We present a logic for reasoning about pairs of interactive quantum programs – quantum relational Hoare logic (qRHL). This logic follows the spirit of probabilistic relational Hoare logic (Barthe et al. 2009) and allows us to formulate how the outputs of two quantum programs relate given the relationship of their inputs. Probabilistic RHL was used extensively for computer-verified security proofs of classical cryptographic protocols. Since pRHL is not suitable for analyzing quantum cryptography, we present qRHL as a replacement, suitable for the security analysis of post-quantum cryptography and quantum protocols. The design of qRHL poses some challenges unique to the quantum setting, e.g., the definition of equality on quantum registers. Finally, we implemented a tool for verifying proofs in qRHL and developed several example security proofs in it.

CCS Concepts: • Theory of computation → Quantum computation theory; Hoare logic; Program verification;

Additional Key Words and Phrases: Quantum cryptography, Hoare logics, formal verification

ACM Reference Format:
Dominique Unruh. 2019. Quantum Relational Hoare Logic. Proc. ACM Program. Lang. 3, POPL, Article 33 (January 2019), 31 pages. https://doi.org/10.1145/3290346

1 INTRODUCTION

Handwritten security proofs of cryptographic protocols are inherently error prone – mistakes may and most likely will happen, and they can stay unnoticed for years. (The most striking examples are probably the OAEP construction [Bellare and Rogaway 1994] whose proof went through a number of fixes [Fujisaki et al. 2001, 2004; Shoup 2002] until it was finally formally proven in [Barthe et al. 2011b] after years of industrial use, and the PRF/PRP switching lemma which was a standard textbook example for many years before it was shown that the standard proof is flawed [Bellare and Rogaway 2006].) This undermines the purpose of said proofs. In order to enable high trust in our cryptographic protocols, computer-aided verification is the method of choice. Methods for the security analysis of protocols have a long history in the formal methods community, starting with the seminal paper by Dolev and Yao [Dolev and Yao 1983]. However, most methods were designed using so-called symbolic abstractions that abstract away from the details of the cryptographic algorithms by treating messages as terms with well-defined adversarial deduction rules. There is no formal justification for this (except in cases where “computational soundness” proofs exist, see [Abadi and Rogaway 2002] and follow-up work). To avoid this heuristic, or for reasoning about the basic cryptographic constructions themselves (e.g., when designing an encryption scheme), we need verification methods that allow us to reason about cryptographic protocols in a fine-grained way, treating computations as what they are: time-bounded probabilistic computations (we call this the “computational model”).

In recent years, a number of logics and tools have been presented for reasoning about cryptographic protocols in the computational model. For example, the CryptoVerif tool [Blanchet 2006;
CryptoVerif authors 2018] employs a heuristic search using special rewriting rules to simplify a protocol into a trivial one. The CertiCrypt tool [Barthe et al. 2009; CertiCrypt authors 2013] and its easier to use EasyCrypt successor [Barthe et al. 2014, 2011a] allow us to reason about pairs of programs (that represent executions of cryptographic protocols) using “probabilistic relational Hoare logic” (pRHL). Proofs in the EasyCrypt tool are developed manually, using backwards reasoning with a set of special tactics for reasoning about pRHL judgments.

All those tools only consider classical cryptography. In recent years, interest in quantum cryptography, both in research and in industry, has surged. Quantum cryptography is concerned with the development of protocols that are secure even against attacks using quantum computers, to ensure that we can securely use cryptography in the future (post-quantum cryptography). And quantum cryptography is concerned with the design of quantum protocols that make active use of quantum mechanics in their communication, such as quantum key distribution protocols (an area pioneered by [Bennett and Brassard 1984]). While these are two different concerns involving different technologies, there is a large overlap at least in the analysis and proof techniques.

Can we formalize quantum cryptographic proofs using the existing tools? For quantum protocols, the answer is clearly no, because the tools do not even allow us to encode the quantum operations performed in the protocol. Yet, when it comes to post-quantum security, the protocols themselves are purely classical and can be modeled in tools such as EasyCrypt, only the adversaries (which stay abstract anyway) cannot. Thus, even though existing tools were not designed with post-quantum security in mind, it is conceivable that the logics underlying those tools happen to be sound in the quantum setting. The logic might happen to only contain deduction rules that also happen to be valid with respect to quantum adversaries. Unfortunately, at least for the EasyCrypt tool, we show that this is not the case. We proved in EasyCrypt the security of a simple protocol (an execution of the CHSH game [Clauser et al. 1969]) which we know not to be secure in the quantum setting. (For more details, see below in Section 1.2.)

Thus the question arises: can we design tools for the verification of quantum cryptography? (Both for post-quantum security proofs, and for actual quantum protocols.) What underlying logics should we use? In this paper, we set out to answer that question.

Our main contribution is the development of a logic for relating quantum programs, quantum relational Hoare logic (qRHL). qRHL generalizes the probabilistic logic pRHL used by EasyCrypt and CertiCrypt, and allows us to reason about the relationship between quantum programs. We have implemented a theorem prover for qRHL programs, and have analyzed quantum teleportation and a simple post-quantum secure protocol with it, to demonstrate the applicability of qRHL to the formal verification of quantum cryptography.

We now describe the contributions in more detail. For this, we first review classical RHL. (Followed by an overview over quantum RHL, of our tools and experiments, related work, and the outline for the rest of the paper.)

1.1 Classical RHL

Deterministic RHL. The simplest case of RHL is deterministic RHL [Benton 2004]. Deterministic RHL allows us to relate two deterministic programs. In our context, a program is always a program in some imperative programming language, operating on a finite set of variables. Consider two programs $c_1$ and $c_2$, operating on variables $x, y, \ldots$. An RHL judgment is an expression $\{A\}c_1 \sim c_2\{B\}$ where $A$ and $B$ are Boolean predicates involving the variables from $c_1$ and $c_2$ (tagged with index 1 and 2, respectively). For example $A$ might be $x_1 = x_2$, and $B$ might be $x_1 > x_2$. Then $\{A\}c_1 \sim c_2\{B\}$ has the following meaning: If the initial memories $m_1, m_2$ of $c_1$ and $c_2$ satisfy the predicate $A$, then the memories $m_1', m_2'$ after execution of $c_1$ and $c_2$, respectively, satisfy $B$. 
Definition 1 (Deterministic RHL, informal). \( \{A\}c_1 \sim c_2 \{B\} \) holds iff: For any memories \( m_1, m_2 \) such that \( \llbracket A \rrbracket_{m_1, m_2} = \text{true} \), we have \( \llbracket B \rrbracket_{m_1', m_2'} = \text{true} \) where \( m_1' := \llbracket c_1 \rrbracket(m_1) \) and \( m_2' := \llbracket c_2 \rrbracket(m_2) \).

In this definition, a memory \( m_i \) is simply an assignment from variables \( x, y, \ldots \) to their contents. And \( \llbracket A \rrbracket_{m_1, m_2} \) is the evaluation of the predicate \( A \) with the contents of memory \( m_1 \) assigned to the variables \( x_1, y_1, \ldots \) and the contents of \( m_2 \) to variables \( x_2, y_2, \ldots \). (That is, in a predicate \( A \), we add indices 1, 2 to the variables \( x, y \) to distinguish the variables of \( c_1 \) and \( c_2 \).) And \( \llbracket c_1 \rrbracket(m_1) \) represents the content of the memory after running \( c_1 \) on initial memory \( m_1 \). (That is, \( \llbracket c_1 \rrbracket \) is the denotational semantics of a deterministic programming language.)

For example, \( \{x_1 = x_2 \} x \leftarrow x + 1 \sim x \leftarrow x - 1 \{x_1 > x_2 \} \) says that if the variable \( x \) has the same value in memory \( m_1 \) and \( m_2 \), and we increase \( x \) in \( m_1 \) and decrease \( x \) in \( m_2 \) (resulting in memories \( m_1', m_2' \), respectively), then \( x \) in \( m_1' \) is greater than \( x \) in \( m_2' \).

Given this definition of RHL, we can then prove a suitable collection of sound rules such as, e.g.,

\[
\frac{\{A\} \Rightarrow \{A'\}c_1 \sim c_2 \{B\}}{\{A\}c_1 \sim c_2 \{B\}}.
\]

These rules can then be used to prove complex relationships between deterministic programs.

Probabilistic RHL. Deterministic RHL as described in the previous paragraph can reason only about deterministic programs. At first glance, it may seem that it is easy to generalize RHL to the probabilistic case (pRHL). However, there are some subtleties in the definition of pRHL. Consider the program \( c \) that assigns a uniformly random bit to \( x \). In this case, is the following pRHL judgment \( \text{true?} \{ \text{true} \} c \sim c \{ x_1 = x_2 \} \). That is, if we have two programs that pick a random bit, should pRHL say that after the execution, the bit is the same? At the first glance, one might expect that the answer should be “no” because \( x_1 \) and \( x_2 \) are chosen independently and thus equal only with probability \( \frac{1}{2} \). However, if we define pRHL that way, we cannot express the fact that the two programs \( c \) do the same. We might argue that \( x_1 = x_2 \) should just mean that \( x \) is chosen according to the same distribution in both programs. The formalization of pRHL from Barthe et al. 2009 takes exactly this approach. They define pRHL as follows:

Definition 2 (Probabilistic RHL, informal). \( \{A\}c_1 \sim c_2 \{B\} \) holds iff: For any memories \( m_1, m_2 \) such that \( \llbracket A \rrbracket_{m_1, m_2} = \text{true} \), there exists a distribution \( \mu' \) on pairs of memories such that \( \llbracket B \rrbracket_{m_1', m_2'} = \text{true} \) for all \( (m_1', m_2') \in \text{supp} \mu' \), and \( \llbracket c_1 \rrbracket(m_1) = \mu_1' \) and \( \llbracket c_2 \rrbracket(m_2) = \mu_2' \).

Here \( \text{supp} \mu' \) is the support of \( \mu' \), i.e., the set of all \( (m_1', m_2') \) that have non-zero probability according to \( \mu' \). And \( \llbracket c_1 \rrbracket(m_1) \) is the probability distribution of the memory after executing \( c_1 \) with initial memory \( m_1 \). And \( \mu_1' \) and \( \mu_2' \) refer to the marginal distributions of \( \mu' \) that return the first and second memory, respectively.

What does this definition mean, intuitively? It requires that there is some hypothetical joint distribution \( \mu' \) on pairs of memories (that satisfy the postcondition \( B \)) such that the final memories of \( c_1 \) and \( c_2 \) are distributed according to the marginals of \( \mu' \). In other words, it requires that there is some way of choosing the randomness of \( c_1 \) and \( c_2 \) in a joint way such that \( B \) becomes true. For example, when \( c \) picks a uniformly random bit \( x \), we can chose \( \mu' \) as the uniform distribution on \( \{(0, 0), (1, 1)\} \) (where we write 0 and 1 for the memory that assigns 0 or 1, respectively, to \( x \)). Then the two marginal distributions are the uniform distribution, thus \( \mu_1' = \llbracket c \rrbracket(m_1) \) and \( \mu_2' = \llbracket c \rrbracket(m_2) \), and we have that all \( (m_1, m_2) \in \text{supp} \mu' \) satisfy \( B := (x_1 = x_2) \). This implies that \( \{ \text{true}\} c \sim c \{ x_1 = x_2 \} \). A curiosity is that the following judgment is also true: \( \{ \text{true}\} c \sim c \{ x_1 \neq x_2 \} \) (choose \( \mu' \) as the uniform distribution on \( \{(0, 1), (1, 0)\} \)). In particular, the following rule does not hold for pRHL (but it does hold for deterministic RHL):

\[
\frac{\{A\}c_1 \sim c_2 \{B_1\} \quad \{A\}c_1 \sim c_2 \{B_2\}}{\{A\}c_1 \sim c_2 \{B_1 \land B_2\}}.
\]

Reference Barthe et al. 2009 derives a number of useful rules for pRHL. Using these rules, one can derive complex relationships.
between probabilistic programs. For example, the successful EasyCrypt tool [Barthe et al. 2014, 2011a] uses these to prove the security of various cryptographic schemes.

pRHL is well-suited for formalizing cryptographic security proofs because those proofs usually proceed by transforming the initial protocol (formulated as a program, called a “game”) in a sequence of small steps to a trivial protocol, and the relationship between any two of those games can be analyzed using pRHL.

1.2 Quantum RHL

The need for a new logic. To illustrate that a new logic is needed, even if we consider only classical protocols (but quantum adversaries, i.e., post-quantum cryptography), we analyzed the CHSH game [Clauser et al. 1969; Cleve et al. 2004] in EasyCrypt. In this game, two potentially malicious parties Alice and Bob may share a state but are not allowed to communicate. Alice and Bob get uniformly random bits $x, y$, respectively, and have to respond with bits $a, b$, respectively. They win if $a \oplus b = x \cdot y$. In the classical setting, the probability of winning is at most $\frac{3}{4}$, while in the quantum setting, it is known that the maximum winning probability is $\cos^2(\pi/8) \approx 0.85$ [Cirel’son 1980; Cleve et al. 2004]. The CHSH game can be seen as a particularly simple case of a multi-prover proof system, see [Cleve et al. 2004].

In EasyCrypt, we show that no adversary (consisting of Alice and Bob) can win the CHSH game with probability greater than $\frac{3}{4}$. Since a quantum adversary would be able to get a larger success probability, this shows that the logic employed by EasyCrypt indeed assumes that adversaries are classical. And if we use EasyCrypt to reason about quantum adversaries, it will give unsound results.

We stress that this is not a flaw in EasyCrypt. It was never claimed that EasyCrypt models quantum adversaries. However, since the logic used by EasyCrypt is not known to be complete, it could have been that all rules hold “by accident” also in the quantum setting. The CHSH example shows that this is not the case. The code of the EasyCrypt development can be found at [Unruh 2018a].

In addition, we performed a similar proof (also of the the CHSH game) in the CryptHOL framework [Basin et al. 2017], with analogous results. The resulting Isabelle theory can be found at [Unruh 2018b].

We would like to address one common fallacy: to prove post-quantum security, we could just use existing tools (and proofs) and strengthen them simply by limiting the hardness assumptions to quantum-safe hardness assumptions (e.g., we do not use the hardness of factoring numbers, but, for example, the hardness of certain lattice problems as our underlying assumption). Unfortunately, this does not work. The underlying argument “if $X$ is proven secure classically based on assumption $Y$, and $Y$ is quantum-safe, then $X$ is post-quantum secure” is a common fallacy. Examples of protocols with classical security proofs that are insecure against quantum adversaries, even when instantiated with quantum hard assumptions, can be found in [Ambainis et al. 2014]. Also, the CHSH game above constitutes a degenerate counterexample to this fallacy: It can be seen as a multiplayer cryptographic protocol\(^1\) that needs no assumptions, thus “hardening” the assumptions would not change anything, so the above fallacy would tell us that CHSH itself is quantum-secure. Which it is not.

Quantum programs. In order to develop a quantum variant of pRHL, we first need to fix the language used for describing quantum programs. We make the following design choices: Our language is a simple imperative programming language (similar to the language pWhile underlying

\(^1\)Where security means that the adversary cannot win with probability greater $3/4$.  

Proc. ACM Program. Lang., Vol. 3, No. POPL, Article 33. Publication date: January 2019.
Quantum Relational Hoare Logic 33:5

pRHL (with both classical and quantum variables. We model only classical control (i.e., conditions in if- and while-statements depend only on classical variables). This is sufficient for our use case of modeling quantum cryptography, since protocols in that field are typically described by semiformal pseudocode with classical control, and with mixed classical and quantum operations. The language supports classical random sampling, application of unitaries, projective measurements. For simplicity, we do not model procedure calls. However, we expect that adding procedures would not lead to any difficulties. Since we do not include any advanced language features, the semantics of the language are easily defined: The denotational semantics $[c]$ of a quantum program $c$ is described by a function that maps quantum states (the state of the memory before execution) to quantum states (the state of the memory after execution). Here quantum states are described by density operators (mixed states) since the language contains probabilistic elements (random sampling, measurement outcomes); density operators intuitively model probability distributions of quantum states. Syntax and semantics of the language are presented in Section 3.

Quantum RHL – predicates. To extend pRHL to the quantum setting, we need to give meaning to judgments of the form $\{A\}c \sim d\{B\}$ where $c, d$ are quantum programs. For this, we need to answer two questions: How are the predicates (pre- and post-conditions) $A, B$ formalized (i.e., what mathematical structure describes $A, B$)? And what is the semantics of the overall qRHL judgment $\{A\}c \sim d\{B\}$?

In classical RHL (both the deterministic and pRHL), predicates are Boolean predicates on pairs of memories $m_1, m_2$ (cf. the notation introduced after Definition 1). Equivalently, we can see a predicate $A$ as a set of pairs of memories, and a distribution $\mu'$ over pairs of memories (cf. Definition 2) satisfies $A$ iff $\text{supp} \mu' \subseteq A$. For formalizing quantum predicates $A$, there are two natural choices:

(a) $A$ could be represented as a set of density operators (mixed states)$^2$ (possibly subject to some natural closure properties, e.g., closure under multiplication with a scalar). And then $\rho$ satisfies $A$ iff $\rho \in A$.

