RapunSL: Untangling Quantum Computing with Separation, Linear Combination and Mixing

YUSUKE MATSUSHITA[*], Kyoto University, Japan

KENGO HIRATA[*], University of Edinburgh, United Kingdom and Kyoto University, Japan

RYO WAKIZAKA, Kyoto University, Japan

EMANUELE D'OSUALDO, University of Konstanz, Germany

Quantum Separation Logic (QSL) has been proposed as an effective tool to improve the scalability of deductive reasoning for quantum programs. In QSL, separation is interpreted as *disentanglement*, and the frame rule brings a notion of *entanglement-local* specification (one that only talks about the qubits entangled with those acted upon by the program). In this paper, we identify two notions of locality unique to the quantum domain, and we construct a novel quantum separation logic, RapunSL, which is able to soundly reduce reasoning about superposition states to reasoning about pure states (*basis-locality*), and reasoning about mixed states arising from measurement to reasoning about pure states (*outcome-locality*). To do so, we introduce two connectives, linear combination and mixing, which together with separation provide a dramatic improvement in the scalability of reasoning, as we demonstrate on a series of challenging case studies.

CCS Concepts: • **Theory of computation** → **Separation logic**; *Quantum computation theory*.

Additional Key Words and Phrases: quantum separation logic, entanglement, quantum program verification

1 Introduction

Separation logic [O'Hearn and Pym 1999] represents a foundational breakthrough in the scalability of deductive verification methods. By introducing a resource-based form of *local* reasoning, separation logic allows program specifications to describe behaviour solely in terms of the resources a program fragment accesses, without reference to the rest of the program state [Ishtiaq and O'Hearn 2001; O'Hearn et al. 2001; Reynolds 2002]. This is formalized through the *frame rule* (HOARE-FRAME in Fig. 1), which guarantees that verified properties of a program fragment remain valid when executed in a larger context, provided the context's resources are *separate*. By giving different interpretations to the notion of "separable resources", separation logic has been shown to support a wide array of different computational phenomena, from the heap to concurrency [O'Hearn 2007; Brookes 2007; Brookes and O'Hearn 2016], to probabilistic non-determinism [Barthe et al. 2019a; Bao et al. 2021; Li et al. 2023, 2024a; Bao et al. 2025].

Recently, the separation logic paradigm has been applied to *quantum computation* [Zhou et al. 2021; Le et al. 2022; Deng et al. 2024; Su et al. 2024]. In the quantum domain, separation has been

---

[*] The first two authors contributed equally to this work.

Authors' Contact Information: Yusuke Matsushita, Kyoto University, Kyoto, Japan, ymat@fos.kuis.kyoto-u.ac.jp; Kengo Hirata, University of Edinburgh, Edinburgh, United Kingdom and Kyoto University, Kyoto, Japan, k.hirata@sms.ed.ac.uk; Ryo Wakizaka, Kyoto University, Kyoto, Japan, wakizaka@fos.kuis.kyoto-u.ac.jp; Emanuele D'Osualdo, University of Konstanz, Konstanz, Germany, emanuele.dosualdo@uni-konstanz.de.

HOARE-FRAME
$$\frac{\{P\}\,C\,\{Q\}}{\{P * R\}\,C\,\{Q * R\}}$$

HOARE-SUM
$$\frac{\{P\}\,C\,\{Q\} \qquad \{P'\}\,C\,\{Q'\}}{\{P + P'\}\,C\,\{Q + Q'\}}$$

HOARE-MIX
$$\frac{\{P\}\,C\,\{Q\} \qquad \{P'\}\,C\,\{Q'\}}{\{P \,_0{\oplus}_1^t\, P'\}\,C\,\{Q \,_0{\oplus}_1^t\, Q'\}}$$

Fig. 1. The three rules of RapunSL embodying the three locality principles.

interpreted as *disentanglement*: roughly speaking, $P * Q$ describes a configuration where the qubits described by P are not entangled with the ones in Q. Since a quantum program acting on some qubits $\bar{\mathsf{x}}$ would not affect any disentangled qubits $\bar{\mathsf{y}}$, the frame rule is valid, improving the modularity of reasoning. For example, in proving $\{\mathsf{x} \mapsto |\psi\rangle * \mathsf{y} \mapsto |\phi\rangle\}\,U[x]\,\{\mathsf{x} \mapsto U\,|\psi\rangle * \mathsf{y} \mapsto |\phi\rangle\}$ one can apply HOARE-FRAME framing $\mathsf{y} \mapsto |\phi\rangle$ and reduce reasoning to the simpler $\{\mathsf{x} \mapsto |\psi\rangle\}\,U[x]\,\{\mathsf{x} \mapsto U\,|\psi\rangle\}$ as one would do in an informal proof. This allows one to focus only on the relevant state for a proof of a component, and still reuse the specification when used in larger contexts. We dub this *entanglement-locality*, as it allows one to focus only on the entangled part.

The starting observation of this paper is that quantum computation adds two unique ways to add context to some computation, *superposition* and *measurement*, and each of them grants a fundamentally new notion of locality which cannot be captured by separation alone.

Basis-locality. A qubit's state is, in general, a superposition $\alpha\,|0\rangle + \beta\,|1\rangle$ of the two classical states $|0\rangle$ and $|1\rangle$ (the basis vectors). Quantum gates, the basic building blocks of quantum computation, act on qubits as unitary operators U: their effect on a state $\alpha\,|0\rangle + \beta\,|1\rangle$ is entirely determined by their effect on $|0\rangle$ and $|1\rangle$, i.e., $U(\alpha\,|0\rangle + \beta\,|1\rangle) = \alpha(U\,|0\rangle) + \beta(U\,|1\rangle)$. This suggests, in addition to disentanglement, a second notion of locality, which we call *basis-locality*, which would allow us to focus on the effect of a program on the basis states, and extrapolate its effect to a superposition.

Outcome-locality. In the quantum world, the act of measuring is a delicate affair: measuring a qubit has the very global effect of making its superposition state probabilistically collapse to a classical state. The state after a measurement can thus be described using a so-called *mixed state*, i.e., a probabilistic ensemble of pure states. The effect of a program continuing after a measurement, however, would be entirely determined by its effect on each of the possible outcomes. This suggests a third notion of locality, which we call *outcome-locality*, where a specification on (potentially pure) states can be lifted to a specification on mixed states.

Unfortunately, none of the quantum logics in the literature has achieved all three locality principles within one logic. The quantum separation logics of Zhou et al. [2021] and Le et al. [2022] support only entanglement-locality. Recent work by Deng et al. [2024] additionally supports outcome-locality, but not basis-locality. This is not for lack of imagination: these logics are built on models that are fundamentally incompatible with basis-locality or outcome-locality.

What we set out to find in this paper is a way to soundly enable all these three locality principles in a program logic for quantum computation, and to articulate the patterns of reasoning they unlock. Due to the extremely subtle interaction of all these three mechanisms—(dis)entanglement, superposition, and measurement—the natural models of assertions all fail. Our solution starts from an analysis of such failures. We identify two sources of incompatibility with the three locality principles in the models used in the literature. The first pertains to the level of abstraction of assertions on pure states. The logics of Zhou et al. [2021] and Ying [2012] use global-phase-insensitive assertions, which is natural considering that measurements are insensitive to global-phase changes. This choice, however, impedes basis-locality: the sum of vectors needed to form a superposition is only meaningful when the global phase is tracked.

The second, and more serious, incompatibility involves the treatment of measurements. The issue is that measurements do not fulfil, strictly speaking, the basis-locality principle, as their

effect is *not* fully determined by their effect on basis states. On a classical state, a measurement is a no-op. However, when applied to a state in a superposition, the measurement collapses it, probabilistically, to a classical state. On the face of it, this seems to suggest that there is a fundamental incompatibility between measurements and basis-locality. In fact, this is indeed the case in the (global-phase-sensitive) model of Le et al. [2022].

The main contribution of this paper is to show that this apparent conflict can be resolved by introducing a new quantum separation logic called RapunSL. We identify the handling of mixed states as the source of the conflict: In the logic of Le et al. [2022], assertions talk, just like in any traditional separation logic, about *single* pure states, so specifications can only assert facts that apply to *every* outcome. In RapunSL, assertions are global-phase-sensitive and predicate over the whole mixed state at once. More precisely, RapunSL includes three logical connectives: *separation* ($P * Q$) representing disentanglement, *sum* ($P + Q$) representing superposition, and *mixing* ($P \,_0\oplus_1^t Q$) joining two outcomes into a mixed state. The conflict is resolved because in our model, a measurement on a classical state still gives us a mixed state where one outcome is a degenerate zero-probability state, and is therefore distinguishable from a no-op. The three locality principles are then embodied by the three rules of Fig. 1: HOARE-FRAME for entanglement-locality, HOARE-SUM for basis-locality, and HOARE-MIX for outcome-locality.

Contributions. The main contributions of this paper are:
(1) A new *assertion language* to describe mixed states using separation, sum and mixing, and its supporting model.
(2) A thorough study of the rich interactions between the new connectives. In particular, we study the distributivity and interchange properties that hold in our model.
(3) A sound program logic, RapunSL, supporting all three locality principles.
(4) Case studies evaluating how effective local reasoning is in RapunSL, and illustrating the new proof patterns available in it.

Outline. We start with an informal overview § 2 explaining the key ideas. Then, after setting things up in § 3, we formalize our logic and prove its soundness in § 4. We present case studies in § 5 and discuss key topics in § 6. Finally, § 7 reviews related work and § 8 concludes with future work. All omitted details and proofs can be found in the Appendix.

