.

6.3 Deutsch–Jozsa's Algorithm

We now solve the general case of Deutsch–Jozsa's algorithm for n qubits [Deutsch and Jozsa 1992]. The program and its circuit are given in Fig. 13. We assume that n qubits in $q^*[0, n-1]$

have states $|0\rangle$ and the single qubit in $p^*[0]$ as state $|1\rangle$ initially. For convenience, we write q^* for $q^*[0, n-1]$ and p^* for $p^*[0]$. The algorithm executes the following steps sequentially:

(1) Apply $\mathcal{H}_n$ (the n-bit Hadamard gate, §6.1) to q^* and $\mathcal{H}$ to p^*
(2) Apply $\mathcal{U}_f$ to the concatenation array q^*p^*
(3) Apply $\mathcal{H}_n$ to q^*. Then measure q^* and assign the result to v.

The type of f is determined via the following mathematical fact:

Lemma 6.1. *Let* $\alpha \triangleq \left(\frac{1}{2^n}\sum_{i=0}^{2^n-1}(-1)^{f(i)}\right)^2$. *Then* $\alpha = 1$ *if* f *is constant and* $\alpha = 0$ *if* f *is balanced.*

Intuitively, α represents the probability associated with the state $|0^n\rangle$ in q^*. Thus the state $|0^n\rangle$ is measured (i.e. $v = 0$) with probability 1 if f is constant and with probability 0 (i.e. $v \neq 0$) if f is balanced. The former also implies that the probabilities of other outcomes are all zero. As before, we use the predicates $f_=$ and $f_\neq$ to indicate that the function f is constant and balanced respectively.

We introduce several intermediate computation results here to offload the complexity of the proof. Let $i \cdot j \triangleq \bigoplus_{k=0}^{n-1} i_k j_k$ be the XOR-sum of bit product and $g(j) \triangleq \frac{1}{2^n}\sum_{i=0}^{2^n-1}(-1)^{f(i)}(-1)^{i\cdot j}$. The states of the qubits after applying $\mathcal{U}_f$ and $\mathcal{H}_n$ satisfy the following properties:

Lemma 6.2. *Let* $|v\rangle \triangleq \frac{1}{\sqrt{2^n}}\sum_{i=0}^{2^n-1}(-1)^{f(i)}$ *and* $|v_1\rangle \triangleq \sum_{j=0}^{2^n-1} g(j)|j\rangle$. *Then:*

$$\mathcal{U}_f(|+^n\rangle|-\rangle) = |v\rangle|-\rangle \qquad \text{and} \qquad \mathcal{H}_n(|v\rangle) = |v_1\rangle$$

Note that $g(0) = \frac{1}{2^n}\sum_{i=0}^{2^n-1}(-1)^{f(i)}$ and so $\alpha = |g(0)|^2$ is the probability associated with the state $|0^n\rangle$ in $|v_1\rangle$. Thus Lemma. 6.1 gives us the desired result.

We take a close look at Fig. 14. In lines 2-3, the states of q^* and p^* are transformed into $|+^n\rangle$ and $|-\rangle$ respectively by the Hadamard operators. Then the combined state of q^* and p^* is transformed into $|v\rangle|-\rangle$ by $\mathcal{U}_f$. This allows us to apply the $\overset{\circledast}{\mapsto}$–rule to split the states of q^* and p^* separately. In lines 8-12, we use the frame rule to reason about the qubits in q^* locally. The results of the transformation by $\mathcal{H}_n$ and of the measurement follow from Lemma. 6.2. In line 13, we use the entailment rules in Fig. 8 — in particular the $\mathbb{P}^\perp$–rule to eliminate infeasible measurement outcomes — and reach the postcondition as desired.

6.4 Grover's Algorithm

Fig. 15 displays the implementation of Grover's algorithm [Grover 1996] with its code on the left (Fig. 15a) and its circuit on the right (Fig. 15b). This algorithm finds a particular input of a function f — with high certainty — using only $O(\sqrt{N})$ evaluations of f, where $N > 1$ is the size of f's domain. In contrast, classical algorithms cannot solve this problem with less than $O(N)$ evaluations. Moreover, Grover's algorithm is asymptotically optimal [Bennett et al. 1997]. For convenience, we assume that $N = 2^n$ and f maps each basic vector in $\mathbb{B}^n$ to $\{0, 1\}$ such that $f(|e_i\rangle) = 1$ iff $i = \omega$ and 0 otherwise. Our task is to identify the index ω. The function f is computed by the following quantum operator $\mathcal{G}_f$ where the last ancillary qubit in $\mathcal{G}_f$ stores the computation of the function f:

$$\mathcal{G}_f \triangleq \sum_{i,j} |e_i\rangle|e'_j \oplus f(|e_i\rangle)\rangle\langle e_i|\langle e'_j| \text{ where } |e_i\rangle \in \mathbb{B}^n \text{ and } |e'_j\rangle \in \mathbb{B} \tag{19}$$

We explain the code in Fig. 15a. Initially, there are n qubits of state $|0\rangle$ in $q^*[0, n-1]$ and a single qubit of state $|1\rangle$ in $p^*[0]$. For convenience, we refer the array segments $q^*[0, n-1]$ and $p^*[0]$ as q^* and p^* respectively. Grover's algorithm executes the following sequence of steps:

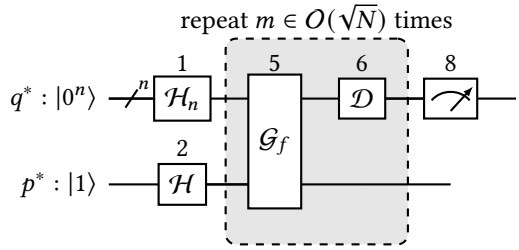
(1) Apply the Hadamard operators $\mathcal{H}_n$ to q^*, and $\mathcal{H}$ to p^*

1 $\{\ q^* \mapsto |0^n\rangle \circledast p^* \mapsto |1\rangle\ \}$
2 $\Longleftrightarrow \{\ q^* \mapsto |0^n\rangle\ \}\mathcal{H}_n(q^*); \{\ q^* \mapsto |+^n\rangle\ \}$
3 $\Longleftrightarrow \{\ p^* \mapsto |1\rangle\ \}\mathcal{H}(p^*); \{\ p^* \mapsto |-\rangle\ \}$
4 $\{\ q^*p^* \mapsto |+^n\rangle|-\rangle\ \}$
5 $\mathcal{U}_f(q^*p^*);$
6 $\{\ q^*p^* \mapsto |v\rangle|-\rangle\ \}$
7 $\{\ q^* \mapsto |v\rangle \circledast p^* \mapsto |-\rangle\ \}$
8 $\Longrightarrow \{\ q^* \mapsto |v\rangle\ \}$
9 $\quad \mathcal{H}_n(q^*);$
10 $\quad \{\ q^* \mapsto |v_1\rangle\ \}$
11 $\quad v := \textbf{measure}(q^*);$
12 $\Longleftarrow \{\ \bigvee_i (v = i \wedge |g(i)|^2 \cdot q^* \mapsto |i\rangle)\ \}$
13 $\{\ \bigvee_i (v = i \wedge |g(i)|^2 \cdot q^*p^* \mapsto |i\rangle|-\rangle)\ \}$
14 $\{\ (v = 0 \leftrightarrow f_=) \wedge (v \neq 0 \leftrightarrow f_\neq)\ \}$

Fig. 14. Proof of Deutsch–Jozsa's algorithm

1 $\mathcal{H}_n(q^*);$
2 $\mathcal{H}(p^*);$
3 $i := 0;$
4 **while** $i < m$ **do**
5 $\quad \mathcal{G}_f(q^*p^*);$
6 $\quad \mathcal{D}(q^*);$
7 $\quad i := i + 1;$
8 $v := \textbf{measure}(q^*);$

(a) The algorithm's code



(b) The algorithm's circuit design

Fig. 15. Grover's algorithm

(2) Apply the operator $\mathcal{G}_f$ to the concatenation q^*p^* and then the *diffusion operator* $\mathcal{D}$ to q^*. The specs of $\mathcal{D}$ is given as:

$$\mathcal{D} \triangleq 2|+^n\rangle\langle+^n| - \mathcal{I}_n \tag{20}$$

Repeat this step $m \in O(\sqrt{N})$ times to amplify the magnitude of $|e_\omega\rangle$ in q^*
(3) Measure q^* and assign the result to v.