(b) $A$ could be represented as a set of pure states$^3$ (possibly subject to some natural closure properties, e.g., a subspace of the Hilbert space of pure quantum states). Then $\rho$ satisfies $A$ iff $\text{supp} \rho \subseteq A$.

Here $\text{supp} \rho$ is the support of $\rho$. (Intuitively, $\rho$ is a probability distribution over states in $\text{supp} \rho$.

For a formal definition see the preliminaries.)

Obviously, approach (a) is the more general approach. However, in this work, we will model predicates as subspaces of pure states, following approach (b). The reason for this is that we would like two properties to be satisfied:

- If $p_1$ and $p_2$ both satisfy $A$, then $p_1 \rho_1 + p_2 \rho_2$ satisfies $A$.

  For example, consider a program $c$ that runs $c_1$ with probability $p_1$, and $c_2$ with probability $p_2$. Assume that $c_1$ and $c_2$ both have final states $\rho_1, \rho_2$ that satisfy $A$. Then we would expect that $c$ also has a final state that satisfies $A$. This is guaranteed if $p_1 \rho_1 + p_2 \rho_2$ satisfies $A$.

- If $p_1 \rho_1 + p_2 \rho_2$ satisfies $A$ (for $p_1, p_2 > 0$), then $\rho_1$ and $\rho_2$ both satisfy $A$.

  For example, consider a program if $b$ then $c$ else $d$, and an initial state $\rho = p_1 \rho_1 + p_2 \rho_2$ where $\rho_1$ satisfies the condition $b$, and $\rho_2$ satisfies $\neg b$. Let $A$ be some predicate that $\rho$ satisfies. (E.g., the $\rho_1, \rho_2$ could differ in the value of a classical variable that the predicate $A$ does not even talk about.)

  Then we would expect that after branching depending on $b$, at the beginning of the execution of $c$ and $d$, respectively, $A$ is still satisfied. That is, $p_1 \rho_1$ and $p_2 \rho_2$ still satisfy $A$.

$^2$Density operators are a formalism for describing quantum states where a quantum state is represented by a positive Hermitian operator. This formalism allows us to model probabilistic mixtures of quantum states (roughly speaking, probability distributions over quantum states). See, e.g., [Nielsen and Chuang 2010, Chapter 2.4].

$^3$Pure states are states that are represented as vectors in a Hilbert space. This more elementary formalism cannot represent probability distributions. Pure states can be seen as special cases of density operators (of rank 1). See, e.g., [Nielsen and Chuang 2010, Chapter 2.2].
The only sets that satisfy both conditions are those represented by a subspace (at least in the finite dimensional case, see the full version [Unruh 2018c] for a proof).

That is, a predicate $A$ is a subspace of the Hilbert space of quantum states, and each $\psi \in A$ is a pure quantum state over two memories (i.e., a superposition of states $|m_1, m_2\rangle$ where $m_1, m_2$ are classical memories that assign classical values to all variables of $c$ and $d$, respectively).

The precise definition of predicates will look somewhat different because of a special treatment of classical variables in our predicates. This is, however, just for notational convenience and will be explained in Section 4.

Note that our reasoning above introduced one implicit assumption: Namely that we want to model predicates for which “$\rho$ satisfies $A$” is either true or false. Instead, it would be possible to have predicates that are fulfilled to a certain amount (say between 0 and 1). For example, the quantum predicates modeled by D’Hondt and Panangaden [D’Hondt and Panangaden 2006] are Hermitian operators, and a state $\rho$ satisfies a predicate $A$ to the amount $\text{tr}\rho$. Hoare judgments then encode the fact that the postcondition is satisfied at least as much as the precondition. We have opted not to pursue this approach in this paper (in particular because the resulting predicates are harder to understand, and we would also deviate more strongly from the design choices made in pRHL and EasyCrypt), but it would be interesting future work to explore such predicates in the relational setting.

To actually work with predicates, we introduce some syntactic sugar. For example, $\emptyset\{b\}$ is the space of all states where the classical variables satisfy the predicate $b$. (E.g., $\emptyset\{x_1 = x_2 + 1\}$ relates two classical variables.) This allows us to express all classical predicates with minimal overhead.

**Quantum RHL – judgments.** We now explain our definition of qRHL judgments. The first attempt would be to generalize the classical definition (Definition 2) to the quantum case by replacing classical objects by their quantum counterparts. For example, the natural quantum analogue for a probability distribution over some set $X$ is a density operator on a Hilbert space with basis $X$. To get this analogue, we first restate Definition 2 in an equivalent way:

**Definition 3 (Probabilistic RHL, informal).** $\{A\}c_1 \sim c_2\{B\}$ holds iff: For any distribution $\mu$ on pairs of memories with $\text{supp} \mu \subseteq A$, there exists a distribution $\mu'$ on pairs of memories with $\text{supp} \mu' \subseteq B$ such that $\mu'_1 = \llbracket c_1 \rrbracket(\mu_1)$ and $\mu'_2 = \llbracket c_2 \rrbracket(\mu_2)$.

Here – in slight abuse of notation – we consider the predicate $A$ as the set of all pairs of memories that satisfy this predicate. And $\mu_1, \mu_2, \mu'_1, \mu'_2$ are the first and second marginals of the distributions $\mu$ and $\mu'$, respectively. And we lift the denotational semantics $\llbracket \cdot \rrbracket$ to distributions in a natural way: $\llbracket c_1 \rrbracket(\mu_1)$ is the distribution of the memory after the execution of $c_1$ if the memory was distributed according to $\mu_1$ before the distribution.

In comparison with Definition 2, instead of deterministic initial memories $m_1, m_2$, we allow distributions $\mu$ of pairs of initial memories. It is easy to see that this definition is equivalent to Definition 2. However, the new definition has the advantage of being more symmetric (we use distributions both for the input and the output), and that it is more obvious how to transfer it to the quantum case. If we restate this definition in the quantum case in the natural way, we get:

**Definition 4 (Quantum RHL, first try, informal).** Let $A, B$ be quantum predicates (represented as subspaces, see above).

$\{A\}c_1 \sim c_2\{B\}$,nonsep holds iff: For any density operator $\rho$ over pairs of memories with $\text{supp} \rho \subseteq A$, there exists a density operator $\rho'$ over pairs of memories with $\text{supp} \rho' \subseteq B$ such that $\rho'_1 = \llbracket c_1 \rrbracket(\rho_1)$ and $\rho'_2 = \llbracket c_2 \rrbracket(\rho_2)$.

Here we replace the distribution $\mu$ by a density operator $\rho$. That is, we consider a quantum system where we have two memories, and the system can be in arbitrary superpositions and probabilistic
mixtures of classical assignments to the variables in these memories. (Formally, a memory is a function mapping variables to values, and we consider a quantum system with basis $|m_1, m_2\rangle$, where $m_1, m_2$ range over all such functions.)

Then $\rho_1$ denotes the density operator resulting from tracing out the second subsystem (i.e., the result of “destroying” the second memory), and analogously for $\rho_2, \rho'_1, \rho'_2$.

And $\text{supp} \rho$ is the support of $\rho$, i.e., it contains all pure states that the mixed state $\rho$ is a mixture of. Note that a state $\rho$ with marginals $\rho_1, \rho_2$ is called a quantum coupling of $\rho_1, \rho_2$ (see, e.g., [Zhou et al. 2018]). So the definition could be rephrased compactly as: “If the initial states of $c, d$ have a quantum coupling with support in $A$, then the final states have a quantum coupling with support in $B$.” Similarly, our other variants of qRHL can be straightforwardly stated in terms of quantum couplings.

At first glance, Definition 4 seems to be a reasonable analogue of pRHL in the quantum setting. At least, it seems to convey the same intuition. However, when we try to derive a suitable set of rules for qRHL judgments, we run into trouble. For example, we were not able to prove the following frame rule:

\[
\text{FRAME (simplified)} \quad \{A\}c \sim d\{B\}_{\text{nonsep}} \quad \text{variables in } R \text{ are disjoint from variables in } c, d, A, B
\]

\[
\{A \cap R\}c \sim d\{B \cap R\}_{\text{nonsep}}
\]

This rule is crucial for compositional reasoning: It tells us that any predicate that is independent of the programs under consideration (e.g., one that is only important for a program executed before or after) will be preserved. (Note that we were not able to prove that the frame rule does not hold for Definition 4, yet we conjecture that it does not.)

There are a number of natural variations of Definition 4 that can be tried, but for most of them, we either failed to prove the frame rule, or we run into problems when trying to prove rules involving predicates that state the equality of quantum variables (see below). We discuss the various unsuccessful approaches in the full version [Unruh 2018c].

The approach that turned out to work (in the sense that we can derive a useful set of rules), is to use the Definition 4, with the only difference that we quantify over separable states $\rho$ and $\rho'$ only. That is, we only consider states where the memories of the two programs are not entangled. (We stress that we do not restrict the entanglement within the programs, only between the programs. In particular, there are no restrictions on algorithms that use entanglement.) That is, the definition of qRHL is:

**Definition 5 (Quantum RHL, informal).** Let $A, B$ be quantum predicates (represented as subspaces, see above).

$\{A\}c_1 \sim c_2\{B\}$ holds iff: For any separable density operator $\rho$ over pairs of memories with $\text{supp} \rho \subseteq A$, there exists a separable density operator $\rho'$ over pairs of memories with $\text{supp} \rho' \subseteq B$ such that $\rho'_1 = \llbracket c_1 \rrbracket(\rho_1)$ and $\rho'_2 = \llbracket c_2 \rrbracket(\rho_2)$.

Note that the only difference to Definition 4 is the occurrence of the word “separable” (twice).

With this definition (stated formally in Section 5, Definition 30), we get a suitable set of rules for reasoning about pairs of programs (frame rule, rules for predicates involving equality, rules for reasoning about the different statements in quantum programs, case distinction, sequential composition, etc.)

**Relationship to classical pRHL.** By considering programs that contain no quantum instructions or quantum variables, and by considering only pre-/postconditions of the form $\text{Cla}[a]$, we get a definition of a classical relational Hoare logic as a special case. As it turns out, all three variants of
qRHL (Definition 4, Definition 5, and a third approach discussed in the full version [Unruh 2018c]) specialize to the same variant of pRHL, namely the original pRHL (as in Definition 2).

On coupling proofs. [Barthe et al. 2015] and [Barthe et al. 2017] elaborate on the connection between pRHL and so-called probabilistic couplings. It is often asked whether couplings can also be used as the basis for qRHL. The approach of [Barthe et al. 2015] uses the fact that pRHL by definition guarantees that there exists a probabilistic coupling for the output distributions of the programs that are related via a pRHL judgment, thus pRHL can be used to derive the existence of couplings with certain properties. This observation carries over immediately to qRHL because qRHL by definition implies the existence of quantum couplings. (In fact, the use cases from [Barthe et al. 2015] can be expressed in qRHL as easily. An interesting question is, of course, whether interesting purely quantum use cases can be found where this method is fruitful.) So in that sense, the answer is “yes”, qRHL is based on couplings.

On the other hand [Barthe et al. 2017] defines a new logic in which an explicit product program is constructed that produces the coupling. (This is then shown to be equivalent to the original pRHL, but having explicit access to this program has various advantages detailed in [Barthe et al. 2017].) Unfortunately, this approach most likely does not carry over to the quantum setting. In fact, one of the qRHL variants that we explored is the quantum equivalent of this approach. We show that this approach is incompatible with the notion of a quantum equality (see below). Details are given in the full version [Unruh 2018c].

Quantum equality. Using the approach from the previous paragraphs, we have a working definition of qRHL judgments, and a formalism for describing pre- and postconditions as subspaces. We can express arbitrary conditions on classical variables (in particular, we can express that \( x_1 = x_2 \) for variables \( x_1, x_2 \) of the left and right program, respectively). However, we cannot yet conveniently relate quantum variables in the left and the right program. That is, we cannot express judgments such as, for example:

\[
\{ q_1 \equiv_{\text{quant}} q_2 \} c \sim \text{skip} \{ q'_1 \equiv_{\text{quant}} q_2 \}
\]

where \( c \) swaps \( q_1 \) and \( q'_1 \). Here \( \equiv_{\text{quant}} \) denotes some (yet to be formalized) equality notion on quantum states.

Note that even if our goal is to use qRHL only for security proofs in post-quantum cryptography we need a notion of quantum equality: In this setting, all protocols contain only classical variables, but the adversary will contain quantum variables. In this context, typical predicates (almost ubiquitous in classical EasyCrypt proofs) are of the form: \( A \land y_1 = y_2 \) where \( y_1, y_2 \) refers to the global state of the adversary in the left/right program, and \( A \) is some predicate on program variables. In the post-quantum setting, \( y_1, y_2 \) will be quantum variables, and we need to express quantum equality.

So, how can quantum equality be defined? In other words, what is the formal meaning of \( q_1 \equiv_{\text{quant}} q_2 \) as a quantum predicate where \( q_1, q_2 \) are quantum variables (or, in the general case, sequences of quantum variables)? If we do not limit ourselves to predicates that are represented by subspaces (i.e., if a predicate can be represented as an arbitrary set of density operators), several natural equality notions exist:

(a) \( \rho \) satisfies \( q_1 \equiv_{\text{quant}} q_2 \) iff \( \rho_{q_1} = \rho_{q_2} \) where \( \rho_{q_1}, \rho_{q_2} \) are the density operators resulting from tracing out all registers except \( q_1, q_2 \), respectively.

(b) \( \rho \) satisfies \( q_1 \equiv_{\text{quant}} q_2 \) iff \( \rho \) is invariant under swapping \( q_1 \) and \( q_2 \). That is \( \rho = U \rho U^\dagger \) where \( U \) is the unitary that swaps \( q_1 \) and \( q_2 \).

(c) \( \rho \) satisfies \( q_1 \equiv_{\text{quant}} q_2 \) iff \( \rho \) is a mixture of states that are fixed under swapping \( q_1 \) and \( q_2 \). That is, for all \( \psi \in \text{supp} \rho \), \( U \psi = \psi \).

Proc. ACM Program. Lang., Vol. 3, No. POPL, Article 33. Publication date: January 2019.
Out of these three, only (c) is a predicate in our sense (i.e., described by a subspace). In fact, we conjecture that (c) is the only notion of equality of quantum variables that can be represented as a subspace and that has the following property (we give partial formal evidence for this in the full version [Unruh 2018c]): \[ \{ \textsf{Cla}[X_1 = X_2] \cap (Q_1 \equiv_{\text{quant}} Q_2) \} c \sim c \{ \textsf{Cla}[X_1 = X_2] \cap (Q_1 \equiv_{\text{quant}} Q_2) \} \]

Here \( X, Q \) are the classical and quantum variables of \( c \), respectively. (And \( X_1, X_2, Q_1, Q_2 \) are those variables indexed with 1 and 2, respectively). This rule says that, if all variables of \( c \) on the left and right are the same before running \( c \) on both sides, then they are the same afterwards, too. This is an essential property a notion of equality should satisfy. Thus we get the following notion of quantum equality (formulated slightly more generally than in (c)):

**Definition 6 (Quantum equality, informal).** The predicate \( (q_1^{(1)} q_2^{(1)} \ldots \equiv_{\text{quant}} q_2^{(2)} q_1^{(2)} \ldots) \) is the subspace \( \{ \psi : U \psi = \psi \} \) where \( U \) is the unitary that swaps registers \( q_1^{(1)} q_2^{(2)} \ldots \) with the registers \( q_2^{(2)} q_1^{(1)} \ldots \).

An example for a state satisfying \( q_1 \equiv_{\text{quant}} q_2 \) would be \( \psi_1 \otimes \psi_2 \otimes \phi \) where \( \psi_1 \) and \( \psi_2 \) are the same state on \( q_1 \) and \( q_2 \), respectively.

This quantum equality allows us to express, for example, relations of the form (“some predicate on classical variables” \( \cap q_1 \equiv_{\text{quant}} q_2 \)) which will be enough for many proofs in post-quantum cryptography. (We give an example of such a proof in Section 6.3.) More generally, however, we might wish for more expressivity. For example, we might wish to express that if \( q_1 \equiv_{\text{quant}} q_2 \), and then we apply the Hadamard \( H \) to \( q_1 \), then afterwards \( q_1 \) equals “\( H \) applied to \( q_2 \).” We generalize the definition of \( \equiv_{\text{quant}} \) to cover such cases, too. Intuitively, \( (U_1 q_1^{(1)} q_2^{(1)} \ldots \equiv_{\text{quant}} U_2 q_1^{(2)} q_2^{(2)} \ldots) \) means that the variables \( q_1^{(1)} q_2^{(2)} \ldots \), when we apply the unitary \( U_1 \) to them, equal the variables \( q_2^{(2)} q_1^{(1)} \ldots \) when we apply the unitary \( U_2 \) to them. We defer the details of the definition to Section 4.3. We only mention that, for example, \( U_1^{-1} \psi_1 \otimes U_2^{-1} \psi_2 \otimes \phi \) satisfies the predicate \( U_1 q_1 \equiv_{\text{quant}} U_2 q_2 \).