2 Overview of RapunSL
In this section, we highlight the key ideas unlocking the development of RapunSL.

2.1 Handling Superposition, Compositionally
To motivate the problem with superposition, imagine we are given the task of designing a quantum circuit C that should implement some Boolean 1-to-1 function $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$ of n qubits x̄, using basic quantum gates, e.g., Toffoli gates. Sometimes, some auxiliary qubits are necessary to do so. A well-understood technique for managing auxiliary state is to employ so-called *dirty qubits* [Häner et al. 2017; Gidney 2018; Nie et al. 2024]: a qubit tmp that serves as auxiliary workspace, and is not assumed to be in any particular state (hence "dirty"). As long as tmp is returned to its original state once the computation is done, even if it was in a superimposed state with other qubits in the context, the only state change would be in x̄.

Informally, one would reason about C as follows. Since C is implemented with linear operators, it is itself linear. Therefore, it is sufficient to analyse its behaviour on *classical* states for x̄ and tmp, prove that x̄ is correctly transformed as dictated by f, and that tmp is restored to its input state.

Then, by linearity, any state in a superposition would simply see the operator U_f (the unitary lifting of f) applied to x̄, and the rest of the qubits left untouched.

More formally, the argument would start by proving the correctness of C on a classical state, obtaining the following triple:

$$\forall \bar{b} \in \{0,1\}^n. \forall t \in \{0,1\}. \left\{ \bar{x} \mapsto |\bar{b}\rangle * \mathsf{tmp} \mapsto |t\rangle \right\} C \left\{ \bar{x} \mapsto |f(\bar{b})\rangle * \mathsf{tmp} \mapsto |t\rangle \right\} \tag{1}$$

The challenge would then be to reuse the triple in a context where there are some other qubits ȳ, and all the qubits are in some arbitrary superimposed state $|\psi\rangle$, i.e., to deduce from (1) the triple:

$$\left\{ (\bar{x}, \mathsf{tmp}, \bar{y}) \mapsto |\phi\rangle \right\} C \left\{ (\bar{x}, \mathsf{tmp}, \bar{y}) \mapsto (U_f \otimes \mathsf{id}_{\mathsf{tmp},\bar{y}}) |\phi\rangle \right\} \tag{2}$$

To perform such a deduction, RapunSL provides two rules encoding the linearity argument:

HOARE-SUM

$$\frac{\left\{ P \right\} C \left\{ Q \right\} \qquad \left\{ P' \right\} C \left\{ Q' \right\}}{\left\{ P + P' \right\} C \left\{ Q + Q' \right\}}$$

HOARE-SCALE

$$\frac{\left\{ P \right\} C \left\{ Q \right\}}{\left\{ \alpha P \right\} C \left\{ \alpha Q \right\}}$$

The rules use the logical connectives $P + P'$ and αP representing the sum and scaling of quantum states respectively, i.e., RapunSL defines them so that $(\mathsf{x} \mapsto |\psi\rangle) + (\mathsf{x} \mapsto |\phi\rangle) \dashv\vdash \mathsf{x} \mapsto (|\psi\rangle + |\phi\rangle)$ and $\alpha (\mathsf{x} \mapsto |\psi\rangle) \dashv\vdash \mathsf{x} \mapsto \alpha |\psi\rangle$ are valid.[1][2]

Using such rules, it is possible to prove (2) by first seeing $|\psi\rangle$ as an explicit superposition of classical states: $|\psi\rangle = \sum_{\bar{b},t,\bar{c}} \alpha_{\bar{b},t,\bar{c}} |\bar{b} t \bar{c}\rangle$. Then by HOARE-SUM and HOARE-SCALE, we reduce the goal to a triple on a classical state; the state of ȳ can now simply be framed to reduce the problem to our original (1). Note how the framing of ȳ is only possible *after* having applied HOARE-SUM: in the original state, x̄, tmp and ȳ are arbitrarily entangled.

Perhaps surprisingly, no logic in the literature can perform the intuitive steps above. Prior logics, in fact, cannot soundly admit HOARE-SUM. The logics of Zhou et al. [2021] and Su et al. [2024] are incompatible with it because they adopt a *global-phase-insensitive* semantics for their assertions (e.g., by representing states as a density matrix). This choice seems justified as no measurement can distinguish between two states that differ only in the global phase. This is, however, fundamentally incompatible with HOARE-SUM. For example, in a global-phase-insensitive logic, the two states $\mathsf{x} \mapsto |1\rangle$ and $\mathsf{x} \mapsto -|1\rangle$ are indistinguishable (i.e., $\mathsf{x} \mapsto |1\rangle \dashv\vdash \mathsf{x} \mapsto -|1\rangle$). If sum were to preserve entailment here, we could derive $\mathsf{x} \mapsto 0 \vdash (\mathsf{x} \mapsto \frac{1}{2}|1\rangle + \mathsf{x} \mapsto -\frac{1}{2}|1\rangle) \vdash (\mathsf{x} \mapsto \frac{1}{2}|1\rangle + \mathsf{x} \mapsto \frac{1}{2}|1\rangle) \vdash \mathsf{x} \mapsto |1\rangle$—a meaningful state out of an impossible probability-zero state.

Retaining the global-phase information in RapunSL allows for the compositional treatment of some deductions, like the ones using linearity. The logic of Le et al. [2022] is the only other global-phase-sensitive quantum separation logic we are aware of. This logic is also fundamentally incompatible with HOARE-SUM because of its model of *measurements*. The main contribution of RapunSL is a new model which can support unrestricted applications of HOARE-SUM *and* measurements in the same logic.

2.2 The Main Challenge: Handling Measurements Soundly

The fundamental issue with measurement is that it is not a unitary operator. Specifically, it collapses a superposition state $\sum_i \alpha_i |s_i\rangle$ into the classical state $|s_i\rangle$ with probability $|\alpha_i|^2$. As a consequence, one would think the following innocent-looking triples should be valid:

$$\left\{ \mathsf{x} \mapsto |0\rangle \right\} \mathsf{M_Z}[\mathsf{x}] \left\{ \mathsf{x} \mapsto |0\rangle \right\} \qquad\qquad \left\{ \mathsf{x} \mapsto |1\rangle \right\} \mathsf{M_Z}[\mathsf{x}] \left\{ \mathsf{x} \mapsto |1\rangle \right\} \tag{3}$$

---

[1] We use ⊢ for entailment, and ⊣⊢ for bidirectional entailment, i.e., logical equivalence.

[2] Interestingly, we can derive HOARE-SCALE from the frame rule HOARE-FRAME, because αP can be represented as $(() \mapsto \alpha) * P$, where $() \mapsto \alpha$ is a zero-qubit state of a one-dimensional vector (i.e., a scalar) α.

The triples say that an already classical state cannot be "collapsed" further by measuring it. Remarkably, however, any logic that would admit (3) is fundamentally incompatible with HOARE-SUM. This is because its application to (3) would yield the invalid triple

$$\left\{ x \mapsto (\alpha \left| 0 \right\rangle + \beta \left| 1 \right\rangle) \right\} \; \mathsf{M_Z}[x] \; \left\{ x \mapsto (\alpha \left| 0 \right\rangle + \beta \left| 1 \right\rangle) \right\} \tag{4}$$

where no collapse of the superposition happens at all. In fact, the logic of Le et al. [2022] would admit (3), and thus cannot support the local reasoning afforded by HOARE-SUM.

At first sight, this might seem an insurmountable obstacle: the HOARE-SUM rule seems to imply every program is linear, but measurement is not. Is our objective even achievable?

The key insight behind the solution we provide with RapunSLis that the real culprit is the handling of *mixed states*, i.e., probabilistic mixtures of pure quantum states (the outcomes of measurement). In a logic like Le et al. [2022]'s, just as in standard separation logic, assertions talk about one possible outcome at a time. For example, the most accurate representation of the mixture of two outcomes P and Q is $P \vee Q$, an assertion that does not fully describe the mixed state, but only an over-approximation of the possible outcomes.

As a first step towards a solution, in RapunSL, we move to assertions that can predicate over the whole mixed state. To handle mixed states compositionally, we introduce a new connective $P \; _0\oplus_1^\iota \; Q$, called *tagged mixing*, which represents the mixed state resulting from running a measurement tagged with ι and with two outcomes 0 and 1.[3] Under this interpretation, the triples (3) become invalid: in RapunSL they would assert that the state resulting from a measurement is equivalent to one where no measurement was taken. A valid rule for measurement in RapunSL is:[4]

HOARE-M$_Z$
$$\left\{ x \mapsto (\alpha \left| 0 \right\rangle + \beta \left| 1 \right\rangle) \right\}^\iota \; \mathsf{M_Z^\iota}[x] \; \left\{ (x \mapsto \alpha \left| 0 \right\rangle) \; _0\oplus_1^\iota \; (x \mapsto \beta \left| 1 \right\rangle) \right\}$$

The rule states that from a state where x is in a superposition of states $\left| 0 \right\rangle$ and $\left| 1 \right\rangle$ with coefficients α and β, respectively, measuring x gives us a mixed state with two outcomes: one where the state of x collapsed to the classical state $\left| 0 \right\rangle$ and one where it collapsed to $\left| 1 \right\rangle$. More precisely, the single outcomes in the postcondition retain their coefficient (i.e., we do not normalize the states) so that we can read off the probabilities of each outcome (as the squared norm of the coefficient).