We encode the loop invariant I in step 2 using several ingredients from trigonometry. Let $|v\rangle \triangleq \sum_{k\neq\omega} |e_k\rangle/\sqrt{N-1}$ and $a_i \triangleq \cos((2i+1)\alpha), b_i \triangleq \sin((2i+1)\alpha)$ where $\alpha \triangleq \arcsin(1/\sqrt{N})$. We express $|+^n\rangle$ — the state of q^* before the loop — as the combination of the two orthornormal vectors $\{|v\rangle, |e_\omega\rangle\}$ as follows:

$$|+^n\rangle = \frac{1}{\sqrt{N}}\sum_k |e_k\rangle = \cos(\alpha)|v\rangle + \sin(\alpha)|e_\omega\rangle = a_0|v\rangle + b_0|e_\omega\rangle \quad \text{where} \quad \langle v|e_\omega\rangle = 0 \tag{21}$$

1 $\{\ q^* \mapsto |0^n\rangle \circledast p^* \mapsto |1\rangle\ \}$
2 $\Longleftrightarrow \{\ q^* \mapsto |0^n\rangle\ \}\mathcal{H}_n(q^*);\{\ q^* \mapsto |+^n\rangle\ \}$
3 $\Longleftrightarrow \{\ p^* \mapsto |1\rangle\ \}\mathcal{H}(p^*);\{\ p^* \mapsto |-\rangle\ \}$
4 $\{\ q^* \mapsto |+^n\rangle \circledast p^* \mapsto |-\rangle\ \}$
5 $i := 0;$
6 $\{\ i = 0 \wedge q^* \mapsto |+^n\rangle \circledast p^* \mapsto |-\rangle\ \}$
7 $\{\ I : i \leq m \wedge q^* \mapsto a_i|v\rangle + b_i|e_\omega\rangle \circledast p^* \mapsto |-\rangle\ \}$
8 **while** $i < m$ **do**
9 $\{\ i < m \wedge q^*p^* \mapsto (a_i|v\rangle + b_i|e_\omega\rangle) \otimes |-\rangle\ \}$
10 $\Longleftrightarrow \{\ q^*p^* \mapsto (a_i|v\rangle + b_i|e_\omega\rangle) \otimes |-\rangle\ \}\mathcal{G}_f(q^*p^*);\{\ q^*p^* \mapsto (a_i|v\rangle - b_i|e_\omega\rangle) \otimes |-\rangle\ \}$
11 $\Longleftrightarrow \{\ q^* \mapsto a_i|v\rangle - b_i|e_\omega\rangle\ \}\mathcal{D}(q^*);\{\ q^* \mapsto a_{i+1}|v\rangle + b_{i+1}|e_\omega\rangle\ \}$
12 $\{\ i < m \wedge q^* \mapsto a_{i+1}|v\rangle + b_{i+1}|e_\omega\rangle \circledast p^* \mapsto |-\rangle\ \}$
13 $i := i + 1;$
14 $\{\ I : i \leq m \wedge q^* \mapsto a_i|v\rangle + b_i|e_\omega\rangle \circledast p^* \mapsto |-\rangle\ \}$
15 $\{\ I \wedge i \geq m\ \}$
16 $\{\ q^* \mapsto a_m|v\rangle + b_m|e_\omega\rangle \circledast p^* \mapsto |-\rangle\ \}$
17 $\Longrightarrow \{\ q^* \mapsto a_m|v\rangle + b_m|e_\omega\rangle\ \}$
18 $v := \mathbf{measure}(q^*);$
19 $\Longleftarrow \{\ (v = e_\omega \wedge |b_m|^2 \cdot q^* \mapsto |e_\omega\rangle) \vee (\bigvee_{k \neq \omega} v = e_k \wedge \frac{|a_m|^2}{N-1} \cdot q^* \mapsto |e_k\rangle)\ \}$
20 $\{\ \mathbb{P}(v = e_\omega) \geq |b_m|^2\ \}$

Fig. 16. Proof of Grover's algorithm

Each iteration in step 2 transforms the state of q^* from $a_i|v\rangle + b_i|e_\omega\rangle$ into $a_{i+1}|v\rangle + b_{i+1}|e_\omega\rangle$. Thus we pick I to be the following predicate where the counter i is set to 0 initially:

$$I \triangleq i \leq m \wedge q^* \mapsto a_i|v\rangle + b_i|e_\omega\rangle \circledast p^* \mapsto |-\rangle \tag{22}$$

As for the choice of m, we want the magnitude $|b_m|$ of $|v_\omega\rangle$ after m iterations to be close to 1, i.e. $(2m+1)\alpha \approx \pi/2$ or $m \approx \pi/4\alpha - 1/2 \approx \pi\sqrt{N}/4 - 1/2$. Here we approximate $\arcsin(1/\sqrt{N}) \approx 1/\sqrt{N}$ using the Maclaurin series expansion. For correctness, we let the postcondition be:

$$\mathbb{P}(v = e_\omega) = |b_m|^2 \tag{23}$$

That is, the probability associated with the outcome e_ω is $|b_m|^2$. By the justification above, it follows that $|b_m|^2$ is close to 1 and so the outcome e_ω is mostly probable.

We explain the proof in Fig. 16. Lines 2-3 contain the local proofs for quantum transformations $\mathcal{H}_n(q^*)$ and $\mathcal{H}(p^*)$. The loop invariant I is set in line 7. For each iteration, the operator $\mathcal{G}_f$ is applied to q^*p^* to flip the coefficient's sign of $|e_\omega\rangle$ in q^*, i.e. from $a_i|v\rangle + b_i|e_\omega\rangle$ to $a_i|v\rangle - b_i|e_\omega\rangle$. After that, the states of q^* and p^* become separable and the diffusion operator $\mathcal{D}$ is applied to q^*. The result of this transformation follows from Lemma 6.3 below:

Lemma 6.3. $\mathcal{D}(a_i|v\rangle - b_i|e_\omega\rangle) = a_{i+1}|v\rangle + b_{i+1}|e_\omega\rangle$.

In line 14, the invariant I is recovered after the increment of the counter i. We exit the loop with the predicate $I \wedge i \geq m$ which allows us to deduce that the state of q^* is $a_m|v\rangle + b_m|e_\omega\rangle$. After q^* is measured, the predicate in line 19 implies that $v = e_\omega \rightarrow |b_m|^2 \cdot q^* \mapsto |e_\omega\rangle$. As $|b_m|^2 \approx 1$, we can deduce that $|b_m|^2 > \frac{1}{2}$. By applying the probability rule for pure predicates in §5.3, we reach the postcondition as desired.