**Incompleteness.** While we prove the soundness of all rules in this paper, we do not strive to achieve completeness. (Incompleteness of pRHL was noted in [Barthe et al. 2017] and carries over to our setting.) In fact, we cannot expect completeness (except for restricted classes of programs) since equivalence even of programs is not semidecidable.

### 1.3 Tool & Experiments

We develop an interactive theorem prover for reasoning about qRHL. The tool consists of three main components: a ProofGeneral [The PG dev team 2018] frontend, the core tool written in Scala, and an Isabelle/HOL [Nipkow et al. 2002] backend for solving subgoals in the ambient logic (i.e., subgoals that do not contain qRHL judgments). The ProofGeneral frontend eases the interactive development of proofs. The Isabelle backend can also be accessed directly (via accompanying Isabelle theory files) to prove verification conditions that are beyond the power of our core tool.

We stress that although we use Isabelle/HOL as a backend, this does not mean that our tool is an LCF-style theorem prover (i.e., one that breaks down all proofs to elementary mathematical proof steps). All tactics in the tool, and many of the simplification rules in Isabelle are axiomatized (and backed by the proofs in this paper).

We developed a number of examples in our tool:

- A security proof that the encryption scheme defined by \( \text{enc}(k, m) := m \oplus G(k) \) is ROR-OT-CPA secure.\(^4\) (Where \( G \) is a pseudorandom generator.)

\(^4\) IND-OT-CPA is indistinguishability under chosen plaintext attacks (e.g., [Katz and Lindell 2014, Def. 3.22]) for one-time encryption. ROR-OT-CPA is a different formulation of the same property.
A security proof that the same encryption scheme is IND-OT-CPA secure. These two proofs are typical examples of reasoning in post-quantum security where quantum variables occur only inside the adversary, and where we maintain the invariant $Q_1 \equiv_{\text{quant}} Q_2$ where $Q_1, Q_2$ are all quantum variables.

A simple example illustrating the interplay of quantum equality and the applications of unitaries.

A proof that quantum teleportation works. That is, we show $\{q_1 = q_2\} \text{teleport} \sim \text{skip}\{q'_1 = q_2\}$ where $\text{teleport}$ is the program that teleports a qubit from $q$ to $q'$. This proof involves reasoning about quantum equality, measurements, unitary operations, initializations, as well as interaction between our tool and Isabelle/HOL for non-trivial subgoals.

The source code is published on GitHub [Unruh 2018a], a binary distribution is available at [Unruh 2018b].

On advanced cryptographic proofs. The simple examples presented above are, of course, only preliminary indications of the usability of qRHL for quantum cryptography. The study of complex quantum protocols (e.g., quantum key distribution) is out of the scope of this first work. For post-quantum cryptography (i.e., the security of classical protocols against quantum adversaries), our examples studying an encryption scheme indicate that derivations of qRHL judgments involving classical programs (but quantum adversaries) are very similar to those that would be performed in EasyCrypt (except, of course, that the EasyCrypt tool is more mature). See the example in Section 6.3. An exception to this are post-quantum security proofs which need to explicitly reason about quantum properties (even in the pen-and-paper proof). Typically, these are proofs involving rewinding (e.g., [Unruh 2012; Watrous 2009]) or quantum random oracles (e.g., [Boneh et al. 2011; Unruh 2015a; Zhandry 2012b]). As far as we know, proofs involving rewinding have not even been modeled classically in EasyCrypt, so it would be premature to speculate on how difficult these would be in qRHL. Proof involving the quantum random oracle model are currently very important, in particular in light of the NIST competition for post-quantum cryptography. Quantum random oracles are challenging because the adversary has superposition access to the oracle. **Modeling** the quantum random oracle is easy in our setting, the challenge lies in reasoning about it. So-called history free reductions [Boneh et al. 2011] (in which the random oracle is replaced by a differently chosen but equally distributed function) should be relatively easy, because in those cases the distribution of the random oracle after replacement is the same as before, so no quantum-specific reasoning is needed. (We checked this by implementing a very small such proof in our tool, \texttt{random-oracle.qrhl}, where the adversary cannot distinguish between those two cases given one query to the random oracle.) However, most post-quantum security proofs in the quantum random oracle model need more advanced techniques, such as, e.g., semi-constant distributions [Zhandry 2012b], one-way-to-hiding lemmas [Unruh 2015b], oracle indistinguishability [Zhandry 2012a], etc. Whether (and how) those techniques can be incorporated into qRHL-based proofs, or whether the logic will need to be extended to accommodate them, will be part of our future work.

1.4 Related Work

Frameworks for verification of cryptographic protocols (in the so-called computational model) include CryptoVerif [Blanchet 2006], CertiCrypt [Barthe et al. 2009], EasyCrypt [Barthe et al. 2011a].

\footnote{Note that the tool currently (version 0.3) only supports the rules \texttt{SEQ, EQUAL, CONSEQ, QRHELIMEq, SKIP, ASSIGN1/2, SAMPLE1/2, JOINTSAMPLE, QAPPLY1/2, QINIT1/2, MEASURE1/2}.

If $h$ is a classical variable of type $X \rightarrow Y$, then a random oracle can modeled by the program $h \leftarrow \text{uniform}$ (where the expression \texttt{uniform} is the uniform distribution on $X \rightarrow Y$) for initialization, and \texttt{apply} $x, y$ to $U_h$ (where $U_f$ is the unitary $|x, y \rangle \rightarrow |x, y @ f(x)\rangle$) for queries.}
FCF [Petcher and Morrisett 2015], CryptHOL [Basin et al. 2017], and Verypto [Berg 2013]. However, all of these only target classical cryptography and do not support quantum programs/protocols. Hoare logics and weakest precondition calculi for quantum programs have been presented by D’Hondt and Panangaden [D’Hondt and Panangaden 2006], Chadha, Mateus and Sernadas [Chadha et al. 2006], Feng, Duan, Ji, and Ying [Feng et al. 2007], Ying [Ying 2012], and Kakutani [Kakutani 2009]. However, these calculi do not allow us to reason about the relation between different programs.

Those quantum Hoare logics fall into two flavours: [D’Hondt and Panangaden 2006; Feng et al. 2007; Ying 2012] take a semantic approach in which a pre-/postcondition is an arbitrary subspace (or more generally, Hermitian operator to model expectations), while [Chadha et al. 2006; Kakutani 2009] represent pre-/postcondition as terms of a specific (fixed) structure. Our work follows the semantic approach.

In independent work, Zhou, Ying, Yu, and Ying [Zhou et al. 2018] study quantum couplings, and also mention (in passing) the applicability of quantum couplings to quantum relational Hoare logics. Using their notions of couplings, the resulting definition would probably be very similar to Definition 4 (which had problems with the frame rule, see above). Unfortunately, their results do not seem to give a method for proving the frame rule, and their Quantum Strassen Theorem does not seem to work for quantum couplings with separable states (as in Definition 5).

Existing approaches for computer-aided verification of the equivalence of quantum programs focus either on the explicit calculation of the quantum state resulting from an execution of the programs, e.g., [Ardeshir-Larijani et al. 2013, 2014], or on bisimilarity of process calculi, e.g., [Feng and Ying 2015; Kubota et al. 2012, 2013]. Explicit calculation will work only for finite-dimensional (and relatively small) systems, and is thus inapplicable to computational cryptography (which has necessarily an exponentially large state space). Bisimilarity can show the exact equivalence between two programs, but does not seem to scale well for modeling computational cryptographic proofs (even in the classical setting, we are not aware of any nontrivial computational security proof based purely on bisimilarity).

Several papers verify the security of the BB84 quantum key distribution (QKD) protocol. [Feng and Ying 2015; Tavala et al. 2011] show security against a very simple, specific intercept-resend attack (that measures all qubits in a random basis). [Kubota et al. 2012, 2013] verify one step of the Shor-Preskill proof [Shor and Preskill 2000] for QKD. It is not clear whether these approaches can scale to the full proof of QKD.

1.5 Organization

Section 2 presents notation and elementary definitions. Section 3 gives syntax and semantics of quantum programs. Section 4 defines what a quantum predicate is and discusses constructions of quantum predicates (in particular quantum equality, Section 4.3). Section 5 defines quantum relational Hoare logic and derives reasoning rules for qRHL. Section 6 shows example derivations in our logic. Full proofs of all claims and some additional technical material are given in the full version [Unruh 2018c].

A note on notation. Many of the proofs in this paper are notationally quite involved. This is partly due to the fact that there are a number of isomorphic spaces (e.g., states over variables, states over variables with index 1), and the isomorphisms need to be explicitly specified to make the proofs logically consistent (there is too much ambiguity to leave them implicit). Similarly, some of the rules also contain variable renamings and isomorphisms. We recommend, upon first reading, to ignore all these isomorphisms (specifically idx_1, idx_2, U_rename, ..., E_rename, ...), i.e., assume that they
are the identity. Without these, the lemmas and proofs, while not technically correct, will still contain the relevant ideas.

2 PRELIMINARIES

In this section, we give the preliminaries for understanding the technical results in this paper. For an intuitive understanding of the basic concepts of quantum mechanics, see a textbook such as [Nielsen and Chuang 2010], especially chapters 1, 2, 4, and 8.

Let $\mathbb{B} := \{\text{true}, \text{false}\}$. Let im $M$ denote the image of $M$. A program variable $x$ (short variable) is an identifier annotated with a set $\text{Type}_x \neq \emptyset$, and with a flag that determines whether the variable is quantum or classical. Given a set/list $V$ of variables, we write $V^{\text{cl}}, V^{\text{qu}}$ for the classical/quantum variables in $V$, respectively. $\text{Type}_V^\text{set}$ is the set of all functions $f$ on $V$ with $f(x) \in \text{Type}_x$ for all $x \in V$. And $\text{Type}_V^{\text{list}} := \text{Type}_{x_1} \times \cdots \times \text{Type}_{x_n}$. For disjoint sets $V, V'$ of variables, we abbreviate $V \vee V' := V \cup V'$.

Let $\text{D}(X)$ denote the set of (discrete) distributions on $X$, i.e., functions $\mu : X \to \mathbb{R}_{\geq 0}$. The set of distributions with $\sum_{x \in X} \mu(x) \leq 1$ is denoted $\text{D}^{\leq 1}(X)$ (subprobability distributions). We call a distribution total if $\sum_{x \in X} \mu(x) = 1$. For a set of variables, let $\text{D}(V) := \text{D}(\text{Type}_V^{\text{set}})$ and $\text{D}^{\leq 1}(V) := \text{D}^{\leq 1}(\text{Type}_V^{\text{set}})$. The support supp $\mu$ is the set of all $x$ with $\mu(x) > 0$.

For $\mu \in \text{D}(X \times Y)$, let marginal$_1(\mu)$ be the first marginal of $\mu$, that is, $\text{marginal}_1(\mu)(m_1) := \sum_{m_2 \in Y} \mu(m_1, m_2)$. And let marginal$_2(\mu)$ be the second marginal.

Let $\ell^2(X)$ denote the Hilbert space of all function $\psi : X \to \mathbb{C}$ with $\|\psi\|^2 := \sum_{x \in X} |\psi(x)|^2 < \infty$. (That is, $\ell^2(X)$ represents pure quantum states.) Let $\text{proj}(\psi) := \psi^\dagger \psi$, where $\psi^\dagger$ is the adjoint. Let $B(X, Y)$ denote the bounded operators on $\ell^2(X) \to \ell^2(Y)$, and $U(X, Y)$ the unitary operators, and $\text{Iso}(X, Y)$ the isometries. And $B(X) := B(X, X), U(X) := U(X, X), \text{Iso}(X) := \text{Iso}(X, X)$. Let $T(X)$ denote the trace-class operators on $\ell^2(X)$, and $T^+(X)$ the positive trace-class operators. For sets of variables $V, W$, we write $B[V, W]$ for $B(\text{Type}_V^{\text{set}}, \text{Type}_W^{\text{set}})$ and analogously $\ell^2[V], B[V], U[V], \text{Iso}[V], T^+[V]$ etc. (That is, $\ell^2[V]$ is the set of all pure quantum states that a memory with variables $V$ can be in.) $\text{Type}_V^{\text{set}}$ are the standard basis vectors of $\ell^2[V]$. With this notation, $T^+[V]$ can be seen as the density operators over quantum registers $V$. (That is, $T^+[V]$ is the set of all mixed quantum states that a memory with variables $V$ can be in.) $\rho \in T[V]$ is a cq-operator if it can be written as $\rho = \sum_{m \in \text{Type}_V^{\text{set}}} \text{proj}(|m\rangle \langle m|) \otimes \rho_m$ with $\rho_m \in T[V^{\text{qu}}]$. $T_{cq}[V]$ is the set of cq-operators. $T_{cq}^+[V] := T_{cq}[V] \cap T^+[V]$. ("cq" stands for "classical-quantum". $T_{cq}^+[V]$ are the density operators that contain classical data in the classical variables $V^{\text{cl}}$.) For $\rho = \sum_i \text{proj}(\psi_i)$, the support supp $\rho$ is span $\{|\psi_i\rangle\}$. (This is independent of the choice of $\psi_i$.) The tensor product $\ell^2[X] \otimes \ell^2[Y]$ of spaces is simply $\ell^2[XY]$ (where $XY = X \cup Y$). For $\psi \in \ell^2[X], \phi \in \ell^2[Y]$, the tensor product is written $\psi \otimes \phi \in \ell^2[X] \otimes \ell^2[Y]$. Note: since $\ell^2[X], \ell^2[Y]$ are "labelled" with the variable names $X, Y$, we have that $\psi \otimes \phi = \phi \otimes \psi$, i.e., the order does not matter. Analogously for the tensor product on $B[X]$ etc. Since $\text{Type}_V^{\text{list}}$ and $\text{Type}_V^{\text{set}}$ are in one-to-one correspondence, $\ell^2(\text{Type}_V^{\text{list}})$ and $\ell^2[V]$ are isomorphic. We write $U_{\text{var}_V}$ for the isomorphism $\ell^2(\text{Type}_V^{\text{list}}) \to \ell^2[V]$. Intuitively, we can think of $U_{\text{var}_V}$ as annotating a state $\psi \in \ell^2(\text{Type}_V^{\text{list}})$ with the variable names $V$.

We call a density operator $\rho \in T^+[V_1V_2]$ ($V_1, V_2$)-separable (short: separable) iff it can be written as $\sum_i \rho'_i \otimes \rho''_i$ for some $\rho'_i \in T^+[V_1], \rho''_i \in T^+[V_2]$. Intuitively, this means that there is no entanglement between subsystems $V_1$ and $V_2$.

A superoperator from $V$ to $W$ is a completely positive trace-decreasing\(^9\) linear map $\mathcal{E} : T[V] \to T[W]$. A cq-superoperator is a completely positive trace-decreasing linear map $\mathcal{E} : T_{cq}[V] \to T_{cq}[W]$.

\(^7\)In the finite-dimensional case, $B(X, Y)$ is simply the set of all linear functions.

\(^8\)Trace-class operators are operators for which the trace exists. In the finite dimensional case, every operator is trace-class.

\(^9\)I.e., $\text{tr}(\mathcal{E}(\rho)) \leq \text{tr}(\rho)$.
For $\rho \in T^+[VW]$, $\text{tr}^{(V)}_W \rho \in T^+[V]$ denotes the partial trace of $\rho$, i.e., the state after “destroying” the subsystem corresponding to $W$. (Note: usually, the partial trace is written simply $\text{tr}_W$, i.e., only the subsystem $W$ that is removed is mentioned. However, for additional clarity, we also mention the retained subsystem $V$ in the symbol $\text{tr}^{(V)}_W$.) $\text{tr}$ without super-/subscripts is the trace.

Let $\text{Meas}(D, E)$ denote the set of all projective measurements on $\ell^2(E)$ with outcomes in $D$. Formally, $\text{Meas}(D, E)$ is the set of functions $M : D \rightarrow B(E)$ such that $M(z)$ is a projector for all $z \in D$, and $\sum_z M(z) \leq \text{id}$. We call a measurement $M$ total iff $\sum_z M(z) = \text{id}$.

3 QUANTUM PROGRAMS

In the context of this paper, an expression $e$ is – informally speaking – an arbitrary mathematical formula, possibly containing free classical variables. (See Section 2 for the definition of variables.) We denote by $fv(e)$ the set of free variables occurring in the expression. For $m \in \text{Type}_X^\text{set}$ with $X \supseteq fv(e)$, let $[e]_m$ denote the result of substituting every $x \in fv(e)$ in $e$ by $m(x)$ and then evaluating the resulting expression (which has no free variables any more). Intuitively, this corresponds to evaluating the expression $e$ on a memory $m$ that contains the content of the variables. With every expression $e$, we associate a set $\text{Type}_e^{\text{exp}}$ such that $[e]_m \in \text{Type}_e^{\text{exp}}$ for all $m$.

Formally, an expression simply consists of a finite set $fv(e) \subseteq V^c$, a set $\text{Type}_e^{\text{exp}}$, and a function $[e] : \text{Type}_V^\text{set} \rightarrow \text{Type}_e^{\text{exp}}$ (with its argument written as subscript) such that $m[fv(e)] = m'[fv(e)] \implies [e]_m = [e]_{m'}$.