The crucial change induced by moving to assertions over mixed states is that now the $P + Q$ connective is not just given the meaning of "linear combination" but of "outcome-wise linear combination". This is essentially what makes HOARE-SUM sound in the presence of measurement: it no longer states that the program is linear, but linear on each outcome (i.e., once all the measurement's effects have been factored out). This is summarized in the following interchange rules, which are validated by RapunSL's model:

MIX-SUM
$$(P_0 \; _0\oplus_1^\iota \; P_1) + (Q_0 \; _0\oplus_1^\iota \; Q_1) \; \dashv\vdash \; (P_0 + Q_0) \; _0\oplus_1^\iota \; (P_1 + Q_1)$$

MIX-SCALE
$$\alpha \, (P_0 \; _0\oplus_1^\iota \; P_1) \; \dashv\vdash \; (\alpha \, P_0) \; _0\oplus_1^\iota \; (\alpha \, P_1)$$

Now we can resolve the apparent conflict between measurement and sum we started with: we can derive the specification HOARE-M$_Z$ of the behaviour of a measurement on a superposition state, from a specification of its behaviour on classical states. In RapunSL, the valid rules for measuring a classical state are:

$$\left\{ x \mapsto \left| 0 \right\rangle \right\}^\iota \; \mathsf{M_Z^\iota}[x] \; \left\{ (x \mapsto \left| 0 \right\rangle) \; _0\oplus_1^\iota \; (x \mapsto 0) \right\} \tag{5}$$

$$\left\{ x \mapsto \left| 1 \right\rangle \right\}^\iota \; \mathsf{M_Z^\iota}[x] \; \left\{ (x \mapsto 0) \; _0\oplus_1^\iota \; (x \mapsto \left| 1 \right\rangle) \right\} \tag{6}$$

---

[3] Deng et al. [2024]'s logic has a similar connective $\oplus$, but their model is fundamentally different from ours and incompatible with basis-locality. This will be discussed in more detail later in § 7.

[4] The superscript ι of the precondition means that the ownership of the variable ι is required. See § 4.4 for the details.

Crucially, for these to be valid in our model, they have to include in the postcondition an explicit tagged mix connective, with only one meaningful branch and one probability-zero outcome (the measurement that cannot materialize) represented as the zero-norm state $x \mapsto 0$. In fact, we can now *derive* HOARE-M_Z from (5) and (6):

$$\frac{\dfrac{\left\{x \mapsto |0\rangle\right\}^t M_Z^t[x] \left\{x \mapsto |0\rangle \ _0\oplus_1^t x \mapsto 0\right\}}{\left\{x \mapsto \alpha |0\rangle\right\}^t M_Z^t[x] \left\{x \mapsto \alpha |0\rangle \ _0\oplus_1^t x \mapsto 0\right\}} (10) \quad \dfrac{\left\{x \mapsto |1\rangle\right\}^t M_Z^t[x] \left\{x \mapsto 0 \ _0\oplus_1^t x \mapsto \alpha |1\rangle\right\}}{\left\{x \mapsto \beta |1\rangle\right\}^t M_Z^t[x] \left\{x \mapsto 0 \ _0\oplus_1^t x \mapsto \beta |1\rangle\right\}} (9)}{\dfrac{\left\{x \mapsto (\alpha |0\rangle + \beta |1\rangle)\right\}^t M_Z^t[x] \left\{(x \mapsto \alpha |0\rangle \ _0\oplus_1^t x \mapsto 0) + (x \mapsto 0 \ _0\oplus_1^t x \mapsto \beta |1\rangle)\right\}}{\left\{x \mapsto (\alpha |0\rangle + \beta |1\rangle)\right\}^t M_Z^t[x] \left\{x \mapsto \alpha |0\rangle \ _0\oplus_1^t x \mapsto \beta |1\rangle\right\}} (7)} (8)$$

We start by applying HOARE-SCALE and MIX-SCALE to (5) and (6) in steps (9) and (10), to introduce the coefficients for each (classical) state. Then, in step (8), we apply HOARE-SUM to combine the states in a superposition; this gives a postcondition which is the sum of two mixed states. In the final step (7), we use MIX-SUM to obtain a mixed state of sums, as required; note how summing a state with the impossible outcome $x \mapsto 0$ leaves the state unchanged.

2.3 The Three Layers of Locality

As we described, RapunSL supports three locality principles at the same time, displayed in Fig. 1. That is, in a mixed state, we can reason on a per-outcome basis (HOARE-MIX); in a superposition state, we can reason on the classical basis states (HOARE-SUM); and in a state where some qubits are disentangled, we can focus on the relevant ones and ignore the others (HOARE-FRAME). Given that a proof in RapunSL will inevitably involve a combination of all three layers, it is crucial to study the interaction between the three constructs of mixing, sum and separation. Our model is carefully constructed to enjoy a number of distributivity/interchange rules that allow for flexible combinations of the connectives, which we review next.

A first illustration of the flexibility of RapunSL was already presented in MIX-SUM, which shows an interchange law between mixing and sum. This allows for handling superposition and mixing in any order without losing information. A similar interchange law holds for mixings arising from two consecutive measurements on different qubits.

MIX-MIX
$$(P_{00} \ _0\oplus_1^t P_{01}) \ _0\oplus_1^\kappa (P_{10} \ _0\oplus_1^t P_{11}) \ \dashv\vdash \ (P_{00} \ _0\oplus_1^\kappa P_{10}) \ _0\oplus_1^t (P_{01} \ _0\oplus_1^\kappa P_{11})$$

The MIX-MIX rule says that RapunSL's model is insensitive to the order in which measurements are taken, showing that the $_0\oplus_1^t$ connective is not simply a representation of the program's measurements in the assertions, but a genuine compositional abstraction over them.

Finally, separation is also well-behaved concerning superposition and mixing:

SUM-FRAME
$$(P + Q) * R \vdash (P * R) + (Q * R)$$

MIX-FRAME
$$(P \ _0\oplus_1^t Q) * R \vdash (P * R) \ _0\oplus_1^t (Q * R)$$

Thanks to MIX-FRAME, adding a disentangled frame to a mixed state does not disallow per-outcome reasoning. As we explain next, the reverse direction of MIX-FRAME is crucial for the scalability of reasoning in RapunSL.

2.4 Abstraction

As we argued, we can only hope to have a sound logic if every measurement introduces a mixing connective in the postcondition. Without care, this can induce an exponential growth in the outcomes to be considered. The key tool provided by RapunSL to control this complexity is the
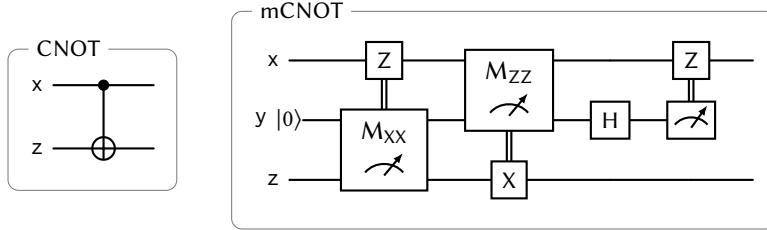
Fig. 2. A CNOT gate (left) and its encoding mCNOT using measurements (right).

reverse direction of the MIX-FRAME rule:

MIX-UNFRAME
$$\frac{R : \text{precise}}{(P * R)\ _0{\oplus}_1^t\ (Q * R)\ \vdash\ (P\ _0{\oplus}_1^t\ Q) * R}$$

The MIX-UNFRAME rule says that it is possible (under a technical condition on R) to factor out the portions of state that are common to multiple outcomes into a single frame. This allows any reasoning that only depends on R to be done once, and for the remaining information to be framed around the reasoning.

To show more concretely the positive effect of this rule on the abstraction capabilities of RapunSL, let us consider a specific example which we treat in full detail in §5.3. In certain quantum architectures designed to support fault tolerance [Horsman et al. 2012], the implementation of basic 2-qubit gates, such as CNOT (a.k.a. CX), can be improved by implementing their functionality through a combination of 1-qubit unitary gates and 2-qubit measurements [Fowler and Gidney 2019]. In Fig. 2, we show the schema of such a circuit called mCNOT, encoding a CNOT gate. The details are explained in §5.3, but for our purposes, what is important is that the mCNOT encodes a CNOT between x and z, using an auxiliary (ancilla) qubit y initialized with state $|0\rangle$, and that the circuit contains several measurements.

Although the mCNOT circuit is designed to be morally equivalent to the CNOT, formally, the two have important differences: mCNOT requires an extra qubit, and the measurements produce a mixed state. Without care, using mCNOT in compositional reasoning instead of the CNOT may cause incorrect results. However, if the circuit using mCNOT is not making use of these differences, reasoning about the overall correctness should proceed essentially as if we used CNOT instead of mCNOT. In RapunSL, we can replicate this rough argument fully formally.

The idea is that mCNOT can be proven to satisfy a specification of the form:[5]

$$\forall\,|\psi\rangle\,.\,\big\{\,(x, z) \mapsto |\psi\rangle * y \mapsto |0\rangle\,\big\}\ \text{mCNOT}[x, y, z]\ \big\{\,(x, z) \mapsto CX\,|\psi\rangle * P_{\oplus}\,\big\}$$

where $P_{\oplus}$ contains mixing connectives $_0{\oplus}_1$ introduced by the measurements and (per-outcome) information about the state of y and the global phase. The rest of the postcondition asserts that the effect on the x and z qubits is the same as the effect of a CNOT gate. The ability to group the measurement "side effects" into $P_{\oplus}$ is given by MIX-UNFRAME. This grouping has two nice effects. First, it allows any user of the specification to lift reasoning that holds for CNOT to reasoning that holds for mCNOT by framing $P_{\oplus}$. Second, when lifting reasoning in this way, framing $P_{\oplus}$ safeguards against possible unsoundness, i.e., if a step applies to CNOT but not to mCNOT, the frame $P_{\oplus}$ prevents lifting the result to mCNOT.

---

[5] To be precise, the precondition has the superscript of the classical variables used for storing the results of measurements.

We also note that, although we adopt a global-phase-sensitive logic, we do not preclude users from employing density matrices when a more compact representation is desired. Density matrices can be encoded within our logic, thus users can switch to use them to have smaller state descriptions; however, this comes at the cost of reduced modularity. We will see the details in § 6.