7 OPERATIONAL SEMANTICS

We define the operational semantics and use it to prove the soundness of the framework in §5, starting from the definition of quantum heaps in §7.1, then the forcing sematics for $\models$ §7.2, to the step semantics §7.3 and the Hoare triples in §7.4. Our semantics expresses quantum systems in their pure states with tagged probabilities, which allows us to reuse the constructs of sequential semantics for SL in [Yang and O'Hearn 2002]. We also discuss on the limitations of our framework in §7.5.

We adopt the following notations for lists. We write nil for empty list, $[t_1, \ldots, t_n]$ for the list enumeration, $|l|$ for the length of l, $l^\frown l'$ for the concatenation of l and l', $l[n, n']$ for the sublist of l from index n to n' inclusively.

7.1 Quantum Heaps

We need several notations for the definition of quantum heaps. We write $[n]$ for the set $\{0, \ldots, n-1\}$ where $n \in \mathbb{N}$. We call a bijective mapping $\overline{\sigma} : [n] \mapsto [n]$ an *index permutation over* $[n]$ — or *n-perm* for short. We override $\overline{\sigma}$ to lists and vectors as follows where $|e_i\rangle \in \mathbb{B}$ and $|e'_j\rangle \in \mathbb{B}^n$:

$$\begin{gathered} \overline{\sigma}([t_0, \ldots, t_{n-1}]) = [t_{\sigma(0)}, \ldots, t_{\sigma(n-1)}] \\ \overline{\sigma}(|e_0 \ldots e_{n-1}\rangle) = \overline{\sigma}(|e_{\overline{\sigma}(0)} \ldots e_{\overline{\sigma}(n-1)}\rangle) \qquad \overline{\sigma}\left(\textstyle\sum_j a_j |e'_j\rangle\right) \triangleq \sum_j a_j \overline{\sigma}(|e'_j\rangle) \end{gathered} \tag{24}$$

These permutations allows us to rearrange the qubits in the quantum states. Let $\Sigma \subseteq \mathbb{N}$ be a finite set such that $|\Sigma| = n$. Its *permutation set* $\mathcal{S}(\Sigma)$ contains all permutations of Σ. For example:

$$\mathcal{S}(\{1, 2, 3\}) = \{[1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1]\} \tag{25}$$

We use Σ represent the ids of the qubits. We require that the quantum heap for Σ contains all of the possible permutations of these qubits. This grants us the flexibility to conveniently rearrange the qubits in the system. More importantly, it is a sufficient condition to make the heap join operator commutative. With that in mind, the quantum heap is defined as follows:

Quantum heaps. The mapping $Q : \mathcal{S}(\Sigma) \mapsto \mathbb{V}^n_{\mathcal{B}} \backslash \{0\}$ is a quantum heap if for every list $l \in \mathcal{S}(\Sigma)$ and n-perm $\overline{\sigma}$, we have

$$\|Q(l)\| = 1 \quad \text{and} \quad Q(\overline{\sigma}(l)) = \overline{\sigma}(Q(l)) \tag{26}$$

That is, each vector in the co-domain of Q has norm 1 and the state of the permuted list $\overline{\sigma}(l)$ is the permuted vector $\overline{\sigma}(Q(l))$. We call Σ the *qubit domain* of Q. We let the *empty heap* be the singleton mapping $Q_e : \{\text{nil}\} \mapsto \{1\}$ from the empty list to the scalar 1.

Orientations. A single evaluation $Q(l)$ is sufficient to determine the entire quantum heap Q. This is because for any list $l' \in \mathcal{S}(\Sigma)$, there exists a unique n-perm $\overline{\sigma}$ such that $l' = \overline{\sigma}(l)$. Thus we have $Q(l') = Q(\overline{\sigma}(l)) = \overline{\sigma}(Q(l))$. For example, let $\Sigma_Q = \{1, 2, 3\}$ and $Q([1, 2, 3]) = \frac{1}{\sqrt{2}}|101\rangle + \frac{1}{\sqrt{2}}|100\rangle$ then $Q([2, 3, 1]) = \frac{1}{\sqrt{2}}|011\rangle + \frac{1}{\sqrt{2}}|001\rangle$, $Q([3, 2, 1]) = \frac{1}{\sqrt{2}}|101\rangle + \frac{1}{\sqrt{2}}|001\rangle$, etc.

Let $Q(l) = |v\rangle$. The singleton mapping $Q_l : \{l\} \mapsto \{|v\rangle\}$ is called an *orientation* of Q. From the above observation, the quantum heap Q can be defined from one of the orientations Q_l since they are just permutation of each others. Nevertheless, orientations are ease to handle and the constructs over quantum heaps can be defined from the corresponding constructs over orientations via lifting. For convenience, we often write $Q(l) = |v\rangle$ to specify the orientation of the quantum heap Q.

Disjointness. We introduce the notion of *disjointness* for quantum heaps. Two heaps Q, Q' are disjoint — denoted by $Q \perp Q'$ — if their qubit domains $\Sigma_Q, \Sigma_{Q'}$ are disjoint, i.e. $\Sigma_Q \cap \Sigma_{Q'} = \emptyset$. Furthermore, if $Q \perp Q'$ we can join them using the *heap join operator* $\uplus$. The *joined heap* $Q \uplus Q'$ over $\Sigma_Q \cup \Sigma_{Q'}$ is defined by the following orientation:

$$(Q \uplus Q')(l^\frown l') \triangleq Q(l) \otimes Q'(l') \text{ for some } l \in \text{dom}(Q), l' \in \text{dom}(Q') \tag{27}$$

$$
\begin{array}{lll}
\delta, Q, \sigma \models \overline{P} & \text{iff} & [\![\overline{P}]\!]_\sigma = \textbf{true} \\
\delta, Q, \sigma \models e^* \mapsto |v\rangle & \text{iff} & \delta = 1 \text{ and } Q([\![e^*]\!]_\sigma) \cong [\![|v\rangle]\!]_\sigma \\
\delta, Q, \sigma \models e \cdot F & \text{iff} & \delta \leq [\![e]\!]_\sigma \leq 1 \text{ and } \delta/[\![e]\!]_\sigma, Q, \sigma \models F \\
\delta, Q, \sigma \models F_1 \circledast F_2 & \text{iff} & \text{exist } Q_1 \perp Q_2 \text{ and } \delta_1, \delta_2 \in (0,1] \text{ s.t. } \delta_1\delta_2 = \delta \text{ and} \\
 & & Q_1 \circledcirc Q_2 = Q \text{ and } \delta_1, Q_1, \sigma \models F_1 \text{ and } \delta_2, Q_2, \sigma \models F_2
\end{array}
$$

Fig. 17. Forcing semantics for quantum predicates

We end this subsection by pointing out that the quantum heaps satisfy the axioms of Separation Algebra in [Calcagno et al. 2007] — thus possess similar characteristics as of traditional heaps in SL.

Theorem 7.1. *Let $\mathbb{Q}$ be the set of all quantum heaps. The structure $(\mathbb{Q}; \circledcirc; Q_e)$ is a Separation Algebra. That is, the following properties are satisfied, where the left hand side in each equality is defined iff the right hand side is defined:*

(1) **Identity element:** $Q \circledcirc Q_e = Q_e \circledcirc Q = Q$
(2) **Commutative:** $Q_1 \circledcirc Q_2 = Q_2 \circledcirc Q_1$
(3) **Associative:** $(Q_1 \circledcirc Q_2) \circledcirc Q_3 = Q_1 \circledcirc (Q_2 \circledcirc Q_3)$
(4) **Cancellative:** $Q \circledcirc Q_1 = Q \circledcirc Q_2 \Rightarrow Q_1 = Q_2$.