3.1 Syntax of Programs

Programs in our language are constructed according to the following grammar. We will typically denote programs with c or d.

$$
c, d ::= \text{skip} \quad \text{(no operation)}
$$

$$
x \leftarrow e \quad \text{(classical assignment)}
$$

$$
x \leftarrow \cdot e \quad \text{(classical sampling)}
$$

$$
\text{if } e \text{ then } c \text{ else } d \quad \text{(conditional)}
$$

$$
\text{while } e \text{ do } c \quad \text{(loop)}
$$

$$
c; d \quad \text{(sequential composition)}
$$

$$
q_1 \ldots q_n \overset{q}{\leftarrow} e \quad \text{(initialization of quantum registers)}
$$

$$
\text{apply } e \text{ to } q_1 \ldots q_n \quad \text{(quantum application)}
$$

$$
x \leftarrow \text{measure } q_1 \ldots q_n \text{ with } e \quad \text{(measurement)}
$$

In the sampling statement, $e$ evaluates to a distribution. In the initialization of quantum registers, $e$ evaluates to a pure quantum state, $q_1 \ldots q_n$ are jointly initialized to that state. In the quantum application, $e$ evaluates to an isometry that is applied to $q_1 \ldots q_n$. In the measurement, $e$ evaluates to a projective measurement, the outcome is stored in $x$. (Recall that an expression $e$ can be an arbitrarily complex mathematical formula in the classical variables. So, e.g., an expression that describes an isometry could be something as simple as just $H$ (here $H$ denotes the Hadamard transform), or something more complex such as, e.g., $H^x$, meaning $H$ is applied if $x = 1$.)

A program is well-typed according to the following rules:

---

\[10\] This formalization is more suitable for modeling expressions in a formal logical system (and thus for implementing in a theorem prover such as Isabelle or Coq). In this paper, we nevertheless write expressions as formulas with free variables because this leads to more readable formulas. E.g., we can simply write $e + f$ if $e, f$ are expressions, instead of having to write $\lambda m.(e(m) + f(m))$ as we would have to if expressions were functions.
• \( x \leftarrow e \) is well-typed iff \( \text{Type}_{e}^{\exp} \subseteq \text{Type}_{x} \),
• \( x \overset{s}{\leftarrow} e \) is well-typed iff \( \text{Type}_{e}^{\exp} \subseteq D^{s}(\text{Type}_{x}) \),
• if \( e \rightarrow c \) else \( d \) is well-typed iff \( \text{Type}_{e}^{\exp} \subseteq \mathbb{B} \) and \( c, d \) are well-typed.
• while \( e \) do \( c \) is well-typed iff \( \text{Type}_{e}^{\exp} \subseteq \mathbb{B} \) and \( c \) is well-typed.
• \( c ; d \) is well-typed iff \( c \) and \( d \) are well-typed.
• \( Q \overset{q}{\leftarrow} e \) is well-typed iff \( \text{Type}_{e}^{\exp} \subseteq \mathcal{L}^{2}(\text{Type}_{Q}^{\exp}) \), and \( \|\psi\| = 1 \) for all \( \psi \in \text{Type}_{e}^{\exp} \).
• apply \( e \) to \( Q \) is well-typed iff \( \text{Type}_{e}^{\exp} \subseteq \text{Iso}(\text{Type}_{Q}^{\exp}) \).

In this paper, we will only consider well-typed programs. That is, “program” implicitly means “well-typed program”, and all derivation rules hold under the implicit assumption that the programs in premises and conclusions are well-typed.

The set of all free variables in a program \( c \) is denoted \( \text{fv}(c) \) and consists of all quantum and classical variables occurring in \( c \) (including the variables occurring in expressions in \( c \)).

### 3.2 Semantics

Fix some set \( V \) of variables relative to which the semantics will be defined (in the remainder of the paper, that set will always be clear from the context, and it will always be called either \( V, V_{1}, \) or \( V_{2}) \).

Given a program \( c \) (with \( \text{fv}(c) \subseteq V \)), we define its semantics \( \llbracket c \rrbracket \) as a cq-superoperator on \( T_{cq}[V] \). In the following, let \( \rho \in T_{cq}[V], m \in \text{Type}_{V}^{\exp}, \) and \( \rho_{m} \in T^{*}[V^{\exp}] \). Note that specifying \( \llbracket c \rrbracket \) on operators \( \text{proj}(m) \otimes \rho_{m} \) specifies \( \llbracket c \rrbracket \) on all \( \rho \in T_{cq}[V] \).

\[
\llbracket \text{skip} \rrbracket (\rho) := \rho
\]
\[
\llbracket x \leftarrow e \rrbracket (\text{proj}(m) \otimes \rho_{m}) := \text{proj}(m(x := \llbracket e \rrbracket_{m})) \otimes \rho_{m}
\]
\[
\llbracket x \overset{s}{\leftarrow} e \rrbracket (\text{proj}(m) \otimes \rho_{m}) := \sum_{z \in \text{Type}_{x}} \llbracket e \rrbracket_{m}(z) \cdot \text{proj}(m(x := z)) \otimes \rho_{m}
\]
\[
\llbracket \text{if } e \text{ then } c \text{ else } d \rrbracket (\rho) := \llbracket c \rrbracket (\downarrow_{e}(\rho)) + \llbracket d \rrbracket (\downarrow_{\neg e}(\rho))
\]
\[
\llbracket \text{while } e \text{ do } c \rrbracket (\rho) := \sum_{i=0}^{\infty} \downarrow_{e} ((\llbracket c \rrbracket \circ \downarrow_{e})^{i}(\rho))
\]
\[
\llbracket c_{1} ; c_{2} \rrbracket := [c_{2}] \circ [c_{1}]
\]
\[
\llbracket Q \overset{q}{\leftarrow} e \rrbracket (\text{proj}(m) \otimes \rho_{m}) := \text{proj}(m) \otimes \text{tr}^{V_{\text{env}}(Q)}_{Q} \rho_{m} \otimes \text{proj}(U_{\text{vars}, Q} [\llbracket e \rrbracket_{m}])
\]
\[
\llbracket \text{apply } e \text{ to } Q \rrbracket (\text{proj}(m) \otimes \rho_{m}) := \text{proj}(m) \otimes U \rho_{m} U^{*}
\]
\[
\text{where } U := U_{\text{vars}, Q} [\llbracket e \rrbracket_{m}] U_{\text{vars}, Q}^{*} \otimes id_{V_{\text{env}}(Q)}
\]
\[
\llbracket x \leftarrow \text{measure } Q \text{ with } e \rrbracket (\text{proj}(m) \otimes \rho_{m}) := \sum_{z \in \text{Type}_{x}} \text{proj}(m(x := z)) \otimes P_{z} \rho_{m} P_{z}
\]
\[
\text{where } P_{z} := U_{\text{vars}, Q} [\llbracket e \rrbracket_{m}] U_{\text{vars}, Q}^{*} \otimes id_{V_{\text{env}}(Q)}
\]

Here \( \downarrow_{e}(\rho) \) is the cq-density operator \( \rho \) restricted to the parts where the expression \( e \) holds. Formally, \( \downarrow_{e} \) is the cq-superoperator on \( V \) such that

\[
\downarrow_{e}(\text{proj}(m) \otimes \rho_{m}) := \begin{cases} 
\text{proj}(m) \otimes \rho_{m} & (\llbracket e \rrbracket_{m} = \text{true}) \\
0 & (\text{otherwise})
\end{cases}
\]
Section 3

We only explain them intuitively here, the necessary definitions to understand them formally will after running the program on initial state \( \rho \).

Definition 8. For a set \( X \) of variables, we call \( c \) X-local iff \( \| c \| = \mathcal{E} \otimes id_{V \setminus X} \) for some cq-superoperator \( \mathcal{E} \) on \( X \).

Lemma 9. If \( fv(c) \subseteq X \), then \( c \) is X-local.

4 QUANTUM PREDICATES

4.1 Defining Predicates

In this section, we discuss what a predicate is, i.e., how predicates are mathematically represented.

In the introduction, we already gave a definition of predicates. Namely, a predicate \( A \) over variables \( V \) is a subspace of \( \ell^2[V] \), and a cq-density operator \( \rho \in T^+_c[V] \) satisfies \( A \) iff \( \text{supp} \ \rho \subseteq A \). Let us call a predicate according to this definition a simple predicate.

While there is nothing wrong per se with simple predicates, it turns out that we can optimize the representation of predicates somewhat. For a simple predicate \( A \), let \( A_m := \{ \psi \in \ell^2[V^{qu}] : |m\rangle_{\text{cl}} \otimes \psi \in A \} \). For any cq-density operator \( \rho = \sum_m \text{proj}(|m\rangle_{\text{cl}}) \otimes \rho_m \), we then have

\[
\rho \text{ satisfies } A \iff \text{supp} \ \rho \subseteq A \iff \forall m. \{ |m\rangle_{\text{cl}} \} \otimes \text{supp} \ \rho_m \subseteq A \iff \forall m. \rho_m \subseteq A_m.
\]

This implies that a simple predicate \( A \) is essentially described by the subspaces \( A_m \subseteq \ell^2[V^{qu}] \).

Thus, we will define predicates on \( V \) not as subspaces of \( \ell^2[V] \), but as families of subspaces of \( \ell^2(V^{qu}) \), indexed by Type\(_{V^cl}^{set}\).

And in fact, a family indexed by Type\(_{V^cl}^{set}\) is most conveniently described as an expression (in the sense of Section 3) with free variables in \( V^{cl} \). Namely, if we represent a predicate as an expression \( A \) with Type\(_A^{exp} \subseteq \ell^2[V^{qu}] \), we get the subspaces \( A_m \) from above as \( [A]_m \).

This leads to the following definition:

Definition 10 (Quantum Predicates). A (quantum) predicate \( A \) over \( V \) is an expression with \( fv(A) \subseteq V^{cl} \) and Type\(_A^{exp} \subseteq \{ S : S \ is \ a \ subspace \ of \ \ell^2[V^{qu}] \} \). A state \( \rho \in T^+_c[V] \) satisfies \( A \) iff \( \text{supp} \ \rho_m \subseteq [A]_m \) for all \( m \), where \( \rho = \sum_{m \in \text{Type}_{V^cl}^{set}} \text{proj}(|m\rangle_{\text{cl}}) \otimes \rho_m \).

This definition of predicates will make notation much more convenient because it will be possible to express the properties of the quantum variables in terms of the values of the classical variables.

Examples of Predicates. We give some examples of predicates to illustrate the concepts involved. We only explain them intuitively here, the necessary definitions to understand them formally will be given in the following sections. Basically, an expression is a subspace of possible quantum states of the program’s memory. In many cases, we want to say that a specific variable satisfies a predicate.

E.g., we want to express that the state of variable \( q \) is in subspace \( P \) (where \( P \) is a subspace of \( \ell^2[q] \)). We will write this as \( P \Rightarrow q \) (and we will formally define this as a subspace later). I.e., \( P \Rightarrow q \) means intuitively \( q \in P \). Similarly, we can write \( \text{span}\{q\} \Rightarrow q \) to denote that \( q \) has state \( \psi \). Since we can refer to classical variables in predicates, we can write, e.g., \( \text{span}\{x\} \Rightarrow q \) to denote that \( q \) has state \( |x\rangle \) where \( x \) is a classical variable. Or, e.g., \( \text{span}\{H^2|0\rangle\} \Rightarrow q \), where we apply \( H \) to \( |0\rangle \) iff \( x = 1 \).

\(^{11}\)We say “essentially” here, because \( A \) is not uniquely determined by the \( A_m \). But the set of cq-density operators that satisfy \( A \) are determined by the \( A_m \).
We can combine predicates with logical connectives. In this paper we will use \(+\) to represent disjunction and \(\cap\) to represent conjunction (Birkhoff-von Neumann quantum logic).\(^{12}\) For example, \(\text{span}\{0\} \cap \text{span}\{1\}\) means that \(q\) has state \(|0\rangle\) and \(r\) has state \(|1\rangle\).

Moreover, we often need to express facts about the classical variables alone. We have syntactic sugar \(\mathbb{Cl}[\ldots]\) for that. For example, \(\mathbb{Cl}[x = 1]\) means that the classical variable \(x\) has value 1. We can write arbitrary formulas inside \(\mathbb{Cl}[\ldots]\) as long as they involve only classical variables. E.g., \(\mathbb{Cl}[\forall z \in \mathbb{X}. z \leq y]\) would say that the \(x\) (a variable of type set) is upper-bounded by \(y\). Note that predicates of the form \(\mathbb{Cl}[\ldots]\) can be combined with non-classical predicates using \(+\) and \(\cap\).

Finally, we can even express that two quantum variables have the same content (formalized below) using expressions such as \(q \equiv \text{quant} \ r\).

We stress that predicates are not restricted to what can be built from the above constructions. Any mathematical expression that describes a subspace is permissible. The above are merely a useful selection of common cases.

### 4.2 Operations on Predicates

We now define a few elementary operations on predicates.

First, any operation that can be performed on subspaces is meaningful on predicates, too. For example, the intersection \(A \cap B\) is the quantum analogue to the conjunction of classical predicates, the sum \(A + B\) is a quantum analogue to the disjunction of classical predicates, and the orthogonal complement \(A^\perp\) is a quantum analogous to the negation of a classical predicate (cf. Lemma 21 below).

**Lemma 11.** A cq-density operator \(\rho\) satisfies \(A \cap B\) iff \(\rho\) satisfies \(A\) and \(\rho\) satisfies \(B\).

(Note that the analogue for \(A + B\) does not hold.)

We can easily specify relations between predicates. A predicate \(A\) implies another predicate \(B\) iff \(A \subseteq B\) for all assignments to the free variables of \(A\) and \(B\). That is:

**Definition 12.** We say \(A \subseteq B\) iff \(\llbracket A \rrbracket_m \subseteq \llbracket B \rrbracket_m\) for all \(m\).

**Lemma 13.** If \(A \subseteq B\), and \(\rho\) satisfies \(A\), then \(\rho\) satisfies \(B\).

Often we need to specify which quantum and classical variables a given predicate refers to: A predicate is \(X\)-local if we only need to look at classical and quantum variables in \(X\) to decide whether the predicate is satisfied. Formally:

**Definition 14 (Locality).** Fix variables \(X \subseteq V\). A predicate \(\Lambda\) on \(V\) is \(X\)-local iff \(fv(\Lambda) \subseteq X^{cl}\), and for all \(m \in \text{Type}_{X^{cl}}\), \(\llbracket A \rrbracket_m = S_m \otimes \ell^2[V^{qu} \setminus X^{qu}]\) for some subspace \(S_m\) of \(\ell^2[X^{qu}]\).

**Lifting of operators/subspaces.** Often, we will need to express predicates like “the state of variables \(q\) \(\in\) \(V\) is in subspace \(S\)” where \(q\) has type \(T\) and \(S \subseteq \ell^2(T)\). Formally, this can be encoded as the predicate \(U_{\text{vars},q}S \otimes \ell^2[V^{qu} \setminus q]\) (which is a subspace of \(\ell^2[V^{qu}]\)) where \(U_{\text{vars},q}\) is the isomorphism \(\ell^2(T) \rightarrow \ell^2[q]\) described in the preliminaries. Since this notation is cumbersome, we abbreviate it as \(S \triangleright q\).

For example, if \(q\) has type \(\{0, 1\}\), then \(\text{span}\{0\} \triangleright q\) is the subspace spanned by all \(|m\rangle\)\(_r\) with \(m(q) = 0\). Informally, this is the predicate “\(q\) is in state \(|0\rangle\).”

Similarly, we will often need to apply an operator \(A\) \(\in\) \(B(T)\) to the (subsystem containing the) quantum variable \(q\). This is done by applying \(U_{\text{vars},q}AU_{\text{vars},q}^* \otimes id_{V^{qu}}\) \(\in B[V^{qu}]\) to the overall system. We abbreviate this as \(A \triangleright q\).
For example, if $H$ is the Hadamard gate, and $q$ is of type $\{0, 1\}$, then applying $H \triangleright q$ to a state (in $\ell^2[\mathbb{V}^q]$) is the same as applying $H$ to the subsystem corresponding to variable $q$.

The following definition generalizes this to lists of variables:

**Definition 15 (Lifting).** Fix a list $Q \subseteq \mathbb{V}^q$. Let $S \subseteq \ell^2(\text{Type}_{Q}^\text{list})$ be a subspace, and $A \in \text{B}(\text{Type}_{Q}^\text{list})$. Then:

$$S \triangleright Q := U_{\text{vars}, Q} S \otimes \ell^2[\mathbb{V}^q \setminus Q] \quad \text{and} \quad A \triangleright Q := U_{\text{vars}, Q} A U_{\text{vars}, Q}^\ast \otimes \text{id}_{\mathbb{V}^q \setminus Q}.$$ 

Here $V$ is a global set of variables, implicitly understood. For example, in the context of qRHL judgments (Section 5), this set will usually be $V_1 V_2$, the union of the sets of all variables indexed with 1 and 2, respectively.