2.5 Summary

In summary, we constructed a new logic, RapunSL, that is able to achieve modular reasoning across three ways of combining quantum programs: by adding a disentangled state via *separation*, by superposition via *sum*, and by adding measurements via *mixing*. In the remainder of the paper, we provide the model and rules of RapunSL, and evaluate it through a series of challenging case studies. The omitted details and full proofs of soundness can be found in § A.

3 Preliminaries and Program Language

3.1 A Primer on Quantum Computing

The basic unit of data in quantum computing is the *qubit*, which can take not only the *classical states* $|0\rangle$ and $|1\rangle$, but also a *superposition* of them, that is, a linear combination of the two states $|\psi\rangle = \alpha |0\rangle + \beta |1\rangle \in \mathbb{C}^2$. Here, the squared norm $\||\psi\rangle\|^2 = |\alpha|^2 + |\beta|^2$ represents the probability of that state. More generally, the *pure state* of a system of n qubits is a vector in a 2^n-dimensional Hilbert space. For example, the pure states of two qubits are described by vectors of the form $\alpha |00\rangle + \beta |01\rangle + \gamma |10\rangle + \delta |11\rangle$. The states $|b\rangle$ with $b \in \mathbb{B}^n$ are the *basis* states. If two spaces $\mathcal{H}_0$ and $\mathcal{H}_1$ represent the states of quantum data x and y respectively, a state of the composite system (x, y) is called *separated* or *disentangled* if it can be represented as $|\psi_0\rangle \otimes |\psi_1\rangle$, i.e., there is no correlation between the two data. If not, then the state is called *entangled*. An example of an entangled state is the Bell state $|\text{Bell}\rangle = (|00\rangle + |11\rangle)/\sqrt{2}$.

Quantum states that differ only in a *global phase* $\alpha \in \mathbb{C}$, i.e., $|\psi\rangle$ and $\alpha |\psi\rangle$, are physically indistinguishable: no measurement can detect the difference. Only *relative* phase—the difference in phase between coefficients of the basis states—can be measured.

Quantum computing is performed by *quantum gates*. A quantum gate on n qubits is a *unitary operator* (or matrix) on the vector space $(\mathbb{C}^2)^{\otimes n}$. In particular, quantum gates are linear, and thus their effect is determined by their effect on basis states: $U(\sum_i \alpha_i |b_i\rangle) = \sum_i \alpha_i U |b_i\rangle$. For example, the X gate (on a single qubit) flips $|0\rangle$ to $|1\rangle$ and vice versa; the Z gate flips the sign of $|1\rangle$; the CX = CNOT gate (on two qubits) maps $|ab\rangle$ to $|a, a \veebar b\rangle$ ($\veebar$ stands for xor). Any classical bijection $f : \{0, 1\}^n \rightarrow \{0, 1\}^n$ can be lifted to a unitary operator U_f on $(\mathbb{C}^2)^{\otimes n}$ that maps $|x\rangle$ to $|f(x)\rangle$.

Performing a measurement on a qubit in state $\alpha |0\rangle + \beta |1\rangle$ makes the state collapse to $|0\rangle$ with probability $|\alpha|^2$ and to $|1\rangle$ with probability $|\beta|^2$. More generally, a measurement M is defined as a set of linear operators $\{M^{(i)}\}_{i=0}^k$ satisfying $\sum_{i=0}^k M^{(i)\dagger} M^{(i)} = \text{id}$. For example, the usual single-qubit Z-basis measurement M_Z is defined as $\{M_Z^{(i)}\}_{i=0}^1$ such that $M_Z^{(0)} \triangleq |0\rangle\langle 0|$ and $M_Z^{(1)} \triangleq |1\rangle\langle 1|$. The process of a measurement M is described as follows:

$$|\psi\rangle \rightarrow M^{(i)} |\psi\rangle / \sqrt{p_i} \qquad p_i = \|M^{(i)} |\psi\rangle\|^2.$$

That is, the quantum state is projected with probability p_i onto the subspace corresponding to each $M^{(i)}$. The probabilistic mixture of pure states resulting from a measurement is called a *mixed state*.

3.2 Program Language

We define a simple, imperative language for quantum computing with minimal features. We equip it with a straightforward small-step operational semantics and extract a denotational collecting semantics which we use as a foundation for the model of our logic.

$$\text{Qubit} \quad x, y \in Qubit \qquad \text{Variable} \quad a, b, c, \iota, \kappa, \lambda \in Var$$

$$\text{Pure expression} \quad e ::= n \mid a \mid op(\bar{e}) \qquad \text{Value} \quad Val \ni v, w ::= n$$

$$\text{Command} \quad Cmd \ni C ::= \text{skip} \mid C; C' \mid \text{if } e \text{ then } C \text{ else } C' \mid \text{while } e \text{ do } C$$
$$\mid a \leftarrow e \mid U[\bar{x}] \mid a \leftarrow M[\bar{x}]$$

$$\text{if } e \text{ then } C \triangleq \text{ if } e \text{ then } C \text{ else skip} \qquad M^a[\bar{x}] \triangleq a \leftarrow M[\bar{x}]$$

Fig. 3. Syntax of the program language.

$$\text{Quantum state} \quad |\psi\rangle \in Qstate \triangleq \sum_{X \subseteq Qubit} \bigotimes_{x \in X} \mathbb{C}^2 \qquad \text{Store} \quad S \in Store \triangleq Var \rightharpoonup Val$$

Fig. 4. Domains for states in the language.

$$\text{Configuration} \quad c ::= (\bar{C}, |\psi\rangle, S)$$

$$(\text{skip}, \bar{C}, |\psi\rangle, S) \rightarrow (\bar{C}, |\psi\rangle, S) \qquad (C_0; C_1, \bar{C}', |\psi\rangle, S) \rightarrow (C_0, C_1, \bar{C}', |\psi\rangle, S)$$

$$(\text{if } e \text{ then } C_1 \text{ else } C_0, \bar{C}', |\psi\rangle, S) \rightarrow (C_{[\![e]\!]_S}, \bar{C}', |\psi\rangle, S)$$

$$(\text{while } e \text{ do } C, \bar{C}', |\psi\rangle, S) \rightarrow (\text{if } e \text{ then } (C; \text{while } e \text{ do } C), \bar{C}', |\psi\rangle, S)$$

$$(a \leftarrow e, \bar{C}, |\psi\rangle, S\{a \leftarrow v\}) \rightarrow (\bar{C}, |\psi\rangle, S\{a \leftarrow [\![e]\!]_S\}) \qquad (U[\bar{x}], \bar{C}, |\psi\rangle, S) \rightarrow (\bar{C}, U_{\bar{x}} |\psi\rangle, S)$$

$$(a \leftarrow M[\bar{x}], \bar{C}, |\psi\rangle, S\{a \leftarrow v\}) \xrightarrow{i} (\bar{C}, M_{\bar{x}}^{(i)} |\psi\rangle, S\{a \leftarrow i\})$$

Fig. 5. Operational semantics of the program language.

Syntax. The syntax of our program language is summarized in Fig. 3. We have pure expressions e and commands C. For simplicity, the command for measurement has the form $a \leftarrow M[\bar{x}]$, requiring that the result of the measurement be stored in some (classical) variable a. The notation $M^a[\bar{x}]$ used in § 2 is just shorthand for this command. We represent Boolean values using 1 for true and 0 for false. The guard of if statements and while loops is an expression. To branch on a measurement result, one can first store the outcome of the measurement into a variable and then use it as a guard.

States. The domains for states in the language are summarized in Fig. 4. The domain *Qstate* assigns a state vector $|\psi\rangle$ to some set of qubits X. We denote its elements as $X \mapsto |\psi\rangle$, or just $|\psi\rangle$ when there is no confusion. A store $S \in Store$ assigns to some set of variables a value, that is, classical data of any type.

Semantics of pure expressions. The semantics $[\![e]\!]_S \in Val$ of a pure expression e under the store S is naturally defined as follows:

$$[\![n]\!]_S \triangleq n \qquad [\![a]\!]_S \triangleq S[a] \qquad [\![op(\bar{e})]\!]_S \triangleq op(\overline{[\![e]\!]_S})$$

Note that it is undefined when e contains a variable a that is not in the domain of S.

Operational semantics. Next, we present the operational semantics of our language. It is summarized in Fig. 5. The configuration c has the syntax shown at the top of Fig. 5. Here, $|\psi\rangle \in Qstate$ and $S \in Store$. The small-step reduction relation $c \rightarrow c'$, $c \xrightarrow{i} c'$ is defined by the rules in Fig. 5. Importantly, we put a label $i \in \mathbb{N}$ on the reduction to indicate the result of a measurement $M[\bar{x}]$.