7.2 Forcing Semantics

Our *quantum program state* s is the triple (δ, Q, σ) where $0 < \delta \leq 1$ is the associated probability, Q the quantum heap, and σ the *variable stack* that maps classical variables to integers and quantum variables to *lists of distinct integers*. Intuitively, σ represents the classical component (i.e. the classical computer) and Q represents the quantum component (i.e. the quantum circuit). The following interactions between σ and Q are enforced to preserve the laws of physics:

(1) σ can only read measurements from Q, and
(2) Q can only read binary inputs from σ.

We write $[\![\cdot]\!]_\sigma$ to denote the evaluation by the stack σ over expressions and pure predicates. In particular, σ maps the empty array $[]$ to the empty list nil, the sub-array $q^*[e_1, e_2]$ to the sub-list $l[v_1, v_2]$ where $[\![q^*]\!]_\sigma = l$ and $[\![e_i]\!]_\sigma = v_i$, and the concatenation $e_1^* e_2^*$ to the concatenated list $l_1 \frown l_2$ where $[\![e_i^*]\!]_\sigma = l_i$. For the concatenation, we further require that the intersection of l_1 and l_2 to be empty to comply with the no-cloning theorem [Wootters and Zurek 1982] which states that the quantum states cannot be perfectly cloned in general.

We write $s \models \Phi$ to indicate that the quantum state s satisfies the quantum predicate Φ. The core interpretations of $\models$ are given in Fig. 17. The interpretations for $\{\wedge, \vee, \exists, \forall\}$ are standard and so omitted. Pure formula $\overline{P}$ is evaluated solely by σ and is independent of δ and Q. As for the quantum mapping $e^* \mapsto |v\rangle$, we require that the probability δ is 1, and $Q([\![e^*]\!]_\sigma)$ is equivalent to the quantum state specified by $|v\rangle$. Here two states $|v_1\rangle$ and $|v_2\rangle$ are equivalent if they differ by a scalar factor of modulus 1, i.e.:

$$|v_1\rangle \cong |v_2\rangle \text{ iff there exists } c \in \mathbb{C} \text{ s.t. } |c| = 1 \text{ and } |v_1\rangle = c|v_2\rangle \tag{28}$$

Lemma 7.2. *The relation $\cong$ is an equivalence relation. Furthermore, it is norm-preserving and closed under linear transformation and tensor product. More precisely, let $\mathcal{V}, \mathcal{V}'$ be Hilbert spaces and $|v_1\rangle, |v_2\rangle \in \mathcal{V}$ and $|v_1'\rangle, |v_2'\rangle \in \mathcal{V}'$ such that $|v_1\rangle \cong |v_2\rangle$ and $|v_1'\rangle \cong |v_2'\rangle$. Then:*

$$\||v_1\rangle\| = \||v_2\rangle\| \qquad \mathcal{T}(|v_1\rangle) \cong \mathcal{T}(|v_2\rangle) \qquad |v_1\rangle \otimes |v_1'\rangle \cong |v_2\rangle \otimes |v_2'\rangle$$

where $\mathcal{T}$ is a linear transformation in $\mathcal{V}$.

The equivalence relation $\cong$ allows us to indiscriminate quantum states that are observably indistinguishable, i.e. no sequence of quantum operators can differentiate one from the other. For example, two quantum states $\frac{|0\rangle+|1\rangle}{\sqrt{2}}$ and $\frac{-|0\rangle-|1\rangle}{\sqrt{2}}$ are observably indistinguishable and so are considered to be equivalent. Semantic-wise, the equivalence relation helps us to ignore the residue constant factor in the **dispose** statement.

For the fractional formula $e \cdot F$, we need the evaluation $[\![e]\!]_\sigma$ is in the range $[\delta, 1]$ — so that the probability $\delta/[\![e]\!]_\sigma$ is valid — and the state $(\delta/[\![e]\!]_\sigma, Q, \sigma)$ satisfies F. The semantics of $F_1 \circledast F_2$ is similar to its SL counterpart $F_1 \star F_2$. In detail, (δ, Q, σ) satisfies $F_1 \circledast F_2$ if the probability and the quantum heap can be split into $\delta = \delta_1, \delta_2$ and $Q = Q_1 \Cup Q_2$ such that (δ_1, Q_1, σ) satisfies F_1 and (δ_2, Q_2, σ) satisfies F_2.

We write $F_1 \models F_2$ — that F_2 is a logical consequence of F_1 — if $s \models P$ implies $s \models Q$ for every quantum state s. The soundness of our proof rules in Fig. 8 can be stated as follows:

Theorem 7.3. *The proof rules in Fig. 8 are sound with respect to $\models$. That is, if $F_1 \vdash F_2$ then $F_1 \models F_2$.*

7.3 Step Relation

A normal *configuration* (config) $\langle s, c\rangle$ consists of the state $s = (\delta, Q, \sigma)$ and the program c. A normal config is *final* if c is **skip**. Besides, we have a faulty config **abort** for the cases where the program accesses an invalid qubit outside the domain of the quantum heap.

We write $C_1 \rightsquigarrow C_2$ for the *step relation* between the two configs C_1, C_2. The small-step operational semantics for our quantum language is provided in Fig. 18. The step relation $\rightsquigarrow$ is non-deterministic as a consequence of the quantum measurement: a config C for the measurement statement can lead to multiple configs that correspond to different outcomes of the measurement. Notation-wise, we write $\sigma[v \Leftarrow e]$ for the updated stack in which the value of v in σ is updated to e.

In Fig. 18a, we focus on the non-faulty steps starting from the config $\langle(\delta, Q, \sigma), c\rangle$. Note that these steps construct new quantum heaps using orientations, i.e. the orientation $Q(l) = |v\rangle$ is used as a definition for the quantum heap Q itself.

(1) For the allocation $q^* := \mathbf{qbit}(e)$, the quantum heap Q is updated into $Q_1 \Cup Q$ where Q_1 stores $v = [\![e]\!]_\sigma$ fresh qubits with the initial states $|0\rangle$. Also, the variable q^* is mapped to a list consisting of v consecutive numbers exclusive from the qubit domain of Q. These numbers are the ids of the new qubits.
(2) For the deallocation $\mathbf{dispose}(q^*)$, the quantum heap is assumed to be $Q_1 \Cup Q_2$ where the heap Q_1 contains the disposing qubits in q^*. This implies that the qubits in q^* are separable from the remaining qubits in Q_2 and thus are safe for disposing. After the qubits in q^* are removed, Q_1 becomes the empty heap and thus the new quantum heap is Q_2. Besides, the variable stack is updated by mapping the variable q^* to the empty list nil.
(3) For the transformation $\mathcal{G}(e^*)$, the operator $\mathcal{G}$ is first extended to $\mathcal{G} \otimes \mathcal{I}_{|l|}$ to match the vector dimension of the qubits in the quantum heap. The state of the quantum heap is updated by applying $\mathcal{G} \otimes \mathcal{I}_{|l|}$ to the quantum state.
(4) For the measurement $v := \mathbf{measure}(e^*)$, the measurement operator is non-deterministically chosen to be $\mathcal{M}_k = (|e_k\rangle\langle e_k|) \otimes \mathcal{I}_{|l|}$ where the basis vector $|e_k\rangle$ matches the dimension of the qubits in e^*. The probability δ is updated into $\delta\delta_k$ where δ_k is the probability associated with the measurement operator $\mathcal{M}_k$. The quantum heap is updated by applying the operator $\mathcal{M}_k$ to the quantum state and then dividing by $\sqrt{\delta_k}$ for normalization.