**“Division” of subspaces.** Finally, we introduce one technical definition that will be needed to express the precondition of quantum initialization statements (rule QINIT1 below):

**Definition 16.** Fix variables $W \subseteq V$, let $A \subseteq \ell^2[V]$ be a subspace, and let $\psi \in \ell^2[W]$. Then $A \div \psi \subseteq \ell^2[V \setminus W]$ is defined by

$$\phi \in A \div \psi \quad \text{iff} \quad \phi \otimes \psi \in A.$$ 

The notation $\div$ is motivated by the fact that if $A = B \otimes \text{span}\{\psi\}$, then $A \div \psi = B$, so $\div$ is in a sense the division operation corresponding to the tensor product $\otimes$.

The following simplification rule is useful for simplifying subgoals arising from rule QINIT1 (in combination with rule CONSEQ):

**Lemma 17.** Let $A, B \subseteq \ell^2[V^q]$ be subspaces. Let $Q \subseteq \mathbb{V}^q$ and $\psi \in \ell^2[Q]$. Assume that $A$ is $(\mathbb{V}^q \setminus Q)$-local. Then

$$A \subseteq (B \div \psi) \otimes \ell^2[Q] \iff A \cap (\text{span}\{\psi\} \otimes \ell^2[\mathbb{V}^q \setminus Q]) \subseteq B.$$ (1)

**Total programs.** Sometimes we need to specify that a program is total, i.e., that it terminates with probability 1. In case the totality holds only for certain initial states, the following definition can be used:

**Definition 18 (Total programs).** A program $c$ is total on $A$ if for every $\rho \in T_{cq}[V]$ that satisfies $A$, we have $\text{tr}[c](\rho) = \text{tr} \rho$.

In many cases, we need predicates that only talk about classical variables. The following definition allows us to do so:

**Definition 19 (Classical predicates).** We define $\mathbb{C}la[b]$ for $b \in \mathbb{B}$ as follows: $\mathbb{C}la[\text{true}] := \ell^2[\mathbb{V}^q]$ and $\mathbb{C}la[\text{false}] := 0$ (the trivial subspace).

The next lemma explains why $\mathbb{C}la[\ldots]$ allows us to encode classical predicates:

**Lemma 20.** Fix an expression $e$ with $\text{Type}_{e}^{\exp} \subseteq \mathbb{B}$ and $\text{fv}(e) \subseteq \mathbb{V}^c$. Then for any $\rho = \sum_m \text{proj}(m)_{m=0} \otimes \rho_m \in T_{cq}[V]$, we have: $\rho$ satisfies $\mathbb{C}la[e]$ iff $[e]_m = \text{true}$ for all $m$ with $\rho_m \not= 0$.

Predicates of the form $\mathbb{C}la[\ldots]$ also illustrate why intersection and sum of subspaces are the analogues to conjunction and disjunction:

**Lemma 21.** For expressions $e$, $f$ with $\text{Type}_{e}^{\exp}$, $\text{Type}_{f}^{\exp} \subseteq \mathbb{B}$, and expressions $g$, we have: (i) $\mathbb{C}la[e] \cap \mathbb{C}la[f] = \mathbb{C}la[e \land f]$. (ii) $\mathbb{C}la[e] + \mathbb{C}la[f] = \mathbb{C}la[e \lor f]$. (iii) $\mathbb{C}la[e] \vdash = \mathbb{C}la[-e]$. (iv) $\bigcap_{z \in g} \mathbb{C}la[e] = \mathbb{C}la[\forall z \in g. e]$. (Here we assume that $\text{Type}_{g}^{\exp} \subseteq \text{Type}_{e}^{\exp}$. $z$ may occur free in $e$.)
4.3 Quantum Equality

One predicate that is very specific to the setting of qRHL is that of quantum equality. As discussed in the introduction (Section 1.2), we need a predicate \( Q_1 \equiv_{\text{quant}} Q_2 \) that encodes the intuitive fact that the quantum variables \( Q_1 \) contain the same state as the quantum variables \( Q_2 \). If we want \( Q_1 \equiv_{\text{quant}} Q_2 \) to be a predicate in our sense (i.e., a subspace), there seems to be only one possible definition, namely, \( Q_1 \equiv_{\text{quant}} Q_2 \) is the space of all quantum states that are invariant under swapping the content of \( Q_1 \) and \( Q_2 \). (The fact that this is the only possibility is a conjecture only, but we give some formal evidence for this in the full version [Unruh 2018c].)

How do we write the unitary \( U \) that swaps \( Q_1 \) and \( Q_2 \)? We cannot simply define \( U \) by \( U(\psi_1 \otimes \psi_2) = \psi_2 \otimes \psi_1 \) for \( \psi_1 \in \ell^2[Q_1] \), \( \psi_2 \in \ell^2[Q_2] \), since \( \ell^2[Q_1] \) and \( \ell^2[Q_2] \) are different (but isomorphic) spaces. The canonical isomorphism from \( \ell^2[Q_1] \) to \( \ell^2[Q_2] \) is \( U_{\text{vars}, Q_1} \psi_{\text{vars}, Q_1} \), and the canonical isomorphism from \( \ell^2[Q_2] \) to \( \ell^2[Q_1] \) is \( U_{\text{vars}, Q_2} U_{\text{vars}, Q_1}^* \). We can then define the swap by \( U(\psi_1 \otimes \psi_2) = U_{\text{vars}, Q_1} U_{\text{vars}, Q_2}^* \psi_2 \otimes U_{\text{vars}, Q_1} U_{\text{vars}, Q_2}^* \psi_1 \). And since the tensor product is commutative (in our formalization which uses variables to identify the different factors), we can write this as: \( U(\psi_1 \otimes \psi_2) = U_{\text{vars}, Q_1} U_{\text{vars}, Q_2}^* \psi_1 \otimes U_{\text{vars}, Q_1} U_{\text{vars}, Q_2}^* \psi_2 \). That is, the swap of \( Q_1 \) and \( Q_2 \) is simply \( U = U_{\text{vars}, Q_1} U_{\text{vars}, Q_2}^* \otimes U_{\text{vars}, Q_1} U_{\text{vars}, Q_2}^* \). If \( Q_1 Q_2 \) is not the set of all quantum variables \( V \), we additionally need to express that \( U \) does not change the other variables, and thus we get

\[
U := U_{\text{vars}, Q_1} U_{\text{vars}, Q_2}^* \otimes U_{\text{vars}, Q_1} U_{\text{vars}, Q_2}^* \otimes \text{id}_{\text{vars}, Q_1} \otimes \text{id}_{\text{vars}, Q_1} \in U[\text{V}^{\text{qu}}].
\]

We can now define \( Q_1 \equiv_{\text{quant}} Q_2 \subseteq \ell^2[\text{V}^{\text{qu}}] \) as the subspace fixed by \( U \) (i.e., the set of all \( \psi \in \ell^2[\text{V}^{\text{qu}}] \) with \( U \psi = \psi \)).

Before we do so (Definition 23 below), however, let us generalize the concept of quantum equality somewhat. Consider the following classical equality: \( f(x_1) = g(x_2) \). This means that if we apply \( f \) to \( x_1 \), we get the same value as when applying \( g \) to \( x_2 \). We would like to express something like this also using quantum variables. Namely, we wish to express that \( Q_1 \), when applying an isometry \( U_1 \), has the same content as \( Q_2 \) when applying an isometry \( U_2 \). (The quantum equality \( Q_1 \equiv_{\text{quant}} Q_2 \) we discussed so far is a special case of this with \( U_1 = U_2 = \text{id} \).) To model this, we define the swap somewhat differently, namely, instead of \( U_{\text{vars}, Q_1} \) (which simply maps \( \psi_1 \in \ell^2[Q_1] \) to \( \ell^2(\text{Type}_{\text{list}}^Q) \)), we use the morphism \( U_1 U_{\text{vars}, Q_1} \) (which maps \( \psi_1 \in \ell^2[Q_1] \) to \( \ell^2(\text{Type}_{\text{list}}^Q) \) and then applies \( U_1 \), and instead of \( U_{\text{vars}, Q_2} \), we use \( U_2 U_{\text{vars}, Q_2} \). Thus we get the following operation \( U' \in B[\text{V}^{\text{qu}}] \) instead of \( U \):

\[
U' := U_{\text{vars}, Q_1} U_{\text{vars}, Q_1}^* U_1 U_{\text{vars}, Q_1}^* \otimes U_{\text{vars}, Q_1} U_{\text{vars}, Q_2} U_2 U_{\text{vars}, Q_2}^* \otimes \text{id}_{\text{vars}, Q_1} \otimes \text{id}_{\text{vars}, Q_2},
\]

and we define \( U_1 Q_1 \equiv_{\text{quant}} U_2 Q_2 \) to consist of all \( \psi \) fixed by \( U' \):

**Definition 22 (Quantum Equality).** Let \( V \) be a set of variables, and \( Q_1, Q_2 \subseteq V^{\text{qu}} \) be disjoint lists of distinct quantum variables. Let \( Z \) be a set. Let \( U_1 \in B(\text{Type}_{\text{list}}^Q, Z) \) and \( U_2 \in B(\text{Type}_{\text{list}}^Q, Z) \). Then \( (U_1 Q_1 =_{\text{quant}} U_2 Q_2) \subseteq \ell^2[\text{V}^{\text{qu}}] \) is defined as the subspace fixed by

\[
U_{\text{vars}, Q_1} U_{\text{vars}, Q_1}^* U_1 U_{\text{vars}, Q_1}^* \otimes U_{\text{vars}, Q_1} U_{\text{vars}, Q_2} U_1 U_{\text{vars}, Q_2}^* \otimes \text{id}_{\text{vars}, Q_1} \otimes \text{id}_{\text{vars}, Q_2}.
\]

From the general definition of \( U_1 Q_1 \equiv_{\text{quant}} U_2 Q_2 \), we can recover the special case \( Q_1 \equiv_{\text{quant}} Q_2 \) that we started with by setting \( U_1 := U_2 := \text{id} \):

**Definition 23.** Let \( Q_1, Q_2 \) be disjoint lists of distinct quantum variables with \( \text{Type}_{\text{list}}^Q \equiv_{\text{quant}} \text{Type}_{\text{list}}^Q \). Then \( (Q_1 =_{\text{quant}} Q_2) \equiv (\text{id} Q_1 =_{\text{quant}} \text{id} Q_2) \) where \( \text{id} \) is the identity on \( \text{Type}_{\text{list}}^Q \).
The following lemma gives a characterization of the quantum equality on separable states. This characterization will make the proofs of rules involving quantum equality much simpler.

**Lemma 24.** Let $Q_1 \subseteq V^{\text{qu}}_1$ and $Q_2 \subseteq V^{\text{qu}}_2$, $U_1 \in B^{\text{in}}(\text{Type}_{Q_1}^\text{list}, Z)$, and $U_2 \in B^{\text{in}}(\text{Type}_{Q_2}^\text{list}, Z)$ for some set $Z$. Let $\psi_1 \in \ell^2[V^{\text{qu}}_1]$, $\psi_2 \in \ell^2[V^{\text{qu}}_2]$ be normalized.

If $\psi_1 \otimes \psi_2 \in (U_1Q_1 \equiv_{\text{quant}} U_2Q_2)$ then there exist normalized $\psi_1^Q \in \ell^2[Q_1]$, $\psi_2^Q \in \ell^2[Q_2]$ such that $U_1U_{\text{vars}, Q_1}\psi_1^Q = U_2U_{\text{vars}, Q_2}\psi_2^Q$ and $\psi_1 = \psi_1^Q \otimes \psi_1^Y$ and $\psi_2 = \psi_2^Q \otimes \psi_2^Y$.

If $U_1, U_2$ are isometries, the converse holds as well.

Note that this lemma assumes that $Q_1$ and $Q_2$ are in different subsystems that are not entangled (in states $\psi_1, \psi_2$, respectively); this will be the case in qRHL judgments as defined in Section 5 below. The lemma shows that in that case, $U_1Q_1 \equiv_{\text{quant}} U_2Q_2$ implies that the variables $Q_1$ and $Q_2$ are not entangled with any other variables.

For the special case $Q_1 \equiv_{\text{quant}} Q_2$, the lemma can be additionally simplified:

**Corollary 25.** Let $Q_1 = (q_1, \ldots, q_n) \subseteq V^{\text{qu}}_1$ and $Q_2 = (q_1', \ldots, q_n') \subseteq V^{\text{qu}}_2$ be disjoint lists of distinct variables with $\text{Type}_{Q_1}^\text{list} = \text{Type}_{Q_2}^\text{list}$. Let $\psi_1 \in \ell^2[V^{\text{qu}}_1]$ be normalized. Let $\sigma : Q_2 \rightarrow Q_1$ be the variable renaming with $\sigma(q_i') = q_i$.

Then $\psi_1 \otimes \psi_2 \in (Q_1 \equiv_{\text{quant}} Q_2)$ iff there exist normalized $\psi_1^Q \in \ell^2[Q_1]$, $\psi_1^Y \in \ell^2[V^{\text{qu}}_1 \setminus Q_1]$, $\psi_2^Q \in \ell^2[Q_2]$, $\psi_2^Y \in \ell^2[V^{\text{qu}}_2 \setminus Q_2]$ such that $\psi_1^Q = U_{\text{rename}, \sigma}\psi_2^Q$ and $\psi_1 = \psi_1^Q \otimes \psi_1^Y$ and $\psi_2 = \psi_2^Q \otimes \psi_2^Y$.

**Simplification rules.** We prove three simplification rules that are useful for rewriting predicates involving quantum equalities:

**Lemma 26.** For lists $Q_1$ and $Q_2$ of quantum variables, and operators $U_1 \in B(\text{Type}_{Q_1}^\text{list}, Y)$ and $U_2 \in B(\text{Type}_{Q_2}^\text{list}, Y)$, and $A \in B(Y, Z)$, we have

$$(AU_1)Q_1 \equiv_{\text{quant}} U_2Q_2 \quad \Rightarrow \quad U_1Q_1 \equiv_{\text{quant}} (A^*U_2)Q_2.$$ 

This is especially useful for canceling out terms, e.g., we get $(A Q_1 \equiv_{\text{quant}} A Q_2) = (\text{id} Q_1 \equiv_{\text{quant}} A^* A Q_2) = (Q_1 \equiv_{\text{quant}} Q_2)$ for isometries $A$.

**Lemma 27.** For lists $Q_1$ and $Q_2$ of quantum variables, and operators $U_1 \in B(\text{Type}_{Q_1}^\text{list}, Z)$ and $U_2 \in B(\text{Type}_{Q_2}^\text{list}, Z)$ and $A_1 \in U(\text{Type}_{Q_1}^\text{list})$ and $A_2 \in U(\text{Type}_{Q_2}^\text{list})$, we have

$$(A_1 \cdot (U_1Q_1 \equiv_{\text{quant}} U_2Q_2)) = (U_1(A_1^*)Q_1 \equiv_{\text{quant}} U_2Q_2) \quad \text{(2)}$$

and

$$(A_2 \cdot (U_1Q_1 \equiv_{\text{quant}} U_2Q_2)) = (U_1Q_1 \equiv_{\text{quant}} (U_2A_2^*)Q_2). \quad \text{(3)}$$

Recall that $\cdot$ is the lifting from Definition 15.

Predicates of this form occur, e.g., in the preconditions of rule QAPPLY1 below. $(U_{\text{vars}, Q_1} A_1 U_{\text{vars}, Q_1}^*)$ is the operator $A_1$, applied to quantum variables $Q_1$.) With this rule, we can collect the effect of several unitary quantum operations inside the quantum equality.

Notice that we require that $A_1, A_2$ are unitary. We leave it as an open problem to generalize this rule suitably to isometries.

**Lemma 28.** Fix disjoint lists $Q_1$ and $Q_2$ of distinct quantum variables, and operators $U_1 \in B(\text{Type}_{Q_1}^\text{list}, Z)$ and $U_2 \in B(\text{Type}_{Q_2}^\text{list}, Z)$ and a vector $\psi \in \ell^2[\text{Type}_{Q_1}^\text{list}]$. Assume $U_1^*U_2U_2^*U_1\psi = \psi$ (e.g., if $U_1, U_2$ are isometries) Then we have

$$(U_1Q_1 \equiv_{\text{quant}} U_2Q_2) \cap (\text{span}\{\psi\} \cdot Q_1) = (\text{span}\{\psi\} \cdot Q_1) \cap (\text{span}\{U_2^*U_1\psi\} \cdot Q_2) \quad \text{(4)}$$

Proc. ACM Program. Lang., Vol. 3, No. POPL, Article 33. Publication date: January 2019.
This lemma tells us (in the special case \( U_1 = U_2 = id \)) that the following two statements are equivalent:

- \( Q_1 \) and \( Q_2 \) have the same content, and \( Q_1 \) contains \( \psi \).
- \( Q_1 \) contains \( \psi \) and \( Q_2 \) contains \( \psi \).