Behaviour tree $\quad BTree \ni t \ ::= \ \mathsf{Branch}(\bar{t}) \ | \ \mathsf{Leaf}(|\psi\rangle, S) \ | \ \mathsf{Nil} \quad \text{(coinductively)}$

$$[\![\epsilon]\!](|\psi\rangle, S) \ \triangleq \ \mathsf{Leaf}(|\psi\rangle, S) \qquad [\![\mathsf{skip}, \bar{C}]\!](|\psi\rangle, S) \ \triangleq \ [\![\bar{C}]\!](|\psi\rangle, S)$$

$$[\![C_0; C_1, \bar{C}']\!](|\psi\rangle, S) \ \triangleq \ [\![C_0, C_1, \bar{C}']\!](|\psi\rangle, S)$$

$$[\![\mathsf{if}\ e\ \mathsf{then}\ C_1\ \mathsf{else}\ C_0, \bar{C}']\!](|\psi\rangle, S) \ \triangleq \ [\![C_{[\![e]\!]_S}, \bar{C}']\!](|\psi\rangle, S)$$

$$[\![\mathsf{while}\ e\ \mathsf{do}\ C, \bar{C}']\!](|\psi\rangle, S) \ \triangleq \ [\![\mathsf{if}\ e\ \mathsf{then}\ (C; \mathsf{while}\ e\ \mathsf{do}\ C), \bar{C}']\!](|\psi\rangle, S)$$

$$[\![a \leftarrow e, \bar{C}]\!](|\psi\rangle, S\{a \leftarrow v\}) \ \triangleq \ [\![\bar{C}]\!](|\psi\rangle, S\{a \leftarrow [\![e]\!]_S\}) \qquad [\![U[\bar{\mathsf{x}}], \bar{C}]\!](|\psi\rangle, S) \ \triangleq \ [\![\bar{C}]\!](U_{\bar{\mathsf{x}}}|\psi\rangle, S)$$

$$[\![a \leftarrow M[\bar{\mathsf{x}}], \bar{C}]\!](|\psi\rangle, S\{a \leftarrow v\}) \ \triangleq \ \mathsf{Branch}\big(\overline{[\![\bar{C}]\!](M_{\bar{\mathsf{x}}}^{(i)}|\psi\rangle, S\{a \leftarrow i\})}^i\big)$$

Fig. 6. Denotational semantics of the program language.

Denotational semantics. Now we present the denotational semantics of our language. Our denotational semantics collects all the possible branches as a whole. It is summarized in Fig. 6.

First, we define the domain of *behaviour trees* $t \in BTree$ as possibly infinite trees *coinductively* by the syntax at the top of Fig. 6. The branch $\mathsf{Branch}(\bar{t})$ represents branching by a measurement. The leaf $\mathsf{Leaf}(|\psi\rangle, S)$ represents a branching-free terminating execution with the final state $(|\psi\rangle, S)$. The nil tree Nil represents a branching-free non-terminating execution.

We also define the partial order $t \leq t'$ over behaviour trees coinductively by the rules $\mathsf{Nil} \leq t$, $\mathsf{Leaf}(|\psi\rangle, S) \leq \mathsf{Leaf}(|\psi\rangle, S)$, and "if $t_i \leq t'_i$ for each i, then $\mathsf{Branch}(\bar{t}) \leq \mathsf{Branch}(\bar{t}')$". In other words, $t \leq t'$ means that t' can be obtained from t by replacing each occurrence of Nil with some tree. Also, we define the child access $t.i$ as $\mathsf{Branch}(\bar{t}').i \triangleq t'_i$ and undefined otherwise.

The denotational semantics $[\![\bar{C}]\!](|\psi\rangle, S) \in BTree$ is defined *inductively* by the equations in Fig. 6. The semantics is defined for a general sequence of commands $\bar{C}$ (ϵ is the empty sequence). Technically, the *least* fixed point for the equations in Fig. 6 is constructed as the limit $\lim_{n \to \infty}[\![-]\!]^n(-, -)$ of n-th approximations $[\![-]\!]^n(-, -) : SemArg \rightharpoonup BTree$, leaving parts that have not terminated in n steps as Nil.[6] Notably, we can enjoy equational reasoning using the denotational semantics.

The following theorem formalizes the relation between operational and denotational semantics.

Theorem 1 (Equivalence of the operational and denotational semantics). *Take any configuration* $c = (\bar{C}, |\psi\rangle, S)$. *The configuration never gets stuck in any branches (i.e., any c' reachable from c is reducible) if and only if $t \triangleq [\![\bar{C}]\!](|\psi\rangle, S)$ is defined. Moreover, assuming that t is defined, $c \xrightarrow{\bar{i}}{}^* (|\phi\rangle, S')$ holds if and only if $t.\bar{i} = \mathsf{Leaf}(|\phi\rangle, S')$.*

PROOF. For the first statement, the backward implication is straightforward. The forward implication follows from the definedness of all the approximations $[\![-]\!]^n$, proved by induction over n. For the second statement, the forward implication is obtained by unfolding $[\![-]\!]$. The backward implication can be proved by the fact that there exists some n such that the n-th approximation has that leaf, i.e., $[\![\bar{C}]\!]^n(|\psi\rangle, S).\bar{i} = \mathsf{Leaf}(|\phi\rangle, S')$. □

---

[6] First, let $\mathsf{Sem}(f)(-, -, -) : SemArg \rightharpoonup BTree$ for $f : SemArg \rightharpoonup BTree$ (where $SemArg \triangleq Cmd^* \times Qstate \times Store$) be the map obtained by replacing all the self-references to $[\![-]\!]$ with f in the definition of $[\![\bar{C}]\!](|\psi\rangle, S)$ in Fig. 6. Then the n-th approximation is inductively defined by $[\![-]\!]^0(-, -) \triangleq \lambda_{-,-,-}.\mathsf{Nil}$ and $[\![-]\!]^{n+1}(-, -) \triangleq \mathsf{Sem}([\![-]\!]^n(-, -))$. Finally, we set $[\![-]\!](-, -) \triangleq \lim_{n \to \infty}[\![-]\!]^n(-, -)$. Here, we take the limit of an ω-chain (i.e., increasing sequence) $[\![-]\!]^0(-, -) \leq [\![-]\!]^1(-, -) \leq [\![-]\!]^2(-, -) \leq \cdots$, which is defined because $BTree$ is ω-complete. Note that partial maps $SemArg \rightharpoonup BTree$ are given the pointwise order, where undefined parts are regarded as the maximum element.

$$a, b \in Res \triangleq Qstate \times Store \qquad P, Q \in SLProp \triangleq \mathcal{P}(\mathcal{M}(Res))$$

$$P \vdash Q \triangleq P \subseteq Q \qquad P \vdash P \qquad \frac{P \vdash Q \qquad Q \vdash R}{P \vdash R}$$

Fig. 7. Model of RapunSL's SL propositions.

4 The RapunSL Logic

In this section, we formally introduce our logic, RapunSL. We treat assertions and judgments semantically, with entailment rules being just valid lemmas about the semantic model.

4.1 Resources, Propositions and Entailment

We start by fixing notation for multisets, which we use to build our model of mixed state resources.

Definition 2 (Multisets). For a set A, the set of multisets $\mathcal{M}(A)$ is defined as the quotient of the set[7] $\bigcup_{I \in \mathbf{Set}} I \to A$ over the following equivalence relation: $f: I \to A$ and $g: J \to A$ are equivalent if there is a bijection $p: I \to J$ such that $f = g \circ p$. In other words, multisets are maps with the key forgotten. We write $\lfloor f \rfloor \in \mathcal{M}(A)$ for the multiset of the equivalence class of $f: I \to A$.

We use the extensional notation $\{\!\{ a_0, \ldots, a_{n-1} \}\!\}$, regarding the sequence as a map from $\{0, \ldots, n-1\}$. For example, the multiset $\{\!\{ 0, 1, 1, 2 \}\!\}$ is the same as $\{\!\{ 2, 1, 0, 1 \}\!\}$ but different from $\{\!\{ 0, 1, 2 \}\!\}$. We also use the comprehension notation for multisets. For example, we can write $\{\!\{ f\,i \mid i \in I \}\!\}$ for $\lfloor f: I \to A \rfloor$. Also, we can take an element from multisets in the comprehension notation, using multiset membership $\in_{\mathrm{m}}$, taking the multiplicity into account. For example, $\{\!\{ n \mid n \in_{\mathrm{m}} \{\!\{ 0, 0, 1, 5 \}\!\}, n < 2 \}\!\}$ means the multiset $\{\!\{ 0, 0, 1 \}\!\}$, not $\{\!\{ 0, 1 \}\!\}$.

Also, multiset inclusion $m \subseteq_{\mathrm{m}} m'$ over $m, m' \in \mathcal{M}(A)$ is defined as follows: $\lfloor f: I \to A \rfloor \subseteq_{\mathrm{m}} \lfloor g: J \to A \rfloor$ holds if and only if there exists an injection $p: I \to J$ such that $f = g \circ p$. For example, $\{\!\{ 0, 1, 2 \}\!\} \subseteq_{\mathrm{m}} \{\!\{ 0, 1, 2, 2 \}\!\}$ holds, but $\{\!\{ 0, 1, 2, 2 \}\!\} \subseteq_{\mathrm{m}} \{\!\{ 0, 1, 2 \}\!\}$ does not. $\qquad \square$

Now we can model RapunSL's SL propositions as presented in Fig. 7. A resource $a \in Res$ consists of a quantum state $\mathsf{X} \mapsto |\psi\rangle \in Qstate$ and a store $S \in Store$. A mixed state is represented as a *multiset* of resources, collecting all the outcomes of branching due to measurement. An SL proposition $P \in SLProp$ is modelled as the set of multisets of resources that satisfy it. The entailment relation $P \vdash Q$ is defined simply by set inclusion, which is clearly reflexive and transitive. We can define the standard connectives, such as the universal and existential quantifiers $\forall x \in A. P_x, \exists x \in A. P_x$, as usual, with standard proof rules; please refer to § A.1 for the details.

4.2 Bare Mixing

Tagged mixing ${}_0\oplus_1^t$ we introduced in § 2 is derived, in RapunSL, from a more fundamental connective $\oplus$ called *bare mixing*, defined in Fig. 8. A proposition $P \oplus Q$ represents all collections of outcomes in the mixed states described by P and by Q. It is therefore naturally expressed via (pointwise) multiset sum.