Fig. 18b contains the faulty steps that lead to the dead-end config **abort**. The step Sa forces the program $c_1; c_2$ to go to **abort** if **abort** is reached from c_1. As for the remaining quantum steps, the config **abort** is reached whenever the program attempts to access a qubit outside the qubit

$$\frac{[\![e]\!]_\sigma = v > 0 \qquad n, \ldots, n+v-1 \notin \Sigma_Q \qquad l' = [n, \ldots, n+v-1] \qquad Q_1(l') = |0^v\rangle \qquad Q' = Q_1 \uplus Q \qquad \sigma' = \sigma[q^* \Leftarrow l']}{\langle (\delta, Q, \sigma), q^* := \mathbf{qbit}(e)\rangle \rightsquigarrow \langle (\delta, Q', \sigma'), \mathbf{skip}\rangle}\ \text{Qs}$$

$$\frac{[\![q^*]\!]_\sigma \in \mathrm{dom}(Q_1) \quad \sigma' = \sigma[q^* \Leftarrow \mathrm{nil}]}{\langle (\delta, Q_1 \uplus Q_2, \sigma), \mathbf{dispose}(q^*)\rangle \rightsquigarrow \langle (\delta, Q_2, \sigma'), \mathbf{skip}\rangle}\ \text{Ds}$$

$$\frac{[\![e^*]\!]_\sigma = l' \quad l'^\frown l \in \mathrm{dom}(Q) \quad \mathcal{G} : \mathbb{V}_{\mathcal{B}}^{|l'|} \mapsto \mathbb{V}_{\mathcal{B}}^{|l'|} \quad Q'(l'^\frown l) = (\mathcal{G} \otimes \mathcal{I}_{|l|})(Q(l'^\frown l))}{\langle (\delta, Q, \sigma), \mathcal{G}(e^*)\rangle \rightsquigarrow \langle (\delta, Q', \sigma), \mathbf{skip}\rangle}\ \text{Ts}$$

$$\frac{[\![e^*]\!]_\sigma = l' \quad l'^\frown l \in \mathrm{dom}(Q) \quad Q(l'^\frown l) = |a\rangle \quad \mathcal{M}_k = (|e_k\rangle\langle e_k|) \otimes \mathcal{I}_{|l|} \text{ where } |e_k\rangle \in \mathbb{B}^{|l'|} \quad \delta_k = \|\mathcal{M}_k|a\rangle\|^2 > 0 \quad \delta' = \delta\delta_k \quad Q'(l'^\frown l) = \frac{\mathcal{M}_k|a\rangle}{\sqrt{\delta_k}} \quad \sigma' = \sigma[v \Leftarrow e_k]}{\langle (\delta, Q, \sigma), v := \mathbf{measure}(e^*)\rangle \rightsquigarrow \langle (\delta', Q', \sigma'), \mathbf{skip}\rangle}\ \text{Ms}$$

(a) Non-faulty steps for quantum statements

$$\frac{\langle \Gamma, c_1\rangle \rightsquigarrow \mathbf{abort}}{\langle \Gamma, c_1; c_2\rangle \rightsquigarrow \mathbf{abort}}\ \text{Sa} \qquad \frac{e_i \in [\![e^*]\!]_\sigma \quad e_i \notin \Sigma_Q}{\langle (\delta, Q, \sigma), v := \mathbf{measure}(e^*)\rangle \rightsquigarrow \mathbf{abort}}\ \text{Ma}$$

$$\frac{e_i \in [\![e^*]\!]_\sigma \quad e_i \notin \Sigma_Q}{\langle (\delta, Q, \sigma), \mathcal{G}(e^*)\rangle \rightsquigarrow \mathbf{abort}}\ \text{Ta} \qquad \frac{e_i \in [\![q^*]\!]_\sigma \quad e_i \notin \Sigma_Q}{\langle (\delta, Q, \sigma), \mathbf{dispose}(q^*)\rangle \rightsquigarrow \mathbf{abort}}\ \text{Da}$$

(b) Faulty steps for **abort**

$$\frac{}{\langle \Gamma, \mathbf{skip}; c\rangle \rightsquigarrow \langle \Gamma, c\rangle} \qquad \frac{\langle \Gamma, c_1\rangle \rightsquigarrow \langle \Gamma', c_1'\rangle}{\langle \Gamma, c_1; c_2\rangle \rightsquigarrow \langle \Gamma', c_1'; c_2\rangle} \qquad \frac{\sigma' = \sigma[v \Leftarrow [\![e]\!]_\sigma]}{\langle (\delta, Q, \sigma), v := e\rangle \rightsquigarrow \langle (\delta, Q, \sigma'), \mathbf{skip}\rangle}$$

$$\frac{\Gamma = (\delta, Q, \sigma) \quad [\![b]\!]_\sigma = \mathbf{true}}{\langle \Gamma, \mathbf{if}\ b\ \mathbf{do}\ c_1\ \mathbf{else}\ c_2\rangle \rightsquigarrow \langle \Gamma, c_1\rangle} \qquad \frac{\Gamma = (\delta, Q, \sigma) \quad [\![b]\!]_\sigma = \mathbf{false}}{\langle \Gamma, \mathbf{if}\ b\ \mathbf{do}\ c_1\ \mathbf{else}\ c_2\rangle \rightsquigarrow \langle \Gamma, c_2\rangle}$$

$$\frac{}{\langle \Gamma, \mathbf{while}\ b\ \mathbf{do}\ c\rangle \rightsquigarrow \langle \Gamma, \mathbf{if}\ b\ \mathbf{do}\ \{c; \mathbf{while}\ b\ \mathbf{do}\ c\}\ \mathbf{else}\ \mathbf{skip}\rangle}$$

(c) Small step for standard statements

Fig. 18. Small step semantics for quantum programs

domain of the quantum heap. The steps for classical statements in Fig. 18c are standard and thus self-explanatory.

7.4 Soundness

Our Hoare triples for partial correctness faithfully adopt the formalism in [Yang and O'Hearn 2002, §3] for sequential SL. We say a config $\langle \delta, Q, \sigma\rangle$ is *safe* if it never leads to **abort**. Also, we let $\rightsquigarrow^*$ denote the closure of the step relation $\rightsquigarrow$. The triple $\{P\}c\{Q\}$ is valid, denoted by $\models \{P\}c\{Q\}$, if for any config $C = \langle s, c\rangle$ such that the state s satisfies P must satisfy the following conditions:

- (C_1) The config C is safe, and
- (C_2) If C leads to a final config $C' = \langle s', \mathbf{skip}\rangle$ (i.e. $C \rightsquigarrow^* C'$) then s' satisfies Q.

For total correctness, one can adopt the formalism in [Yang and O'Hearn 2002, §3] by including a third condition (C_3) requiring there is no infinite sequence of steps starting from the initial config C. We are now ready to state the main soundness result of our quantum rules in Fig. 6 and Fig. 7.

Theorem 7.4 (Soundness). *If* $\{P\}c\{Q\}$ *then* $\models \{P\}c\{Q\}$.