**Lemma 29.** Fix disjoint lists \( Q_1 \) and \( Q_2 \) and \( R \) of distinct quantum variables, and operators \( U_1 \in B(\text{Type}_{\text{list}}^{Q_1}, Z) \) and \( U_2 \in B(\text{Type}_{\text{list}}^{Q_2}, Z) \) and a unit vector \( \psi \in \ell^2(\text{Type}_{\text{list}}^R) \). Then we have

\[
(U_1 Q_1 \equiv_{\text{quant}} U_2 Q_2) \land (\langle \text{span}\{\psi\}\rangle R) = \langle (U_1 \otimes' id) Q_1 R \rangle \equiv_{\text{quant}} (U_2 \otimes' \psi) Q_2 .
\]

In this equation, \( \otimes' \) refers to the usual positional (unlabeled) tensor product, not to the labeled tensor product defined in Section 2. And on the rhs, \( \psi \) is interpreted as a bounded operator from \( \mathbb{C} \) to \( \ell^2(\text{Type}_{\text{list}}^R) \).

This rule allows us to rewrite an equality between \( Q_1 \) and \( Q_2 \) into an equality between \( Q_1 R \) and \( Q_2 \) if we know the content of \( R \). It is the quantum analogue of the following obvious classical fact:

\[
f(x_1) = g(x_2) \land y_1 = y \quad \iff \quad (f \times id)(x_1, y_1) = (\lambda x. (x, y))(x_2) .
\]

## 5 QUANTUM RELATIONAL HOARE LOGIC (QRHL)

We now formally introduce our quantum Relational Hoare Logic qRHL. See the introduction (especially the discussion before **Definition 5**) for a motivation of the definition.

**Tagged variables.** In the following, there will always be a fixed set \( V \) of program variables, and programs mentioned in qRHL judgments will always have free variables in \( V \). In addition, we will use two disjoint copies of the set \( V \), called \( V_1 \) and \( V_2 \), consisting of the variables from \( V \) tagged with 1 or 2, respectively. These tagged variables will be necessary to distinguish between the variables of the left and right program in a qRHL judgment (e.g., in the pre- and postcondition of a qRHL judgment).

Formally, \( V_1 \), \( V_2 \) are disjoint sets with bijective variable renamings \( \text{id}_{x_i} : V \to V_i \) for \( i = 1, 2 \). We will write \( x_i \) for \( \text{id}_{x_i}(x) \).

For an expression \( e \) with \( \text{fv}(e) \subseteq V \), we write \( \text{id}_{x_i} e \) for the expression \( e \) with every classical variable \( x \) replaced by \( x_i \). Formally, \( \llbracket \text{id}_{x_i} e \rrbracket_m := \llbracket e \rrbracket_{m \circ (\text{id}_{x_i})^{-1}} \).

For a program \( c \) with \( \text{fv}(c) \subseteq V \), we write \( \text{id}_{x_i} c \) for the program \( c \) with every classical or quantum variable \( x \) in statements replaced by \( x_i \), and every expression \( e \) replaced by \( \text{id}_{x_i} e \).

**The definition.** We now have all the necessary language to make **Definition 5** formal:

**Definition 30 (Quantum Relational Hoare Judgments).** Let \( c_1, c_2 \) be programs with \( \text{fv}(c_1), \text{fv}(c_2) \subseteq V \). Let \( A, B \) be predicates over \( V_1V_2 \).

Then \( \{A\}c_1 \sim c_2 \{B\} \) holds iff for all \( (V_1, V_2)\)-separable \( \rho \in T^+_\text{eq}[V_1V_2] \) that satisfy \( A \), we have that there exists a \( (V_1, V_2)\)-separable \( \rho' \in T^+_\text{eq}[V_1V_2] \) that satisfies \( B \) such that \( \text{tr}_{V_2}^{V_1}[\rho] = \llbracket \text{id}_{x_1} c_1 \rrbracket \text{tr}_{V_2}^{V_1}[\rho] \) and \( \text{tr}_{V_2}^{V_1}[\rho'] = \llbracket \text{id}_{x_2} c_2 \rrbracket \text{tr}_{V_2}^{V_1}[\rho'] \).

(Recall in particular **Definition 10** for the definition of “predicates” and “satisfy”, and the preliminaries for the remaining notation.)

The following is a technical lemma that gives an alternative characterization of qRHL judgments in terms of deterministic initial values for the classical variables, and pure initial states for the quantum variables. This definition will simplify many proofs.

**Lemma 31 (qRHL for pure states).** \( \{A\}c \sim d \{B\} \) holds if:

For all \( m_1 \in \text{Type}_{V_1}^{\text{ext}} \), \( m_2 \in \text{Type}_{V_2}^{\text{ext}} \) and all normalized \( \psi_1 \in \ell^2[V_1^{\text{qu}}] \), \( \psi_2 \in \ell^2[V_2^{\text{qu}}] \) such that \( \psi_1 \otimes \psi_2 \in \llbracket A \rrbracket_{m_1m_2} \), there exists a \( (V_1, V_2)\)-separable \( \rho' \in T^+_\text{eq}[V_1V_2] \) such that:

Proc. ACM Program. Lang., Vol. 3, No. POPL, Article 33. Publication date: January 2019.
\[ \rho' \text{ satisfies } B, \text{ and } \text{tr}_{\text{cl}} \rho' = |\text{idx}_1 \text{d}|(\text{proj}(m_1) \otimes \text{proj}(\psi_1)), \text{ and } \text{tr}_{\text{quant}} \rho' = |\text{idx}_2 \text{d}|(\text{proj}(m_2) \otimes \text{proj}(\psi_2)). \]

5.1 Sound Rules for qRHL

From the definition of qRHL judgments, we can derive a number of reasoning rules for qRHL. Our reasoning rules are sound (each rule follows from the definition of qRHL judgments), but we make no claim that they are complete. (This situation is similar to that of pRHL, where no complete set of rules has been proposed either. Instead, the set of rules is chosen to cover all use cases that occur in common cryptographic proofs.)

We group the rules in three categories: In Figure 1, we list an excerpt of the general reasoning rules. These rules are not specific to individual statements (such as assignment or measurement), but instead describe structural properties of qRHL (such as the frame rule, or case distinction, etc.) In Figure 2, we list the rules specific to the classical statements in our language (e.g., assignment, sampling, if-statement). Those are very similar to those in pRHL, we omit a detailed discussion here. And in Figure 3, we list rules specific to the quantum statements (e.g., measurement, quantum initialization). We will shortly discuss those rules.

**Theorem 32.** *The rules in Figures 1, 2, and 3 are sound.*

Proofs are in the full version [Unruh 2018c].

**General rules.**

**Rule Seq** allows us to reason about the sequential composition of programs by introducing a predicate that has to hold in the middle of the execution of the two programs.

**Rule Equal** states that, if all classical and quantum variables of \( c \) are the same on the left and right before execution, they are the same after execution. This rule allows us to reason about unknown code \( c \) (adversary invocations).

**Rule Frame** formalizes the fact that if some predicate holds before execution of the programs, and the predicate refers only to variables not written in the programs, then it also holds after the execution. For example, if we know \( \{ A \} c \sim d \{ B \} \), then we can derive \( \{ A \cap R \} c \sim d \{ B \cap R \} \), assuming that \( R \) does not refer to any variables occurring in \( c, d \). (Actually, it is sufficient to assume that \( R \) only refers to classical variables that are not written by \( c, d \), and only to quantum variables that do not occur in \( c, d \). See the formal statement of the rule.)

This rule is useful for compositional analysis. If \( c, d \) are subprograms of larger programs, we can first analyze \( c, d \) (and derive some judgment \( \{ A \} c \sim d \{ B \} \)) and then use **rule Frame** to show that additional invariants \( R \) occurring in the analysis of the larger programs are preserved.

An important special case is where \( c \) is an unspecified program (modeling an adversary). Say we know that \( c \) only accesses variables \( x \) and \( q \). Then by combining **rule Frame** with **rule Equal**, we get \( \{ \text{cl}[x_1 = x_2] \cap (q_1 \equiv_{\text{quant}} q_2) \cap R \} c \sim c \{ \text{cl}[x_1 = x_2] \cap (q_1 \equiv_{\text{quant}} q_2) \cap R \} \) for \( R \) that does not refer to \( x, q \). (Formally, \( R \) is \( Y \)-local with \( x, q \notin Y \).) This means that when analyzing a pair \( d_1, d_2 \) of programs that both invoke \( c \), any invariant \( R \) between \( d_1 \) and \( d_2 \) is preserved by the call to the adversary \( c \) (assuming that \( \text{cl}[x_1 = x_2] \cap (q_1 \equiv_{\text{quant}} q_2) \) holds before the call to \( c \), i.e., that the adversary’s variables before invocation have the same state on both sides).

Since typically, we do not know anything about an adversary except variables it may or may not access, this is the only way to reason about adversaries.

---

13For classical variables \( X \), a program \( c \) is called \( X \)-readonly if for all \( x \in X \), all \( z \in \text{Type}_x \), all \( \rho \) that satisfy \( \text{cl}[x = z] \), we have that \([c](\rho)\) satisfies \( \text{cl}[x = z] \). Of course, in most cases one will use a simple sufficient syntactic criterion: if no variable from \( X \) occurs on the lhs of an assignment, sampling, or measurement, then the program is \( X \)-readonly.
**QrhlElimEq** allows us to deduce statements about probabilities from qRHL judgments. As the final goal of a proof in qRHL is to make statements about, e.g., attack probabilities, it is necessary to translate the judgments into statements about probabilities of certain events in programs. The rule **QrhlElimEq** allows us to show statements of the form $\Pr[e : c(\rho)] \leq \Pr[f : d(\rho)]$ (or analogous equalities) by proving qRHL judgments of the form $\{\text{Eqs}\}c \sim d\{\text{Eqs}\}$ where $\text{Eqs}$ is the predicate that states equality between all variables of $c$ and $d$. (And analogously for $\Pr[e : c(\rho)] = \Pr[f : d(\rho)]$.)

That is, to show that $e$ is less likely to occur in $c$ (with initial state $\rho_1$) than $f$ is likely to occur in $d$ (with initial state $\rho_2$), we need to show that $c$, $d$ can be related in such a way that $e$ implies $f$ after execution.

If additionally, we know that the initial predicate $\rho$ satisfies a certain predicate $A$, we can strengthen the precondition accordingly, see the formal statement of the rule. An application of rule **QrhlElimEq** is probably the last step (or first when performing backwards reasoning) in most qRHL-based proofs, since in a game-based proof, we usually compare the probabilities of events that hold in different games (programs) when running them with the same

---

**Conseq**

$$
A \subseteq A' \quad B' \subseteq B \quad \{A'\}c \sim d\{B'\}
$$

$$
\{A\}c \sim d\{B\}
$$

**SEQ**

$$
\{A\}c_1 \sim c_2\{B\} \quad \{B\}d_1 \sim d_2\{C\}
$$

$$
\{A\}c_1; d_1 \sim c_2; d_2\{C\}
$$

**EQUAL**

- $X$ is a list of classical variables
- $Y$ is a list of distinct quantum variables
- $c$ is $XY$-local
- $d$ is $X_2$-local
- $R$ is $Y_1Y_2'$-local
- $X_1 \cap Y_1$ and $X_2 \cap Y_2$ are classical

$$
\{\text{Cla}[X_1 = X_2] \cap (Y_1 \equiv_{\text{quant}} Y_2)\}c \sim c\{\text{Cla}[X_1 = X_2] \cap (Y_1 \equiv_{\text{quant}} Y_2)\}
$$

**Frame**

$$
\{A \cap R\}c \sim d\{B \cap R\}
$$

**QrhlElimEq**

$$
\rho \in T^{\alpha}_{c_{\text{q}}} \Rightarrow \rho \text{ satisfies } A
$$

$$
A \text{ is a list of classical variables} \quad Y \text{ is a list of distinct quantum variables} \quad c, d \text{ are } XY\text{-local}
$$

$$
\{\text{Cla}[X_1 = X_2] \cap (Y_1 \equiv_{\text{quant}} Y_2) \cap A_1 \cap A_2\}c \sim d\{\text{Cla}[^{\text{id}}(x) e \Rightarrow \text{id}(f)]\}
$$

**TransSimple**

$$
X_{\rho_i} := \text{id}(x_i) f^i(\rho_1), \quad Q_{\rho_i} := \text{id}(x_i) f^i(\rho_2) \text{ for } \rho = c, d, e, \ i = 1, 2
$$

$$
\{\text{Cla}[X_{c1} = X_{d2}] \cap (Q_{c1} \equiv_{\text{quant}} Q_{d2})\}c \sim d\{\text{Cla}[X_{c1} = X_{d2}] \cap (Q_{c1} \equiv_{\text{quant}} Q_{d2})\}
$$

$$
\{\text{Cla}[X_{d1} = X_{e2}] \cap (Q_{d1} \equiv_{\text{quant}} Q_{e2})\}d \sim e\{\text{Cla}[X_{d1} = X_{e2}] \cap (Q_{d1} \equiv_{\text{quant}} Q_{e2})\}
$$

$$
\{\text{Cla}[X_{c1} = X_{e2}] \cap (Q_{c1} \equiv_{\text{quant}} Q_{e2})\}c \sim e\{\text{Cla}[X_{c1} = X_{e2}] \cap (Q_{c1} \equiv_{\text{quant}} Q_{e2})\}
$$

Fig. 1. Rules for qRHL (general rules).
Figure 3
Assign1 is a simple version of transitivity, where the relationship between Sample1 If1 Rule While1 TransSimple Rules for quantum statements. While1 Rule application, measurement) we provide one or several rules (Trans powerful rule is probably too restrictive, but we included this simplified rule for illustrative purposes. A more that all classical and quantum variables are equal in pre- and postcondition. In many cases, equality then we expect to be able to conclude what the relationship between c be initialized by the programs.) A initial state. (And in most cases, we will have A = C la [true] because all relevant variables would be initialized by the programs.)

Rule Trans: If c stands in a certain relationship to d, and d stands in a certain relationship to e, then we expect to be able to conclude what the relationship between c and e is (transitivity). Rule TransSimple is a simple version of transitivity, where the relationship between c and d and e is that all classical and quantum variables are equal in pre- and postcondition. In many cases, equality is probably too restrictive, but we included this simplified rule for illustrative purposes. A more powerful rule Trans is given in the full version [Unruh 2018c].

Rules for quantum statements. For each quantum statement of the language (initialization, application, measurement) we provide one or several rules (Figure 3). Rules ending with 1 operate
only on the left program and assume that the right program is skip. The rules can be applied to a larger program by combining them with rule SEQ. The symmetric variants of those rules (with names ending in 2) are omitted for brevity. For some statements, we have joint rules. E.g., joint measurement for showing qRHL judgments with a measurement on both sides. These rules are used to prove qRHL judgments where the two programs are “in sync” (see rule JointMeasureSimple below).

Rule QINIT1 proves qRHL judgments that have a quantum initialization of the form \( Q \leftarrow e \) on the left side. We assume just a single skip statement on the right side, more complex programs can be first decomposed using rule SEQ.\(^{14}\) The precondition \((A \div e') \otimes \ell^2[Q']\) (with \(\div\) as in Definition 16) consists of those states that, when we replace the content of variables \(Q'\) by the state described by the expression \(e'\), we have a state in \(A\).

Here \(Q'\) and \(e'\) are derived from \(Q\) and \(e\) by certain natural isomorphisms: \(Q'\) is \(idx_1 Q\), i.e., all variables are indexed with 1. (To distinguish left and right variables in the pre- and postcondition.) And \(e'\) is \(U_{vars,Q'} idx_1 e\), i.e., we index the classical variables in \(e\) with 1, and then map the state returned by \(e\) from \(\ell^2(\text{Type}_{exp})\) (not labeled with quantum variables) into \(\ell^2[Q']\) (labeled with quantum variables). Similar natural conversions occur in the following rules as well, we will ignore them in our informal discussions.

Note that there is no joint rule for quantum initializations. That is because initialization does not make any probabilistic choices (like measurements do), hence a joint rule would not be any different from simply applying rules QINIT1 and QINIT2 consecutively.

Rule QAPPLY1 proves qRHL judgments that have a unitary quantum operation of the form \(\text{apply } e \text{ to } Q\) on the left side. (As with rule QINIT1, we assume skip on the right side, and there is an analogous omitted rule QAPPLY2 for the right side.)

\(^{14}\)Here we implicitly use the (trivial to prove) associativity of \(\div\) and the fact that that \(\text{skip} . c = c = c . \text{skip}\) (w.r.t. to the denotation of programs).
If \( e \) (and thus \( e' \)) which is lifted using Definition 15 is unitary, then the precondition can be \( e'' \cdot B \), because after applying \( e \) on \( Q \) in a state in \( e'' \cdot B \), we get a state in \( e' \cdot e'' \cdot B = B \). But since we also allow isometries \( e \) in quantum applications, we need to restrict \( B \) to those states that are in the image \( \text{im} \) of \( e' \), leading to the precondition \( e'' \cdot (B \cap \text{im} \ e') \).

As with quantum initialization, we have no joint rule for quantum application since we can simply apply rules QAPPLY1 and QAPPLY2 and get the same result.

**Rule Measure1** proves qRHL judgments that have a measurement of the form \( x \leftarrow \text{measure } Q \text{ with } e \) on the left side. (As with rule QINIT1 and rule QAPPLY1, we assume skip on the right side, and there is an analogous omitted rule QMEASURE2 for the right side.)