Definition 3 (Sum of multisets). The sum of multisets $m \uplus m'$ is defined as follows: $\lfloor f: I \to A \rfloor \uplus \lfloor g: J \to A \rfloor \triangleq \lfloor h: I + J \to A \rfloor$ where $h\,(\mathrm{inl}\,i) \triangleq f\,i$ and $h\,(\mathrm{inr}\,j) \triangleq g\,j$. For example, $\{\!\{ 0, 1 \}\!\} \uplus \{\!\{ 1, 2 \}\!\} = \{\!\{ 0, 1, 1, 2 \}\!\}$. Moreover, the indexed sum $\uplus_{x \in I} m_x$ over an indexed family of multisets $(m_x)_{x \in I}$ is defined as $\uplus_{x \in I} \lfloor f_x: J_x \to A \rfloor \triangleq \lfloor g: \bigsqcup_{x \in I} J_x \to A \rfloor$ where $g\,(x, j) \triangleq f_x\,j$. We can also use the comprehension notation $\{\!\{ a \mid x \in I, a \in_{\mathrm{m}} m_x \}\!\}$ for $\uplus_{x \in I} m_x$. $\qquad \square$

---

[7] Here, we fix some universe of sets $\mathbf{Set}$.

$$\text{nb} \triangleq \{\!\!\{ \{\!\!\{\}\!\!\} \}\!\!\} \qquad P \oplus Q \triangleq \big\{ m \uplus m' \mid m \in P, \, m' \in Q \big\}$$

$$\bigoplus_{x \in I} P_x \triangleq \big\{ \uplus_{x \in I} m_x \mid \forall x \in I. \, m_x \in P_x \big\}$$

BMIX-MONO
$$\frac{P \vdash P' \qquad Q \vdash Q'}{P \oplus Q \vdash P' \oplus Q'}$$

NB-BMIX
$$\text{nb} \oplus P \dashv\vdash P$$

BMIX-COMM
$$P \oplus Q \dashv\vdash Q \oplus P$$

BMIX-ASSOC
$$(P \oplus Q) \oplus R \dashv\vdash P \oplus (Q \oplus R)$$

BIGBMIX-MONO
$$\frac{\forall x \in I. \, (P_x \vdash Q_x)}{\bigoplus_{x \in I} P_x \vdash \bigoplus_{x \in I} Q_x}$$

BIGBMIX-COMM
$$\frac{f : I \to J \text{ is a bijection}}{\bigoplus_{x \in I} P_{fx} \dashv\vdash \bigoplus_{y \in J} P_y}$$

BIGBMIX-ASSOC
$$\bigoplus_{x \in I} \bigoplus_{y \in J_x} P_{x,y} \dashv\vdash \bigoplus_{(x,y) \in \bigsqcup_{x \in I} J_x} P_{x,y}$$

NB-BIGBMIX
$$\text{nb} \dashv\vdash \bigoplus_{\_ \in \varnothing}$$

BMIX-BIGBMIX
$$P_0 \oplus P_1 \dashv\vdash \bigoplus_{i \in \{0,1\}} P_i$$

Fig. 8. Bare mixing and its proof rules in RapunSL.

$$\varepsilon_{Qstate} \triangleq \varnothing \mapsto 1 \qquad (\mathsf{X} \mapsto |\psi\rangle) \cdot_{Qstate} (\mathsf{Y} \mapsto |\phi\rangle) \triangleq \mathsf{X} \cup \mathsf{Y} \mapsto |\psi\rangle \otimes |\phi\rangle \quad \text{if } \mathsf{X} \cap \mathsf{Y} = \varnothing$$

$$\varepsilon_{Store} \triangleq \varnothing \qquad S \cdot_{Store} S' \triangleq S \cup S' \quad \text{if } \text{dom } S \cap \text{dom } S' = \varnothing$$

$$\varepsilon_{Res} \triangleq (\varepsilon_{Qstate}, \varepsilon_{Store}) \qquad (|\psi\rangle, S) \cdot_{Res} (|\phi\rangle, S') \triangleq (|\psi\rangle \cdot_{Qstate} |\phi\rangle, \, S \cdot_{Store} S)$$

Fig. 9. PCM structure over *Qstate*, *Store*, and *Res*.

Figure 8 present rules that reflect basic properties of the multiset sum semantics. Notably, binary mixing ⊕ has the unit nb (NB-BMIX) and is commutative (BMIX-COMM) and associative (BMIX-ASSOC). Here, we introduce the no-behaviour assertion nb, modelled as the empty multiset. As we allow finite and infinite multisets, we can also introduce the indexed mixing $\bigoplus_{x \in I} P_x$ over an arbitrary (possibly infinite) set I. Indexed mixing is commutative and associative (BIGBMIX-COMM, BIGBMIX-ASSOC). Note that nb corresponds to mixing indexed over the empty set $\varnothing$ (NB-BIGBMIX; here the body of $\bigoplus$ is an empty family of propositions) and binary ⊕ corresponds to mixing indexed over $\{0, 1\}$ (BMIX-BIGBMIX).

4.3 Separating Conjunction

Our next question is how to define separating conjunction ∗. Following the usual approach, we use PCMs (partial commutative monoids).

Definition 4 (PCM). A partial commutative monoid (PCM) $(A, \varepsilon_A \in A, \cdot_A : A \times A \rightharpoonup A)$ is a set A equipped with the unit ε_A and the partial product $\cdot_A$ (we may omit the subscript) such that $\varepsilon_A \cdot_A a = a$, $a \cdot_A b = b \cdot_A a$, and $(a \cdot_A b) \cdot_A c = a \cdot_A (b \cdot_A c)$. Here, we use the Kleene equality, where the left-hand side is defined if and only if the right-hand side is defined. □

The PCM structure over *Qstate*, *Store* and *Res*, presented in Fig. 9, is standard. For stores *Store*, the product is defined only if the domains are disjoint. For quantum states *Qstate*, the product is defined only if the qubit sets are disjoint and uses the tensor product $|\psi\rangle \otimes |\phi\rangle$ for the qubit vector. For resources *Res*, we simply define operations component-wise.

Now, our SL propositions *SLProp* are a set of multisets $\mathcal{M}(Res)$. We want to give a PCM structure to them to define separating conjunction ∗, the core of separation logic. The top of Fig. 10 shows how we can do that. The unit is just the singleton multiset consisting of the unit resource. The product distributes over each argument multiset. Note that membership $a \in_m m$, $b \in_m m'$ in the

$$\varepsilon_{\mathcal{M}(Res)} \triangleq \{\!\{\varepsilon_{Res}\}\!\} \qquad m \cdot_{\mathcal{M}(Res)} m' \triangleq \{\!\{ a \cdot_{Res} b \mid a \in_{\mathrm{m}} m, b \in_{\mathrm{m}} m' \}\!\}$$

$$\mathrm{emp} \triangleq \{ \varepsilon_{\mathcal{M}(Res)} \} \qquad P * Q \triangleq \{ m \cdot_{\mathcal{M}(Res)} m' \mid m \in P, m' \in Q \}$$

SEP-MONO
$$\frac{P \vdash P' \qquad Q \vdash Q'}{P * Q \vdash P' * Q'}$$

EMP-SEP
$$\mathrm{emp} * P \dashv\vdash P$$

SEP-COMM
$$P * Q \dashv\vdash Q * P$$

SEP-ASSOC
$$(P * Q) * R \dashv\vdash P * (Q * R)$$

BIGBMIX-FRAME
$$\left(\bigoplus_{x \in I} P_x\right) * Q \vdash \bigoplus_{x \in I} (P_x * Q)$$

BIGBMIX-UNFRAME
$$\frac{Q : \mathrm{precise}}{\bigoplus_{x \in I} (P_x * Q) \vdash \left(\bigoplus_{x \in I} P_x\right) * Q}$$

Fig. 10. PCM structure over $\mathcal{M}(Res)$, separating conjunction, and its proof rules in RapunSL.

$$P : \mathrm{precise} \triangleq \forall m, m' \in P. \ m = m'$$

$$\frac{P : \mathrm{precise} \quad Q \vdash P}{Q : \mathrm{precise}} \qquad \mathrm{nb, emp : precise} \qquad \frac{\forall x. \ (P_x : \mathrm{precise})}{\bigoplus_{x \in I} P_x : \mathrm{precise}} \qquad \frac{P, Q : \mathrm{precise}}{P * Q : \mathrm{precise}}$$

Fig. 11. Precision of SL propositions.

comprehension notation takes the multiplicity into account. More explicitly, $\cdot_{\mathcal{M}(Res)}$ is defined as follows: $\lfloor f : I \to Res \rfloor \cdot_{\mathcal{M}(Res)} \lfloor g : J \to Res \rfloor \triangleq \lfloor \lambda(i, j) \in I \times J. \ f \ i \cdot_{Res} g \ j \rfloor$.

Directly using this PCM structure, we can define empty ownership emp and separating conjunction $P * Q$. We enjoy the rules SEP-MONO, EMP-SEP, SEP-COMM, SEP-ASSOC directly from the definition and the PCM structure. Also, bare mixing $\oplus$ distributes over separating conjunction $*$ by rules BIGBMIX-FRAME and BIGBMIX-UNFRAME. This comes from the fact that multiset sum $\uplus$ distributes over $\cdot_{\mathcal{M}(Res)}$, i.e., $(m_0 \uplus m_1) \cdot_{\mathcal{M}(Res)} m' = (m_0 \cdot_{\mathcal{M}(Res)} m') \uplus (m_1 \cdot_{\mathcal{M}(Res)} m')$ (with the Kleene equality). Note that framing out an assertion Q BIGBMIX-UNFRAME requires that the assertion Q is precise, meaning that it represents up to one multiset of resources. Very roughly speaking, assertions 'without disjunction' are precise. For example, in the rule BIGBMIX-UNFRAME, if we could set $Q = (a \mapsto 0 \lor a \mapsto 1)$ (which is *not* precise) and $P_0 = P_1 = \mathrm{emp}$ with $I = \{0, 1\}$, then this is unsound, because the state $a \mapsto 0 \oplus a \mapsto 0$ is included in the left-hand side but not in the right-hand side. Figure 11 shows the definition and basic rules of the precision judgment $P :$ precise, which are straightforward.