The quantum core rules in Fig. 7 can be proven directly from the step semantics in 7.3. The quantum frame rule QFrame — the heart of our reasoning framework — is proven with techniques similar to its classical counterpart in [Yang and O'Hearn 2002, §4]. The following lemma provides the key ingredients for the main proof:

Lemma 7.5. *Let* $Q_P \perp Q_F$ *and* $\delta_P, \delta_F \in (0,1]$. *Then*

(1) *If* $\langle(\delta_P, Q_P, \sigma), c\rangle$ *is safe then* $\langle(\delta_P\delta_F, Q_P \uplus Q_F, \sigma), c\rangle$ *is also safe*
(2) *If* $\langle(\delta_P, Q_P, \sigma), c\rangle$ *is safe and* $\langle(\delta_P\delta_F, Q_P \uplus Q_F, \sigma), c\rangle \rightsquigarrow^* \langle(\delta', Q', \sigma'), c'\rangle$ *then there exist* $\delta'_P \in (0,1]$ *and* Q'_P *such that:*
   (a) $\delta' = \delta'_P\delta_F$ *and* $Q' = Q'_P \uplus Q_F$
   (b) $\langle(\delta_P, Q_P, \sigma), c\rangle \rightsquigarrow^* \langle(\delta'_P, Q'_P, \sigma'), c'\rangle$.

Intuitively, the lemma says that if a config is safe in a small heap with the probability δ_P then it is also safe in a bigger heap with the joint probability $\delta_P\delta_F$. Furthermore, the steps made by the config with the bigger heap can be traced back to the steps made by the config with the small heap. We are now ready to prove the frame rule QFrame.

Proof of QFrame. Let $\delta, Q, \sigma \models P \circledast F$. Then there exist Q_P, Q_F and $\delta_P, \delta_F \in (0,1]$ such that:

(1) $Q_P \uplus Q_F = Q$ and $\delta_P\delta_F = \delta$
(2) $\delta_P, Q_P, \sigma \models P$ and $\delta_F, Q_F, \sigma \models F$

On the other hand, the triple $\{P\}c\{Q\}$ means that the config $\langle(\delta_P, Q_P, \sigma), c\rangle$ is safe and if $\langle(\delta_P, Q_P, \sigma), c\rangle \rightsquigarrow^* \langle(\delta', Q', \sigma'), \mathbf{skip}\rangle$ then $\delta', Q', \sigma' \models Q$.
By Lemma. 7.5, it follows that $\langle(\delta_P\delta_F, Q_P \uplus Q_F, \sigma), c\rangle$ is also safe. Let

$$\langle(\delta_P\delta_F, Q_P \uplus Q_F, \sigma), c\rangle \rightsquigarrow^* \langle(\delta', Q', \sigma'), \mathbf{skip}\rangle$$

Then there exist δ'_P and Q'_P such that:

(1) $\delta' = \delta'_P\delta_F$ and $Q' = Q'_P \uplus Q_F$
(2) $\langle(\delta_P, Q_P, \sigma), c\rangle \rightsquigarrow^* \langle(\delta'_P, Q'_P, \sigma'), \mathbf{skip}\rangle$

Hence $\delta'_P, Q'_P, \sigma' \models Q$. Note that the side condition $\mathrm{free}(F) \cap \mathrm{mod}(c) = \emptyset$ implies $[\![v]\!]_\sigma = [\![v]\!]_{\sigma'}$ for every variable v in F. Therefore $\delta_F, Q_F, \sigma' \models F$ and so $\delta'_P\delta_F, Q'_P \uplus Q_F, \sigma' \models Q \circledast F$. □

The other two rules $\overline{\text{Frame}}$ and FFrame can be derived from QFrame as follows.

Proof of PFrame. We apply the disjunctive rule over the two sub-proofs below.

(1) Case 1: $0 < e \leq 1$. We apply the QFrame rule where the frame F is $e \cdot |\mathbf{emp}\rangle$.

$$\dfrac{\dfrac{\{P\}c\{Q\} \quad \dfrac{\mathrm{mod}(c) \cap \mathrm{free}(e) = \emptyset}{\mathrm{mod}(c) \cap \mathrm{free}(e \cdot |\mathbf{emp}\rangle \wedge 0 < e \leq 1) = \emptyset}}{\{(e \cdot |\mathbf{emp}\rangle \wedge 0 < e \leq 1) \circledast P\}c\{(e \cdot |\mathbf{emp}\rangle \wedge 0 < e \leq 1) \circledast Q\}}\ \text{QFrame}}{\{e \cdot P \wedge 0 < e \leq 1\}c\{e \cdot Q \wedge 0 < e \leq 1\}}\ \mathbb{P}^I, \mathbb{P}^{\circledast}, \overline{\circledast^{\wedge}}, \circledast^I$$
$$\overline{\{e \cdot P \wedge 0 < e \leq 1\}c\{e \cdot Q\}}$$

(2) Case 2: $e \leq 0$ or $e > 1$. These cases are infeasible w.r.t. our semantics. And so we apply the rule Bot in which the precondition is a contradiction.

$$\frac{\overline{\{\mathbf{false}\}c\{e \cdot Q\}}\ \text{Bot}}{\{e \cdot P \wedge (e \leq 0 \vee e > 1)\}c\{e \cdot Q\}}\ \mathbb{P}^{\perp}$$

□

Proof of $\overline{\text{Frame}}$. We apply the QFrame rule where the frame F is $|\mathbf{emp}\rangle \wedge \overline{F}$.

$$\frac{\{P\}c\{Q\} \quad \dfrac{\text{mod}(c) \cap \text{free}(\overline{F}) = \emptyset}{\text{mod}(c) \cap \text{free}(|\mathbf{emp}\rangle \wedge \overline{F}) = \emptyset}}{\dfrac{\{P \circledast (|\mathbf{emp}\rangle \wedge \overline{F})\}c\{Q \circledast (|\mathbf{emp}\rangle \wedge \overline{F})\}}{\{P \wedge \overline{F}\}c\{Q \wedge \overline{F}\}}\ \overline{\circledast^{\wedge}, \circledast^{I}}}\ \text{QFrame}$$

□

7.5 Discussion on the Limitations of the Framework

In our step semantics (Fig. 18), the mixed states are split into pure states to enable local reasoning. Although such interpretation is consistent with the standard model [Selinger and Valiron 2008], we lose complete information about the probabilistic ensembles. Consequently, it is challenging to express the notion of *almost surely termination* (e.g. [Li and Ying 2017; Zhou et al. 2019]), which is defined based on the limit of the state distribution. Such limitation of the semantics also causes incompleteness for computing the weakest precondition and strongest postcondition, as well as the reasoning of probabilities. In particular, the postcondition of the measurement rule (Fig. 7) can only be in disjunctive form which is not expressive enough to specify general probabilistic distributions. For example, consider the following program where the two qubits $q^*[0]$ and $q^*[1]$ — each has the state $|+\rangle$ — are measured and then the results are added together:

$$v_0 := \mathbf{measure}(q^*[0]); v_1 := \mathbf{measure}(q^*[1]); v := v_0 + v_1;$$

The following postcondition is derived by our framework:

$$\left(v = 1 \wedge \frac{1}{4} \cdot q^*[0,1] \mapsto |01\rangle\right) \vee \left(v = 1 \wedge \frac{1}{4} \cdot q^*[0,1] \mapsto |10\rangle\right) \vee \ldots \tag{29}$$

However, the disjunctive form above is too weak for us to prove that $\mathbb{P}(v = 1) = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$. This boils down to the fact that our semantics treats the two outcomes above as two separate computation paths and thus their probabilities cannot be combined to infer the exact probability for pure predicate $v = 1$. As a result, the probability for pure predicates in §5 is of the weak form $\mathbb{P}(\overline{Q}) = np$ for some integer $n \geq 1$ — in this case $p = \frac{1}{4}$ and so $\mathbb{P}(v = 1) \in \{\frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\}$ — so that it can be interpreted consistently with our semantics for pure states tagged with probabilities and also with the standard semantics for mixed states with complete probabilistic distributions.