To satisfy the postcondition \( B \), we need the following things to hold in the precondition:

- \( e \) needs to be a total measurement. Otherwise, the probability that the left program terminates might be \( < 1 \), and the right program terminates with probability \( = 1 \), and then \( \{A\}x \leftarrow \text{measure } Q \text{ with } e \sim \text{skip}(B) \) cannot hold, no matter what the postcondition \( B \) is.

- For any possible outcome \( z \), we need that the post-measurement state after outcome \( z \) is in \( B\{z/x_1\} \) (i.e., \( B \) with \( x_1 \) set to be the outcome \( z \)). This is the case if the initial state lies in the complement of the image \( \text{im} \ e'_z \) (then the measurement will not pass), or if it lies in \( B \cap \text{im} \ e'_z \) (then it will pass and stay in \( B \)), or if it is a sum of states satisfying those two conditions. Here \( e'_z \) is the projector corresponding to outcome \( z \). Thus, for every outcome \( z \), we have the term \( (B\{z/x_1\} \cap \text{im} \ e'_z) + (\text{im} \ e'_z)^\perp \) in the precondition \( A \).

**Rule JointMeasureSimple** allows us to analyze two measurements that are performed “in sync”.

For example, consider a judgment of the following form:

\[
\{A'\}x \leftarrow \text{measure } q \text{ with } e \sim y \leftarrow \text{measure } q \text{ with } e\{\exists a[x_1 = y_2]\}
\]

where \( e := M \) is a measurement in the diagonal basis, and \( q \) is a qubit variable, and \( A' := \text{span}\{|00\rangle_{q_1q_2}\} \otimes \ell^2[V_1V_2 \setminus q_1q_2] \) is a precondition that ensures that \( q_1 \) and \( q_2 \) are both initialized as \( |0\rangle \). Then the measurement on each side would yield a uniformly random bit \( x_1 \) and \( y_2 \), respectively, and – analogous to the case where we just sample a uniformly random bit on both sides – we would expect to be able to “match” the equally-distributed random choices and show the postcondition \( \exists a[x_1 = y_2] \).

However, by applying rules Measure1 and Measure2, we will not be able to prove this since they treat the measurement outcome as non-deterministic (i.e., they ignore the distribution of the outcomes).

Thus we need rule JointMeasureSimple for this case. This rule requires the following things to hold in the precondition:

- For any possible outcome \( z \), we need that the post-measurement state after both measurements produce outcome \( z \) is in \( B\{z/x_1, z/y_2\} \) (i.e., \( B \) with \( x_1, y_2 \) set to the same outcome \( z \)). Analogous to rule Measure1 (but generalized to the application of two measurements), this means we need to satisfy in the precondition:

\[
(B\{z/x_1, z/y_2\} \cap \text{im} \ e'_{1z} \cap \text{im} \ e'_{2z}) + (\text{im} \ e'_{1z})^\perp + (\text{im} \ e'_{2z})^\perp
\]

for every \( z \). Here \( e'_{1z} \) is the projector for outcome \( z \) on variables \( Q'_1 \) and \( e'_{2z} \) the projector for outcome \( z \) on \( Q'_2 \).

In our example, this would become (we omit the \( \otimes \ell^2[\ldots] \) subterms for readability):

\[
(\exists a[z = z] \land \text{span}(H|z\rangle_{h_1}) \land \text{span}(H|z\rangle_{h_2})) + \text{span}(H|z\rangle_{h_1})^\perp + \text{span}(H|z\rangle_{h_2})^\perp
\]

\[
= \text{span}(H|z\rangle_{h_1} \otimes H|z\rangle_{h_2}) + \text{span}(H|z\rangle_{h_1})^\perp + \text{span}(H|z\rangle_{h_2})^\perp = \ell^2[V_1^{\text{qu}}V_2^{\text{qu}}]
\]
where $H$ is the Hadamard matrix $\frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$. (That this simplifies to $\ell^2[V_1^{\text{quant}}V_2^{\text{quant}}]$ is not unexpected: since in this example, the postcondition $B$ does not refer to $q_1$, $q_2$, the post-measurement state will always be in $B$, no matter what the initial state is.)

- The measurement on the left and right side need to be the same. In the general case, this leads to the condition $\mathcal{C}(\text{lab})[\text{id}_x e_1 = \text{id}_x e_2]$. In our example, this trivializes to $\mathcal{C}(\text{lab})[M = M] = \ell^2[V_1 V_2]$.

- Finally, in order for the two measurements to give the same outcome with the same probability, we need the measured state to be the same on both sides. This is captured by the quantum equality $Q_1^e \equiv_{\text{quant}} Q_2^e$. In our example: $q_1 \equiv_{\text{quant}} q_2$.

So, in our example, the precondition $A$ given by rule $\text{JointMeasureSimple}$ is simply $A = (q_1 \equiv_{\text{quant}} q_2)$, and we have \{ $q_1 \equiv_{\text{quant}} q_2$ $\}$ $x \leftarrow \text{measure } q$ with $e$ $\sim$ $y \leftarrow \text{measure } q$ with $e$ $\{ \mathcal{C}(\text{lab})[x_1 = y_2] \}$.

Since $A' \subseteq (q_1 \equiv_{\text{quant}} q_2)$ (because $|00\rangle_{q_1q_2}$ is invariant under swapping $q_1$ and $q_2$), we also have $\{ A' \} x \leftarrow \text{measure } q$ with $e$ $\sim$ $y \leftarrow \text{measure } q$ with $e$ $\{ \mathcal{C}(\text{lab})[x_1 = y_2] \}$ as desired.

Note that in contrast to rule $\text{Measure1}$, we do not require the measurements to be total. A more complex and more general rule $\text{JointMeasure}$ is deferred to the full version [Unruh 2018c].

6 EXAMPLES

6.1 EPR Pairs

We present a small example of a derivation in our logic. What we show is that, if we initialize two variables with an EPR pair, it does not matter whether we apply a Hadamard transform to the first or to the second variable.

The two programs we analyze are thus:

$$c := qr \quad \text{EPR; apply } H \text{ to } q. \quad \quad d := qr \quad \text{EPR; apply } H \text{ to } r.$$  

Here EPR := $1/\sqrt{2} \{ (0, 0) \} + 1/\sqrt{2} \{ 1, 1 \}$ is an EPR pair, and $H := \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ the Hadamard transform.

We want to show that both programs return the same quantum state, i.e., we want to show

$$\{ \mathcal{C}(\text{lab})[\text{true}] \} c \sim d(q_1 r_1 \equiv_{\text{quant}} q_2 r_2).$$  

(6)

We refer to the first/second statement of $c$ as $c_1, c_2$. Analogously for $d$. Let $I_0 := (q_1 r_1 \equiv_{\text{quant}} q_2 r_2)$.

By rule $\text{QApply1}$, we have $\{ I_1 \} c_2 \sim \text{skip}(I_0)$ with

$$I_1 := (H \triangleright q_1) \ast \cdot (I_0 \cap \text{im}(H \triangleright q_1)) \equiv (H \otimes \text{id})q_1 r_1 \equiv_{\text{quant}} q_2 r_2).$$

Here (*) follows by the fact that $\text{im}(H \triangleright q_1)$ is the full space (since $H$ is unitary), and Lemma 27.

By rule $\text{QApply2}$, we have $\{ I_2 \} \text{skip} \sim d_2(I_1)$ with

$$I_2 := (H \triangleright r_2) \ast \cdot (I_1 \cap \text{im}(H \triangleright r_2)) \equiv (H \otimes \text{id})q_1 r_1 \equiv_{\text{quant}} (\text{id} \otimes H)q_2 r_2).$$

Here (*) is simplified analogously as the previous step.

By rule $\text{QInit1}$, we have $\{ I_3 \} c_1 \sim \text{skip}(I_2)$ with

$$I_3 := I_2 \div U_{\text{vars} q_1 r_1} \text{EPR} \otimes \ell^2[q_1 r_1].$$

By rule $\text{QInit2}$, we have $\{ I_4 \} \text{skip} \sim d_1(I_3)$ with

$$I_4 := I_3 \div U_{\text{vars} q_2 r_2} \text{EPR} \otimes \ell^2[q_2 r_2].$$

Using three applications of rule $\text{Seq}$, those three qRHL judgments imply $\{ I_4 \} c \sim d(I_0)$. In order to show $\{ \mathcal{C}(\text{lab})[\text{true}] \} c \sim d(I_0)$ using $\text{Conseq}$, we need to show that $\mathcal{C}(\text{lab})[\text{true}] \subseteq I_4$.

---

15Here we implicitly use the (trivial to prove) fact that that $\text{skip}; c = c = c; \text{skip}$ (w.r.t. to the denotation of programs).
We have
\[ \mathsf{Cl}[\text{true}] \subseteq I_4 \]

Lemma 27
\[ \iff \mathsf{Cl}[\text{true}] \cap \text{span}\{U_{vars,q_2r_2,\text{EPR}} \otimes \ell^2[V_1^{\text{qu}}V_2^{\text{qu}} \setminus q_2r_2] \subseteq I_3 \]
\[ \iff \text{span}\{\text{EPR} \otimes q_2r_2\} \subseteq I_3 \]

Lemma 27
\[ \iff \text{span}\{\text{EPR} \otimes q_2r_2\} \cap \text{span}\{U_{vars,q_2r_2,\text{EPR}} \otimes \ell^2[V_1^{\text{qu}}V_2^{\text{qu}} \setminus q_1r_1] \subseteq I_2 \]
\[ \iff \text{span}\{\text{EPR} \otimes q_2r_2\} \cap \text{span}\{\text{EPR} \otimes q_1r_1\} \subseteq I_2 \]
\[ \iff \forall x. U_{vars,q_2r_2,\text{EPR}} \otimes U_{vars,q_1r_1,\text{EPR}} \otimes [x]_1^{\text{qu}}[x]_2^{\text{qu}}[x]_1r_1[x]_2r_2 \in I_2 = (H \otimes id)q_1r_1 \equiv_{\text{quant}} (id \otimes H)q_2r_2 \]
\[ \iff (H \otimes id)\text{EPR} = (id \otimes H)\text{EPR}. \]

The last line can be verified by elementary computation. Hence \( \mathsf{Cl}[\text{true}] \subseteq I_4 \) is true. (In fact, we can also prove the statement \( \mathsf{Cl}[\text{true}] \subseteq I_4 \) directly by explicit computation, but the involved matrices are too unwieldy for a pen-and-paper proof. However, in our tool such a computation can be performed automatically.)

From \( \mathsf{Cl}[\text{true}] \subseteq I_4 \) and \( \{I_4\}c \sim d\{I_0\} \), by rule CONSEQ, we get (6) as desired.

We have formalized this proof in our tool in the contributed file epr.qrhl.

6.2 Measurements

We extend the previous example to include a measurement of the two quantum registers in the computational basis. The fact that the quantum registers between the two programs are equal (in the sense of \( \equiv_{\text{quant}} \)) before the measurements will imply that the outcomes are the same. That is, with \( c, d \) as in the previous section, we define

\[ c' := c; x \leftarrow \text{measure qr with } M_{\text{class}}, \quad d' := d; x \leftarrow \text{measure qr with } M_{\text{class}}. \]

Here \( x \) has type \( \text{Type}_x = \mathbb{B} \times \mathbb{B} \), i.e., two bits, and \( M_{\text{class}} \) is a measurement in the computational basis on two qubits. (Formally, \( M_{\text{class}}((a, b)) := \text{proj}([(a, b)]) \).)

And we want to show:
\[ \langle \mathsf{Cl}[\text{true}] \rangle c' \sim d' \langle \mathsf{Cl}[x_1 = x_2] \rangle. \tag{7} \]

By rule JOINTMEASURESIMPLE (with \( Q_1 := Q_2 := \text{qr}, e_1 := e_2 := M_{\text{class}} \)), we have
\[ \{I_1\}e \sim e \langle \mathsf{Cl}[x_1 = x_2] \rangle \]

with
\[ I_1 := \mathsf{Cl}[M_{\text{class}} = M_{\text{class}}] \cap (q_1r_1 \equiv_{\text{quant}} q_2r_2) \cap \bigcap_{a,b \in \{0,1\}} (\mathsf{Cl}[z = z] \cap A_{ab} \cap A_{ab}^\perp) + (A_{ab}^\perp)^\perp + (A_{ab}^\perp)^\perp. \]

and \( A_{ab}^\perp := \text{im proj}([(a, b)])_{q_1r_1}. \)

Since \( M_{\text{class}} = M_{\text{class}} \) and \( z = z \) are trivially true, the above simplifies to
\[ I_1 = (q_1r_1 \equiv_{\text{quant}} q_2r_2) \cap \bigcap_{a,b \in \{0,1\}} (A_{ab} \cap A_{ab}^\perp) + (A_{ab}^\perp)^\perp + (A_{ab}^\perp)^\perp. \]

Furthermore, \( A_{ab}^\perp \cap A_{ab}^\perp + (A_{ab}^\perp)^\perp + (A_{ab}^\perp)^\perp = \ell^2[V_1 V_2]. \) This is shown by noting that every vector \( \psi \) is a linear combination of vectors in \( A_{ab}^\perp \cap A_{ab}^\perp + (A_{ab}^\perp)^\perp, (A_{ab}^\perp)^\perp, (A_{ab}^\perp)^\perp \) and then verifying the equation for each of those three cases. Thus the definition of \( I_1 \) simplifies to
\[ I_1 = (q_1r_1 \equiv_{\text{quant}} q_2r_2). \]

Hence \( \{q_1r_1 \equiv_{\text{quant}} q_2r_2\}e \sim e \langle \mathsf{Cl}[x_1 = x_2] \rangle. \) With (6) and rule SEQ, we get \( \{\mathsf{Cl}[\text{true}]\}c; e \sim d; e \langle \mathsf{Cl}[x_1 = x_2] \rangle \). Using the definitions of \( c', d', e \), we obtain our goal (7).\footnote{We have not formalized this proof in version 0.3 of the tool (the version archived with this paper) because it did not support JOINTMEASURESIMPLE yet. The current development version contains a formalization of the proof.}
6.3 Post-Quantum Cryptography

Our second example proof is the ROR-OT-CPA security of a simple one-time encryption scheme. This example illustrates how post-quantum cryptographic proofs may end up being very similar to classical pRHL proofs, except for a bit of additional syntactic clutter.

The setting. The encryption scheme is defined by

\[
\begin{align*}
\text{enc} : K \times M &\to M, \quad \text{enc}(k, m) := G(k) \oplus m \\
\text{dec} : K \times M &\to M, \quad \text{dec}(k, c) := G(k) \oplus c
\end{align*}
\]

where \( G : K \to M \) is a pseudorandom generator, \( k \) is the key, and \( m \) is the message (plaintext).

The ROR-OT-CPA security notion says, informally: The adversary cannot distinguish between an encryption of \( m \) and an encryption of a random message, even if the adversary itself chooses \( m \). More formally:

**Definition 33 (ROR-OT-CPA advantage).** For a stateful adversary \( A_1, A_2 \), let

\[
\text{Adv}_{\text{ROR}}^{A_1, A_2} := \left| \Pr[b = 1 : k \gets \mathcal{U}, \ m \gets A_1(), \ c \gets \text{enc}(k, m), \ b \gets A_2(c)] - \Pr[b = 1 : k \gets \mathcal{U}, \ m \gets A_1(), \ r \gets \mathcal{U}, \ c \gets \text{enc}(k, r), \ b \gets A_2(c)] \right|
\]

where \( \mathcal{U} \) is the uniform distribution (on the domain of the respective variable), and the notation \( \Pr[e : G] \) denotes the probability that \( e \) holds after executing the instructions in \( G \).