4.4 Reasoning about Quantum Programs

Now we present the features of RapunSL for reasoning about quantum programs. The sum $P + Q$ of SL assertions, the heart of RapunSL, is discussed later in § 4.5.

Tokens for quantum and classical states. Now we introduce the tokens for quantum and classical states as shown in Fig. 12. The quantum points-to token $\bar{\mathrm{x}} \mapsto |\psi\rangle$ is defined simply as the quantum state with the domain $\{\bar{\mathrm{x}}\}$. Here, distinct($\bar{\mathrm{x}}$) means that the qubit names $\bar{\mathrm{x}}$ are mutually distinct, i.e., $\mathrm{x}_i \neq \mathrm{x}_j$ if $i \neq j$. As a special case, we have $() \mapsto \alpha$ for the state of a one-dimensional vector of zero qubit, which is just a complex-number coefficient α working as a global phase of the (global) state. We abbreviate such an assertion with $\ulcorner \alpha \urcorner$ (or sometimes just α) and represent scaling αP as $\ulcorner \alpha \urcorner * P$. The classical points-to token $a \mapsto v$ is modelled as standard. For utility, we introduce the classical variable token $[\bar{a}]$, which represents ownership of the variables a, while ignoring their values. We also introduce the return-value token $\uparrow v$ for representing the return value of a pure

$$\bar{x} \mapsto |\psi\rangle \;\triangleq\; \left\{\!\!\left\{ (\{\bar{x}\} \mapsto |\psi\rangle, \varepsilon) \right\}\!\!\right\} \;\middle|\; \text{distinct}(\bar{x}) \right\} \qquad \ulcorner \alpha \urcorner \;\triangleq\; () \mapsto \alpha \qquad \alpha\,P \;\triangleq\; \ulcorner \alpha \urcorner * P$$

$$a \mapsto v \;\triangleq\; \left\{\!\!\left\{ (\varepsilon, \{(a, v)\}) \right\}\!\!\right\} \qquad [\bar{a}] \;\triangleq\; \mathop{\text{\Large∗}}_i \exists v.\, a_i \mapsto v \qquad \uparrow v \;\triangleq\; \text{ret} \mapsto v$$

$$\bar{x} \mapsto |\psi\rangle,\, a \mapsto v : \text{precise}$$

$$\text{BIGBMIX-SCALE}$$
$$\alpha\left(\bigoplus_{x \in I} P_x\right) \;\dashv\vdash\; \bigoplus_{x \in I} \alpha\,P_x$$

$$\text{QPOINTS-SEP}$$
$$\bar{x} \mapsto |\psi\rangle * \bar{y} \mapsto |\phi\rangle \;\dashv\vdash\; (\bar{x}, \bar{y}) \mapsto (|\psi\rangle \otimes |\phi\rangle)$$

$$\text{SCALE-QPOINTS}$$
$$\alpha\,(\bar{x} \mapsto |\psi\rangle) \;\dashv\vdash\; \bar{x} \mapsto \alpha\,|\psi\rangle$$

Fig. 12. Tokens for quantum and classical states and their proof rules.

expression (see Fig. 13 shown later). For simplicity, it is derived as a classical points-to token for a special variable ret $\in$ *Var* that clients cannot use in programs.

The tokens satisfy natural proof rules, as shown in Fig. 12. First, the quantum and classical points-to tokens are precise. Thanks to this, in particular, the phase assertion $\ulcorner \alpha \urcorner$ is precise, and combining this with BIGBMIX-UNFRAME along with BIGBMIX-FRAME, we can derive the property that mixing distributes over scaling (BIGBMIX-SCALE). Also, the quantum points-to tokens can be merged and split according to the tensor product $\otimes$ (QPOINTS-SEP). So separating conjunction $*$ represents the disentanglement of quantum states along with the ownership disjointness. Note that from it we can derive the scaling rule SCALE-QPOINTS. We also have a proof rule for permuting the qubits of a quantum points-to token; see § A.3 for the details.

Hoare triples. Now we are ready to introduce Hoare triples, as summarized in Fig. 13. We first define Hoare triples over pure expressions e. For that, we introduce the *logical semantics* $(\!|e|\!)(m) \in \mathcal{M}(Res)$ for $m \in \mathcal{M}(Res)$ that adds the value of the expression to the store at the special variable ret in every outcome. This induces a notion of weakest precondition WP $e\;\{P\} \in SLProp$, and we can derive the Hoare triple $\{P\}\;e\;\{Q\}$ as an entailment towards the weakest precondition. The definition satisfies the expected rules for triples.

We now turn to Hoare triples over *commands* C. We introduce an auxiliary function Leaves$(t) \in \mathcal{M}(Res)$ for $t \in BTree$, which collects all the leaves of the tree t as a multiset of resources. Then we define the *logical semantics* $(\!|C|\!)(m) \in \mathcal{M}(Res)$ from the denotational semantics $[\![C]\!](|\psi\rangle, S)$ by simply collecting all the leaves starting from each outcome in the input multiset. Then the weakest precondition WP $C\;\{P\} \in SLProp$ and the Hoare triple $\{P\}\;C\;\{Q\}$ are derived as expected. Notably, this Hoare triple provides a probabilistic version of total correctness, collecting the results of terminating branches only. For utility, we put classical variables $\bar{a}$ as the superscript of pre- or postconditions of Hoare triples to describe ownership of them $[\bar{a}]$.

The Hoare triples satisfy the expected proof rules. The rules for quantum operations HOARE-UNITARY, HOARE-MEASURE and storing HOARE-STORE follow immediately from the model. The sequential execution also satisfies the natural chain rule HOARE-SEQ. Notably, the Hoare triple of RapunSL naturally satisfies the frame rule HOARE-FRAME, the core source of modularity of separation logic, without any side conditions. The scale rule HOARE-SCALE immediately follows from the frame rule. Also, executions can be combined using bare mixing HOARE-BIGBMIX. For example, to handle a conditional where both booleans may occur, we can first prove $\{P_i\}\;e\;\{\uparrow i * Q_i\}$ and $\{Q_i\}\;C_i\;\{R_i\}$ for both $i \in \{0, 1\}$, apply HOARE-IF, and mix them with HOARE-BIGBMIX to obtain $\{P_0 \oplus P_1\}$ if e then C_1 else C_0 $\{R_0 \oplus R_1\}$.

Verifying loops generally. The key technical idea of RapunSL is to represent, in the model, mixed states as *multisets of outcomes*. The adoption of multisets allows RapunSL to support a very general

$$(\! (e) \!) (m) \triangleq \big\{\!\big| \, (|\psi\rangle, S \uplus \{(\mathrm{ret}, [\![e]\!]_S)\}) \mid (|\psi\rangle, S) \in_\mathsf{m} m \, \big|\!\big\}$$

$$\mathrm{WP}\, e\, \{P\} \triangleq \{m \mid (\! (e) \!)(m) \in P\} \qquad \{P\}\, e\, \{Q\} \triangleq P \vdash \mathrm{WP}\, e\, \{Q\}$$

$$\{\mathrm{emp}\}\, v\, \{\uparrow v\} \qquad \{a \mapsto v\}\, a\, \{\uparrow v * a \mapsto v\} \qquad \frac{\forall i.\, \{P\}\, e_i\, \{\uparrow v_i * P\}}{\{P\}\, op(\bar e)\, \{\uparrow op(\bar v) * P\}}$$

$$\frac{\{P\}\, e\, \{Q\}}{\{P * R\}\, e\, \{Q * R\}} \qquad \frac{\forall x \in I.\, \{P_x\}\, e\, \{Q_x\}}{\{\bigoplus_{x \in I} P_x\}\, e\, \{\bigoplus_{x \in I} Q_x\}} \qquad \frac{\{P\}\, e\, \{Q\} \quad Q \vdash Q'}{\{P\}\, e\, \{Q'\}}$$

$$\mathrm{Leaves}(t) \triangleq \big\{\!\big| \, (|\psi\rangle, S) \mid t.\bar i = \mathrm{Leaf}(|\psi\rangle, S) \, \big|\!\big\}$$

$$(\! (C) \!)(m) \triangleq \big\{\!\big| \, (|\phi\rangle, S') \mid (|\psi\rangle, S) \in_\mathsf{m} m,\, (|\phi\rangle, S') \in_\mathsf{m} \mathrm{Leaves}([\![C]\!](|\psi\rangle, S)) \, \big|\!\big\}$$

$$\mathrm{WP}\, C\, \{P\} \triangleq \{m \mid (\! (C) \!)(m) \in P\} \qquad \{P\}\, C\, \{Q\} \triangleq P \vdash \mathrm{WP}\, C\, \{Q\}$$

$$\{P\}^{\bar a}\, e\, \{Q\}^{\bar b} \triangleq \{P * [\bar a]\}\, e\, \{Q * [\bar b]\} \qquad \{P\}^{\bar a}\, C\, \{Q\}^{\bar b} \triangleq \{P * [\bar a]\}\, C\, \{Q * [\bar b]\}$$

HOARE-UNITARY
$$\{\bar x \mapsto |\psi\rangle\}\, U[\bar x]\, \{\bar x \mapsto U\,|\psi\rangle\}$$

HOARE-STORE
$$\frac{\{P\}^a\, e\, \{\uparrow v * Q\}^a}{\{P\}^a\, a \leftarrow e\, \{a \mapsto v * Q\}}$$

HOARE-MEASURE
$$\{\bar x \mapsto |\psi\rangle\}^a\, a \leftarrow M[\bar x]\, \{\bigoplus_i (a \mapsto i * \bar x \mapsto M^{(i)}|\psi\rangle)\}$$