To overcome the above problems, the semantics needs to be revised. A promising direction is to come up with a structure for quantum states that can contain the entire state distribution while at the same time remains applicable for local reasoning.

8 RELATED WORKS

A comprehensive and complete framework of quantum Hoare logic was proposed by [Ying 2012], which models quantum states using the density operator. Nevertheless, the framework is short of treatments for local reasoning and classical variables. The work in [Feng and Ying 2021] integrates classical states into quantum states using positive-operator valued distribution that maps each

classical state to an appropriate quantum state. The work in [Zhou et al. 2021] enables local reasoning for quantum computation using a quantum interpretation of the Bunched Implications [O'Hearn and Pym 1999] for SL. Our framework improves from these works by supporting classical variables directly in the semantics and also the dynamic allocation/deallocation of quantum qubits.

Based on Ying's framework, the work in [Li and Ying 2017] reasons about almost surely termination of quantum programs via the synthesis of the linear ranking super-martingale, which can be reduced to a problem in Semi-Definite Programming. The work in [Ying 2019] introduces the notion of proof outline and auxiliary rules to support automatic reasoning for quantum programs. The work in [Kakutani 2009] extends the probabilistic Hoare logic in [Hartog 1999] to reason about quantum programs but the reasoning cannot be achieved locally.

On the other hand, various schemes of program reasoning besides normal Hoare logic have been recently applied into the realm of quantum computing, e.g. Quantum Relational Hoare Logics [Barthe et al. 2019; Li and Unruh 2021], predicate transformation [Chadha et al. 2006; D'hondt and Panangaden 2006; Feng et al. 2007], testing and debugging [Huang and Martonosi 2019; Li et al. 2020]. Entailment rules for quantum systems were also studied in [Baez and Stay 2010] from the general viewpoint of category theory where $P \vdash Q$ means that there exists a morphism from the quantum state P to the quantum state Q. For the mechanization of quantum reasoning, there are the works of [Liu et al. 2019] that mechanizes Ying's framework [Ying 2012] in Isabelle/HOL [Naraschewski and Nipkow 2020], and of [Shi et al. 2021] that formalizes the reasoning of quantum circuits in Coq [Bertot and Castéran 2013].

Our operational semantics §7 follows in the footsteps of the sequential semantics for SL in [Yang and O'Hearn 2002] with a minor probability twist. For the last few decades, SL [O'Hearn et al. 2001; Reynolds 2002; Yang and O'Hearn 2002] has become a prominent framework for the local reasoning of heap-manipulated programs. The assertion language of SL comes from the logic of Bunched Implication [O'Hearn and Pym 1999], a comprehensive framework to reason about resource ownership. For instance, memory cells are resources in SL whereas quantum qubits are resources in our framework.

9 CONCLUSION

In this work, we have proposed an inference framework for quantum computation infused with separation logic. Our framework directly supports classical variables and dynamic allocation/deallocation of quantum qubits. We apply our framework to verify various quantum programs including Deutsch–Jozsa's algorithm and Grover's algorithm. For future work, we plan to mechanize the framework and develop verification techniques to support automatic reasoning.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers and shepherds for their constructive comments, especially for the discussion on the probability reasoning and the construct of the $\mathbb{P}$–rule for entailment. This work is supported by project MOE2018- T2-1-068 and project MOET32020-0004, both are funded by the Singapore Ministry of Education.

REFERENCES

John C. Baez and Mike Stay. 2010. Physics, Topology, Logic and Computation: A Rosetta Stone. *Lect. Notes Phys.* 813 (2010), 95–172. https://doi.org/10.1007/978-3-642-12821-9_2

Gilles Barthe, Justin Hsu, Mingsheng Ying, Nengkun Yu, and Li Zhou. 2019. Relational Proofs for Quantum Programs. *Proc. ACM Program. Lang.* 4, POPL, Article 21 (Dec. 2019), 29 pages. https://doi.org/10.1145/3371089

Charles H. Bennett, Ethan Bernstein, Gilles Brassard, and Umesh Vazirani. 1997. Strengths and Weaknesses of Quantum Computing. *SIAM J. Comput.* 26, 5 (Oct. 1997), 1510–1523. https://doi.org/10.1137/S0097539796300933

Yves Bertot and Pierre Castéran. 2013. *Interactive Theorem Proving and Program Development: Coq'Art: The Calculus of Inductive Constructions*. Springer Science & Business Media. https://doi.org/10.1007/978-3-662-07964-5

Benjamin Bichsel, Maximilian Baader, Timon Gehr, and Martin Vechev. 2020. Silq: A High-Level Quantum Language with Safe Uncomputation and Intuitive Semantics. In *PLDI*. 286–300. https://doi.org/10.1145/3385412.3386007

C. Calcagno, P. W. O'Hearn, and H. Yang. 2007. Local Action and Abstract Separation Logic. In *LICS*. 366–378. https://doi.org/10.1109/LICS.2007.30

R. Chadha, P. Mateus, and A. Sernadas. 2006. Reasoning about Imperative Quantum Programs. *Electronic Notes in Theoretical Computer Science* 158 (2006), 19–39. https://doi.org/10.1016/j.entcs.2006.04.003

Andrew W. Cross, Lev S. Bishop, John A. Smolin, and Jay M. Gambetta. 2017. Open Quantum Assembly Language. https://arxiv.org/abs/1707.03429v2. arXiv:1707.03429 [quant-ph]

D Deutsch and R Jozsa. 1992. Rapid Solution of Problems by Quantum Computation. (1992). https://doi.org/10.1098/rspa.1992.0167

Ellie D'hondt and Prakash Panangaden. 2006. Quantum Weakest Preconditions. *Mathematical. Structures in Comp. Sci.* 16, 3 (June 2006), 429–451. https://doi.org/10.1017/S0960129506005251

Yuan Feng, Runyao Duan, Zhengfeng Ji, and Mingsheng Ying. 2007. Proof Rules for the Correctness of Quantum Programs. *Theoretical Computer Science* 386, 1 (2007), 151–166. https://doi.org/10.1016/j.tcs.2007.06.011

Yuan Feng and Mingsheng Ying. 2021. Quantum Hoare Logic with Classical Variables. arXiv:2008.06812 [cs.LO]

Google. 2018a. A Preview of Bristlecone, Google's New Quantum Processor. https://ai.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html.

Google. 2018b. Google Cirq. https://github.com/quantumlib/Cirq.