Analogously, we define pseudorandomness of \( G : K \to M \) by defining the PRG advantage of \( G \):

**Definition 34 (PRG advantage).** For an adversary \( B \), let

\[
\text{Adv}_{\text{PRG}}^{B} := \left| \Pr[b = 1 : s \gets \mathcal{U}, \ r \gets G(s), \ b \gets B(r)] - \Pr[b = 1 : r \gets \mathcal{U}, \ b \gets B(r)] \right|
\]

What we want to show is the following well-known fact: “If \( G \) is pseudorandom, then enc is ROR-OT-CPA.” In other words, if \( \text{Adv}_{\text{ROR}}^{A} \) is big for some efficient adversary \( A \), then \( \text{Adv}_{\text{PRG}}^{B} \) is big for some related efficient adversary \( B \). In the present case, we can even show the stronger result \( \text{Adv}_{\text{ROR}}^{A} = \text{Adv}_{\text{PRG}}^{B} \) for some \( B \). This is shown by proving that for a suitable \( B \), the lhs of \( \text{Adv}_{\text{ROR}}^{A} \) equals the lhs of \( \text{Adv}_{\text{PRG}}^{B} \), and the rhs of \( \text{Adv}_{\text{ROR}}^{A} \) equals the rhs of \( \text{Adv}_{\text{PRG}}^{B} \). Since our formalism does not include procedures with parameters and return values, we first need to rewrite the games (program fragments) in the definitions of \( \text{Adv}_{\text{ROR}}^{A} \) and \( \text{Adv}_{\text{PRG}}^{B} \). This is done by defining \( A_1 \) to be an arbitrary adversary that can access the output variable \( m \), and \( A_2 \) to be one that can access the input/output variables \( c, b \). In addition, they can keep an internal state, thus \( A_1 \) and \( A_2 \) have access to some additional classical variables \( C \) and (very important for the post-quantum setting) quantum variables \( Q \). That is, \( A_1 \) has free variables \( m, C, Q \), and \( A_2 \) has free variables \( c, b, C, Q \). We can then write the lhs of \( \text{Adv}_{\text{ROR}}^{A} \) and \( \text{Adv}_{\text{PRG}}^{B} \) as the following programs:

\[
G_1 :\begin{align*}
\text{k} &\gets \mathcal{U}; \ A_1; \text{c} \gets \text{enc(k, m)}; \ A_2
\end{align*}
\quad G_2 :\begin{align*}
\text{s} &\gets \mathcal{U}; \ r \gets G(s); \ B
\end{align*}
\]

where \( k, r, c, s, m, b \) are classical variables. (We omit the proof of the equality of the right hand sides from this example.) Furthermore, we need to chose an efficient adversary \( B \):

\[
B : \begin{align*}
A_1; \text{c} &\gets r \oplus m; \ A_2
\end{align*}
\]

We now need to show \( \Pr[b = 1 : G_1(r)] = \Pr[b = 1 : G_2(r)] \) for arbitrary initial state \( r \). (See Definition 7 for the notation \( \Pr[\cdot : (\rho)] \).) For this, we first show \( (I)G_1 \sim G_2[\chi_{0[b_1 = b_2]}] \) for a suitable precondition \( A \). To make the following calculation more compact, let \( xy \ldots \) mean \( x_1 = x_2 \land y_1 = y_2 \land \ldots \) for classical \( x, y, \ldots \), and let \( \equiv_{\text{quant}} \) mean \( Q_1 \equiv_{\text{quant}} Q_2 \). We perform a
backward reasoning, starting with the desired post-condition, and working out way backward
through the statements in $G_1, G_2$ (sometimes individually, sometimes pairwise). Note that the
precondition of each line matches the postcondition of the next one.

\[
\{ \text{Cla}[b \cdot c] \cap Q \} A_2 \sim A_2 \{ \text{Cla}[b] \} 
\]

\[
\{ \text{Cla}[e \cdot k_1, m_1] = c_2 \land b \cdot c \cap Q \} c \leftarrow e \cdot (k, m) \sim \text{skip} \{ \text{Cla}[b \cdot c] \cap Q \}
\]

\[
\{ \text{Cla}[e \cdot k_1, m_1] = r_2 \oplus m_2 \land b \cdot c \cap Q \} \text{skip} \sim r \leftarrow m \{ \text{Cla}[e \cdot k_1, m_1] = c_2 \land b \cdot c \cap Q \}
\]

\[
\{ \text{Cla}[G(k_1) = r_2 \land b \cdot c] \cap Q \} \subseteq \{ \text{Cla}[e \cdot k_1, m_1] = r_2 \oplus m_2 \land b \cdot c] \cap Q \}
\]

\[
\{ \text{Cla}[G(k_1) = r_2 \land b \cdot c] \cap Q \} \subseteq A_1 \{ \text{Cla}[G(k_1) = r_2 \land b \cdot c] \cap Q \}
\]

\[
\{ A \} k \leftarrow U \sim s \leftarrow U \{ \text{Cla}[G(k_1) = G(s_2) \land b \cdot c] \cap Q \}
\]

In the last step (JointSample), we define $\mu$ to be the uniform distribution on pairs $(z, z)$. Using $f := \mu$ in rule JointSample, we get the precondition

\[
A := \text{Cla}[\text{marginal}_1(\mu) = U \land \text{marginal}_2(\mu) = U] \cap \bigcap_{(z_1, z_2) \in \text{supp } \mu} \{ \text{Cla}[G(z_1) = G(z_2) \land b \cdot c] \cap Q \}
\]

\[
\text{Cla}[\forall (z_1, z_2) \in \text{supp } \mu. G(z_1) = G(z_2) \land b \cdot c] \cap Q = \text{Cla}[b \cdot c] \cap Q \supseteq \text{Cla}[k_{\text{rcs}} b \cdot c] \cap Q
\]

Here the first expression after := is what rule JointSample gives us, the next equality holds since both marginals of $\mu$ are the uniform distribution, and by Lemma 21. The last equality is elementary first-order logic (using the fact that $\text{supp } \mu$ contains only pairs $(z, z)$).

Combining all those judgments using rules Seq and Conseq, we get

\[
\{ \text{Cla}[k_{\text{rcs}} b \cdot c] \cap Q \} G_1 \sim G_2 \{ \text{Cla}[b] \}
\]

By rule QRHLElMEQ, this implies

\[
\Pr[b = 1 : G_1(\rho)] = \Pr[b = 1 : G_2(\rho)]
\]

as desired.

Notice how each predicate in the above calculation is of the form $\text{Cla}[e] \cap (Q_1 \equiv_{\text{quant}} Q_2)$ for some classical expression $e$, and that the $(Q_1 \equiv_{\text{quant}} Q_2)$-part of the predicates is not touched by the rules.\footnote{Rule JointSample gave us a more complex precondition but we can easily see that the precondition of JointSample can always be simplified to something of the form $\text{Cla}[\ldots] \cap Q$ with steps analogous to what we did here, provided the postcondition is of the form $\text{Cla}[\ldots] \cap Q$. In fact, we could formulate a special case of JointSample that directly has pre- and postcondition of the form $\text{Cla}[\ldots] \cap Q$.
}

This means that, as long as we treat the quantum adversary as a black box, and no other quantum operations are performed in our games, the reasoning in qrHRL will be almost identical to that in pRHL. This is good news because it means that we need to deal with the extra complexity of quantum mechanics only in those cases where we would have to deal with them in a pen-and-paper proof as well.

We have formalized this proof in our tool in the contributed file prg-enc-ror-cpa.qrl.

**Acknowledgments**

We thank Gilles Barthe, Tore Vincent Carstens, François Dupressoir, Benjamin Gregoire, Yangjia Li, Pierre-Yves Strub for valuable discussions. This work was supported by institutional research funding IUT2-1 of the Estonian Ministry of Education and Research, by the Estonian Centre of Excellency in IT (EXCITE) funded by ERDF, and by the Air Force Office of Scientific Research through the project “Verification of quantum cryptography” (AOARD Grant FA2386-17-1-4022). We also used Sage [Stein et al. 2014] for calculations and experiments, and the Sage Cluster funded by National Science Foundation Grant No. DMS-0821725.
REFERENCES

Martin Abadi and Phillip Rogaway. 2002. Reconciling two views of cryptography (The computational soundness of formal encryption). J Cryptology 15, 2 (2002), 103–127.

Andris Ambainis, Ansis Rosmanis, and Dominique Unruh. 2014. Quantum Attacks on Classical Proof Systems (The Hardness of Quantum Rewinding). In FOCS 2014. IEEE, 474–483.

Ebrahim Ardeshir-Larjani, Simon J. Gay, and Rajagopal Nagarajan. 2013. Equivalence Checking of Quantum Protocols. In TACAS 2013. Springer, 478–492. https://doi.org/10.1007/978-3-642-36742-7_33

Ebrahim Ardeshir-Larjani, Simon J. Gay, and Rajagopal Nagarajan. 2014. Verification of Concurrent Quantum Protocols by Equivalence Checking. In TACAS 2014. Springer, 500–514. https://doi.org/10.1007/978-3-642-54862-8_42

Gilles Barthe, François Dupressoir, Benjamin Grégoire, César Kunz, Benedikt Schmidt, and Pierre-Yves Strub. 2014. EasyCrypt: A Tutorial. In FOSAD 2012/2013 Tutorial Lectures. Springer, 146–166. https://doi.org/10.1007/978-3-319-10082-1_6

Gilles Barthe, Thomas Espitau, Benjamin Grégoire, Justin Hsu, Léo Stefanesco, and Pierre-Yves Strub. 2015. Relational Reasoning via Probabilistic Coupling. In Proceedings of the 20th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning - Volume 9450 (LPAR-20 2015). Springer-Verlag, Berlin, Heidelberg, 387–401. https://doi.org/10.1007/978-3-662-48899-7_27

Gilles Barthe, Benjamin Grégoire, Sylvain Heraud, and Santiago Zanella Béguelin. 2011a. Computer-Aided Security Proofs for the Working Cryptographer. In Crypto 2011 (LNCS), Vol. 6841. Springer, 71–90.

Gilles Barthe, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. 2017. Coupling Proofs Are Probabilistic Product Programs. In Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL 2017). ACM, New York, NY, USA, 161–174. https://doi.org/10.1145/3009837.3009896

Gilles Barthe, Benjamin Grégoire, Yassine Lakhnech, and Santiago Zanella Béguelin. 2011b. Beyond Provable Security Verifiable IND-CCA Security of OAEP. In CT-RSA 2011 (LNCS), Vol. 6558. Springer, 180–196.

Gilles Barthe, Benjamin Grégoire, and Santiago Zanella Béguelin. 2009. Formal Certification of Code-Based Cryptographic Proofs. In POPL 2009. ACM, 90–101. https://doi.org/10.1145/1480881.1480894

David A. Basin, Andreas Lochbihler, and S. Reza Sefidgar. 2017. CryptHOL: Game-based Proofs in Higher-order Logic. IACR ePrint 2017/753.

Mihir Bellare and Phillip Rogaway. 1994. Optimal Asymmetric Encryption. In Eurocrypt ’94 (LNCS), Vol. 950. Springer, 92–111.

Mihir Bellare and Phillip Rogaway. 2006. The Security of Triple Encryption and a Framework for Code-Based Game-Playing Proofs. In Proceedings of the 40th Annual ACM Symposium on Principles of Programming Languages (POPL 2006). ACM, Vol. 2006 (LNCS), Vol. 4004. Springer, 409–426.

Charles H. Bennett and Gilles Brassard. 1984. Quantum cryptography: Public-key distribution and coin tossing. In Computers, Systems and Signal Processing. IEEE, 175–179.

Nick Benton. 2004. Simple Relational Correctness Proofs for Static Analyses and Program Transformations. In POPL ’04. ACM, 14–25. https://doi.org/10.1145/964001.964003

Matthias Berg. 2013. Formal verification of cryptographic security proofs. Ph.D. Dissertation. Saarland University.

Bruno Blanchet. 2006. A Computationally Sound Mechanized Prover for Security Protocols. In Security and Privacy. IEEE, 140–154.

Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. 2011. Random oracles in a quantum world. In Asiacrypt 2011. Springer, 41–69. https://doi.org/10.1007/978-3-642-25385-0_3

CertiCrypt authors. 2013. CertiCrypt: Computer-Aided Cryptographic Proofs in Coq. http://certicrypt.gforge.inria.fr/. Accessed 2018-10-24.

Rohit Chadha, Paulo Mateus, and Amilcar Sernadas. 2006. Reasoning About Imperative Quantum Programs. ENTCS 158 (May 2006), 19–39. https://doi.org/10.1016/j.entcs.2006.04.003

Boris S. Cirel’son. 1980. Quantum generalizations of Bell’s inequality. Letters in Mathematical Physics 4, 2 (01 Mar 1980), 93–100. https://doi.org/10.1007/BF00417500

John F. Clauser, Michael A. Horne, Abner Shimony, and Richard A. Holt. 1969. Proposed Experiment to Test Local Hidden-Variable Theories. Phys. Rev. Lett. 23 (Oct 1969), 880–884. Issue 15. https://doi.org/10.1103/PhysRevLett.23.880

Richard Cleve, Peter Høyer, Ben Toner, and John Watrous. 2004. Consequences and limits of nonlocal strategies. In CCC 2004. 236–249. https://doi.org/10.1109/CCC.2004.1313847

CryptoVerif authors. 2018. CryptoVerif: Cryptographic protocol verifier in the computational model. http://prosecco.gforge.inria.fr/personal/bblanche/cryptoverif/. Accessed 2018-10-24.

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

Danny Dolev and Andrew C. Yao. 1983. On the Security of Public Key Protocols. Transactions on Information Theory 29, 2 (1983), 198–208.

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

Proc. ACM Program. Lang., Vol. 3, No. POPL, Article 33. Publication date: January 2019.
Quantum Relational Hoare Logic

Yuan Feng and Mingsheng Ying. 2015. Toward Automatic Verification of Quantum Cryptographic Protocols. In CONCUR 2015 (LIPIcs), Vol. 42. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 441–455. https://doi.org/10.4230/LIPIcs.CONCUR.2015.441

Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. 2001. RSA-OAEP Is Secure under the RSA Assumption. In Crypto '01 (LNCS), Vol. 2139. Springer, 260–274.

Eiichiro Fujisaki, Tatsuaki Okamoto, David Pointcheval, and Jacques Stern. 2004. RSA-OAEP Is Secure under the RSA Assumption. J Cryptology 17, 2 (2004), 81–104.

Yoshihiko Kakutani. 2009. A Logic for Formal Verification of Quantum Programs. In ASIAN 2009, Anupam Datta (Ed.). Springer, Berlin, Heidelberg, 79–93.

Jonathan Katz and Yehuda Lindell. 2014. Introduction to Modern Cryptography, Second Edition. Taylor & Francis.

Takahiro Kubota, Yoshihiko Kakutani, Go Kato, Yasuhiro Kawano, and Hideki Sakurada. 2012. Application of a Process Calculus to Security Proofs of Quantum Protocols. In FCS ’12, 141–147.

Takahiro Kubota, Yoshihiko Kakutani, Go Kato, Yasuhiro Kawano, and Hideki Sakurada. 2013. Automated Verification of Equivalence on Quantum Cryptographic Protocols. In SCSS 2013 (EPiC Series in Computing), Vol. 15. EasyChair, 64–69.

Michael A. Nielsen and Isaac L. Chuang. 2010. Quantum Computation and Quantum Information (10th anniversary ed.). Cambridge University Press, Cambridge.

Tobias Nipkow, Larry Paulson, and Markus Wenzel. 2002. Isabelle/HOL: A Proof Assistant for Higher-Order Logic. LNCS, Vol. 2283. Springer.

Adam Petcher and Greg Morrisett. 2015. The Foundational Cryptography Framework. In POST 2015. Springer, 53–72. https://doi.org/10.1007/978-3-662-46666-7_4

Peter W. Shor and John Preskill. 2000. Simple Proof of Security of the BB84 Quantum Key Distribution Protocol. Phys. Rev. Lett. 85, 2 (Jul 2000), 441–444. https://doi.org/10.1103/PhysRevLett.85.441

Victor Shoup. 2002. OAEP Reconsidered. J Cryptology 15, 4 (2002), 223–249.

William A. Stein et al. 2014. Sage Mathematics Software (Version 5.12). The Sage Development Team. http://www.sagemath.org.

Dominique Unruh. 2012. Quantum Proofs of Knowledge. In Eurocrypt 2012 (LNCS), Vol. 7237. Springer, 135–152.

Dominique Unruh. 2015a. Non-interactive zero-knowledge proofs in the quantum random oracle model. In Eurocrypt 2015, Vol. 9057. Springer, 755–784. Full version is IACR ePrint 2014/587.

Dominique Unruh. 2015b. Revocable quantum timed-release encryption. J ACM 62, 6 (2015), 49:1–49:76.

Dominique Unruh. 2018a. chsh.ec. https://raw.githubusercontent.com/dominique-unruh/qrhl-tool/v0.3/examples/chsh.ec. Also contained in [Unruh 2018c].

Dominique Unruh. 2018b. Chsh.thy. https://raw.githubusercontent.com/dominique-unruh/qrhl-tool/v0.3/examples/Chsh.thy. Also contained in [Unruh 2018c].

Dominique Unruh. 2018c. Verification tool for quantum relational Hoare Logic (artifact accompanying this paper). Available in the ACM Digital Library. https://doi.org/10.1145/3291619

Dominique Unruh. 2018a. dominique-unruh/qrhl-tool: Prototype proof assistant for qRHL. GitHub. https://github.com/dominique-unruh/qrhl-tool Also contained in [Unruh 2018c].

Dominique Unruh. 2018b. qrhl-tool binary distribution, version 0.3. https://github.com/dominique-unruh/qrhl-tool/releases/tag/v0.3 Also contained in [Unruh 2018c].

Dominique Unruh. 2018c. Quantum Relational Hoare Logic. arXiv:1802.03188 [quant-ph].

John Watrous. 2009. Zero-Knowledge against Quantum Attacks. SIAM J Comput 39, 1 (2009), 25–58.

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

Mark Zhandry. 2012a. How to Construct Quantum Random Functions. In FOCS 2013. IEEE Computer Society, Los Alamitos, CA, USA, 679–687. https://doi.org/10.1109/FOCS.2012.37 Online version is IACR ePrint 2012/182.

Mark Zhandry. 2012b. Secure Identity-Based Encryption in the Quantum Random Oracle Model. In Crypto 2012 (LNCS), Vol. 7417. Springer, 758–775. https://doi.org/10.1007/978-3-642-32009-5_44

Li Zhou, Shenggang Ying, Nengkun Yu, and Mingsheng Ying. 2018. Quantum Coupling and Strassen Theorem. arXiv:1803.10393 [quant-ph].