Alexander S. Green, Peter LeFanu Lumsdaine, Neil J. Ross, Peter Selinger, and Benoît Valiron. 2013. Quipper: A Scalable Quantum Programming Language. In *PLDI*. 333–342. https://doi.org/10.1145/2491956.2462177

Lov K. Grover. 1996. A Fast Quantum Mechanical Algorithm for Database Search. In *STOC*. 212–219. https://doi.org/10.1145/237814.237866

J. I. den Hartog. 1999. Verifying Probabilistic Programs Using a Hoare Like Logic. In *Proceedings of the 5th Asian Computing Science Conference on Advances in Computing Science*. 113–125. https://doi.org/10.1007/3-540-46674-6_11

C. A. R. Hoare. 1969. An Axiomatic Basis for Computer Programming. *Commun. ACM* 12, 10 (1969), 576–580. https://doi.org/10.1145/363235.363259

Yipeng Huang and Margaret Martonosi. 2019. Statistical Assertions for Validating Patterns and Finding Bugs in Quantum Programs. In *Proceedings of the 46th International Symposium on Computer Architecture*. Association for Computing Machinery, New York, NY, USA, 541–553. https://doi.org/10.1145/3307650.3322213

IBM. 2020. https://www.ibm.com/quantum-computing/.

Ali JavadiAbhari, Shruti Patil, Daniel Kudrow, Jeff Heckey, Alexey Lvov, Frederic T. Chong, and Margaret Martonosi. 2015. ScaffCC: Scalable Compilation and Analysis of Quantum Programs. *Parallel Comput.* 45 (2015), 2–17. https://doi.org/10.1145/2597917.2597939

Yoshihiko Kakutani. 2009. A Logic for Formal Verification of Quantum Programs. In *Advances in Computer Science - ASIAN 2009. Information Security and Privacy (Lecture Notes in Computer Science, Vol. 5913)*, Anupam Datta (Ed.). 79–93. https://doi.org/10.1007/978-3-642-10622-4_7

Xuan-Bach Le and Aquinas Hobor. 2018. Logical Reasoning for Disjoint Permissions. In *ESOPS (Lecture Notes in Computer Science, Vol. 10801)*. 385–414. https://doi.org/10.1007/978-3-319-89884-1_14

Gushu Li, Li Zhou, Nengkun Yu, Yufei Ding, Mingsheng Ying, and Yuan Xie. 2020. Projection-Based Runtime Assertions for Testing and Debugging Quantum Programs. *Proc. ACM Program. Lang.* 4, OOPSLA, Article 150 (2020), 29 pages. https://doi.org/10.1145/3428218

Yangjia Li and Dominique Unruh. 2021. Quantum Relational Hoare Logic with Expectations. In *ICALP (LIPIcs, Vol. 198)*. 136:1–136:20. https://doi.org/10.4230/LIPIcs.ICALP.2021.136

Yangjia Li and Mingsheng Ying. 2017. Algorithmic Analysis of Termination Problems for Quantum Programs. In *POPL*. 29 pages. https://doi.org/10.1145/3158123

Junyi Liu, Bohua Zhan, Shuling Wang, Shenggang Ying, Tao Liu, Yangjia Li, Mingsheng Ying, and Naijun Zhan. 2019. Formal Verification of Quantum Algorithms Using Quantum Hoare Logic. In *CAV*. 187–207. https://doi.org/10.1007/978-3-030-25543-5_12

Shusen Liu, Li Zhou, Ji Guan, Yang He, Runyao Duan, and Mingsheng Ying. 2017. $Q|SI\rangle$: A Quantum Programming Environment. *Scientia Sinica Informationis* 47, 10 (2017), 1300–1315. https://doi.org/10.1360/N112017-00095

X. Liu and J. Kubiatowicz. 2013. Chisel-Q: Designing Quantum Circuits with a Scala Embedded Language. In *ICCD*. 427–434. https://doi.org/10.1109/ICCD.2013.6657075

Wolfgang Naraschewski and Tobias Nipkow. 2020. Isabelle/HOL. http://www.cl.cam.ac.uk/research/hvg/Isabelle/

Peter O'Hearn, John Reynolds, and Hongseok Yang. 2001. Local Reasoning about Programs that Alter Data Structures. In *CSL*. 1–19. https://doi.org/10.1007/3-540-44802-0_1

Peter W. O'Hearn and David J. Pym. 1999. The Logic of Bunched Implications. *The Bulletin of Symbolic Logic* 5, 2 (1999), 215–244. https://doi.org/10.2307/421090

Jennifer Paykin, Robert Rand, and Steve Zdancewic. 2017. QWIRE: A Core Language for Quantum Circuits. In *POPL*. 846–858. https://doi.org/10.1145/3009837.3009894

J. C. Reynolds. 2002. Separation Logic: A Logic for Shared Mutable Data Structures. In *LICS*. 55–74. https://doi.org/10.1109/LICS.2002.1029817

Peter Selinger and Benoît Valiron. 2005. A lambda calculus for quantum computation with classical control. In *TLCA (Lecture Notes in Computer Science, Vol. 3461)*. 354–368. https://doi.org/10.1007/11417170_26

Peter Selinger and Benoît Valiron. 2008. On a Fully Abstract Model for a Quantum Linear Functional Language. *Electron. Notes Theor. Comput. Sci.* 210 (July 2008), 123–137. https://doi.org/10.1016/j.entcs.2008.04.022

Wenjun Shi, Qinxiang Cao, Yuxin Deng, Hanru Jiang, and Yuan Feng. 2021. Symbolic Reasoning about Quantum Circuits in Coq. arXiv:2005.11023 [cs.PL]

P. W. Shor. 1994. Algorithms for Quantum Computation: Discrete Logarithms and Factoring. In *FOCS*. 124–134. https://doi.org/10.1109/SFCS.1994.365700

Damian S. Steiger, Thomas Häner, and Matthias Troyer. 2018. ProjectQ: An Open Source Software Framework for Quantum Computing. *Quantum* 2 (2018), 49. https://doi.org/10.22331/q-2018-01-31-49

Krysta Svore, Alan Geller, Matthias Troyer, John Azariah, Christopher Granade, Bettina Heim, Vadym Kliuchnikov, Mariia Mykhailova, Andres Paz, and Martin Roetteler. 2018. Q#: Enabling Scalable Quantum Computing and Development with a High-Level DSL. In *RWDSL*. Article 7, 10 pages. https://doi.org/10.1145/3183895.3183901

Dave Wecker, Krysta M. Svore, and Krysta M. Svore. 2014. LIQUi|⟩: A Software Design Architecture and Domain-Specific Language for Quantum Computing. http://research.microsoft.com/en-us/projects/liquid/.

W. K. Wootters and W. H. Zurek. 1982. A Single Quantum Cannot Be Cloned. *Nature* 299, 5886 (28 Oct. 1982), 802–803. https://doi.org/10.1038/299802a0

Hongseok Yang and Peter O'Hearn. 2002. A Semantic Basis for Local Reasoning. In *FoSSaC*. 402–416. https://doi.org/10.1007/3-540-45931-6_28

Mingsheng Ying. 2012. Floyd–Hoare Logic for Quantum Programs. *ACM Trans. Program. Lang. Syst.* 33, 6, Article 19 (2012), 49 pages. https://doi.org/10.1145/2049706.2049708

Mingsheng Ying. 2019. Toward Automatic Verification of Quantum Programs. *Formal Aspects of Computing* 31 (2019), 3–25. https://doi.org/10.1007/s00165-018-0465-3

Li Zhou, Gilles Barthe, Justin Hsu, Mingsheng Ying, and Nengkun Yu. 2021. A Quantum Interpretation of Bunched Logic & Quantum Separation Logic. In *LICS*. 1–14. https://doi.org/10.1109/LICS52264.2021.9470673

Li Zhou, Nengkun Yu, and Mingsheng Ying. 2019. An Applied Quantum Hoare Logic. In *PLDI*. 1149–1162. https://doi.org/10.1145/3314221.3314584