HOARE-SEQ
$$\frac{\{P\}\, C\, \{Q\} \quad \{Q\}\, C'\, \{R\}}{\{P\}\, C; C'\, \{R\}}$$

HOARE-IF
$$\frac{\{P\}\, e\, \{\uparrow i * Q\} \quad \{Q\}\, C_i\, \{R\}}{\{P\}\, \mathrm{if}\, e\, \mathrm{then}\, C_1\, \mathrm{else}\, C_0\, \{R\}}$$

HOARE-WHILE
$$\frac{\forall n.\, \{P_n\}\, e\, \{(\uparrow 0 * Q_n) \oplus (\uparrow 1 * R_n)\} \quad \forall n.\, \{R_n\}\, C\, \{P_{n+1}\}}{\{P_0\}\, \mathrm{while}\, e\, \mathrm{do}\, C\, \{\bigoplus_{n \in \mathbb N} Q_n\}}$$

HOARE-FRAME
$$\frac{\{P\}\, C\, \{Q\}}{\{P * R\}\, C\, \{Q * R\}}$$

HOARE-BIGMIX
$$\frac{\forall x \in I.\, \{P_x\}\, C\, \{Q_x\}}{\{\bigoplus_{x \in I} P_x\}\, C\, \{\bigoplus_{x \in I} Q_x\}}$$

HOARE-POST
$$\frac{\{P\}\, C\, \{Q\} \quad Q \vdash Q'}{\{P\}\, C\, \{Q'\}}$$

HOARE-SCALE
$$\frac{\{P\}\, C\, \{Q\}}{\{\alpha P\}\, C\, \{\alpha Q\}}$$

Fig. 13. Hoare triples and their proof rules.

and expressive handling of *unbounded loops* by the rule HOARE-WHILE. Here, we benefit from supporting the *infinitary* mixing $\bigoplus_{n \in \mathbb N} P_n$, modelled as the infinite sum of multisets described by P_n, that is, the limit of all finite mixing $\bigoplus_{n \le N} P_n$. The rule uses two assertions indexed by the current iteration n: P_n describes the state at the beginning of the $(n + 1)$-th iteration (and consequently, at the end of the n-th); Q_n describes the states the system can be in at the exit of the loop. Then the rule asserts that the postcondition of the whole loop is simply the infinite mixing, or the limit of all the finite mixing, of Q_n assertions. Although RapunSL provides natural proof principles for reasoning about such infinite mixing within the logic, which works for useful examples, more thoroughly exploring the elimination principles of $\bigoplus_{n \in \mathbb N}$ is left for future work.

Notably, this achieves reasoning about *probabilistic total correctness*. Since all the outcomes need to be accounted for when predicating over such resources, assertions can insist on the global probability mass being some definite quantity by looking at the norms of all the outcomes. This

$$(\mathsf{X} \mapsto |\psi\rangle) +_{Qstate} (\mathsf{Y} \mapsto |\phi\rangle) \triangleq \mathsf{X} \mapsto (|\psi\rangle + |\phi\rangle) \quad \text{if } \mathsf{X} = \mathsf{Y}$$

$$(|\psi\rangle, S) +_{Res} (|\phi\rangle, S') \triangleq (|\psi\rangle +_{Qstate} |\phi\rangle, S) \quad \text{if } S = S'$$

Fig. 14. Resource ring structure over *Qstate* and *Res*.

allows us to specify *almost-sure termination*, or even specify the *exact probability* of termination when non-termination can happen with non-zero probability.

4.5 The Sum, the Heart of RapunSL

Now we also want the sum $P + Q$ over SL assertions, the heart of RapunSL. Extending the PCM structure for separating conjunction $*$, we introduce what we dub the *resource ring*, a new algebra equipped with the *partial* sum $+$.

Definition 5 (Resource ring). A resource ring $(A, \varepsilon_A \in A, \cdot_A : A \times A \rightharpoonup A, +_A : A \times A \rightharpoonup A)$ is a PCM $(A, \varepsilon_A, \cdot_A)$ equipped with the partial sum $+_A$ (we may omit the subscript) such that $a +_A b = b +_A a$, $(a +_A b) +_A c = a +_A (b +_A c)$, and $(a +_A b) \cdot_A c = a \cdot_A c +_A b \cdot_A c$.[8]　□

We can define the resource ring structure over *Qstate* and *Res* as presented in Fig. 14. As expected, the vector sum $|\psi\rangle + |\phi\rangle$ is used for defining $+_{Qstate}$. The point is that the sum is defined only when the two arguments agree on the same classical information, namely, the domain of qubits and the store, which makes the sum partial.

Now, how can we define the sum $P + Q$ over SL assertions? Again, recall that SL assertions are a set of *multisets* of resources. Naively, we want something like the 'parallel pointwise sum' of multisets m and m' as the sum for the resource ring. For example, something like $\{\!\{a, a'\}\!\} + \{\!\{b, b'\}\!\} = \{\!\{a + b, a' + b'\}\!\}$. However, this makes the sum ill-defined, because the multiset forgets the 'order' of the elements. For the example above, the sum can also be $\{\!\{a + b', a' + b\}\!\}$. In other words, to take the sum, the elements of the two multisets should be matched *in parallel*, and there can be multiple ways to match the elements of two multisets. This is quite different from separating conjunction $*$, where the product is just distributive over the multiset elements.

To formalize the matching over multisets, we introduce the notion of the *multiset bijection*.

Definition 6 (Multiset bijection). A multiset bijection $r : m \cong_m m'$ between two multisets $m \in \mathcal{M}(A)$ and $m' \in \mathcal{M}(B)$ is a multiset $r \in \mathcal{M}(A \times B)$ such that $m = \{\!\{a \mid (a, b) \in_m r\}\!\}$ and $m' = \{\!\{b \mid (a, b) \in_m r\}\!\}$. In other words, a multiset bijection between $\lfloor f : I \to A\rfloor$ and $\lfloor g : I \to B\rfloor$ is $\lfloor h : I \to A \times B\rfloor$ such that $(h\,i).0 = f\,i$ and $(h\,i).1 = g\,i$ (and there is no multiset bijection between multisets of different sizes).　□

Using multiset bijections, we can define the sum of SL assertions, as presented in Fig. 15. First, the partial sum $m +_r m'$ over multisets can be defined once we fix the multiset bijection $r : m \cong_m m'$. Note that $\mathcal{M}(Res)$ itself does not form a resource ring, due to the extra parameter r. Now we define the sum $P + Q$ of SL assertions by taking the sum of multisets using an arbitrary multiset bijection. The sum satisfies natural rules SUM-MONO, SUM-COMM and SUM-ASSOC. Quantum points-to tokens sum up according to the vector sum (QPOINTS-SUM), as the model suggests.

The core rule of RapunSL is HOARE-SUM, summing Hoare triples according to $+$. Intuitively, this holds because quantum programs cannot change what to execute depending on quantum state vectors. The precondition $P + P'$ takes the sum of two initial states, taken respectively from P and P', that agree on the classical states and can only differ in the quantum states. The two executions from these initial states give behaviour trees of the same form. The behaviour tree for the sum initial

---

[8] We use '=' for the Kleene equality, where the left-hand side is defined if and only if the right-hand side is.

$$m +_r m' \triangleq \{\!| a + b \mid (a, b) \in_{\mathsf{m}} m |\!\} \quad \text{where } r : m \cong_{\mathsf{m}} m'$$

$$P + Q \triangleq \left\{ m +_r m' \mid m \in P, m' \in Q, r : m \cong_{\mathsf{m}} m' \right\}$$

SUM-MONO

$$\frac{P \vdash P' \qquad Q \vdash Q'}{P + Q \vdash P' + Q'}$$

SUM-COMM

$$P + Q \dashv\vdash Q + P$$

SUM-ASSOC

$$(P + Q) + R \dashv\vdash P + (Q + R)$$

QPOINTS-SUM

$$\bar{\mathsf{x}} \mapsto |\psi\rangle \; + \; \bar{\mathsf{x}} \mapsto |\phi\rangle \; \dashv\vdash \; \bar{\mathsf{x}} \mapsto (|\psi\rangle + |\phi\rangle)$$

HOARE-SUM

$$\frac{\{P\}\,C\,\{Q\} \quad \{P'\}\,C\,\{Q'\}}{\{P + P'\}\,C\,\{Q + Q'\}}$$

HOARE-FRAME-UNTANGLE

$$\frac{\forall |\psi\rangle . \left\{ \bar{\mathsf{x}} \mapsto |\psi\rangle \right\} C \left\{ \bar{\mathsf{x}} \mapsto U\,|\psi\rangle \right\}}{\left\{ (\bar{\mathsf{x}}, \bar{\mathsf{y}}) \mapsto |\phi\rangle \right\} C \left\{ (\bar{\mathsf{x}}, \bar{\mathsf{y}}) \mapsto U_{\bar{\mathsf{x}}} |\phi\rangle \right\}}$$

SUM-BIGMIX

$$\bigoplus_{x \in I} (P_x + Q_x) \vdash \left(\bigoplus_{x \in I} P_x \right) + \left(\bigoplus_{x \in I} Q_x \right)$$

SUM-FRAME

$$(P + Q) * R \vdash (P * R) + (Q * R)$$

Fig. 15. Sum over $\mathcal{M}(Res)$, the sum connective and its proof rules.

state can simply be obtained by summing the corresponding leaves of the two behaviour trees. The reasoning principle HOARE-SUM is remarkable. For example, from this rule, HOARE-SCALE and HOARE-FRAME, we can derive the rule HOARE-FRAME-UNTANGLE that can reason about a (possibly) entangled state $|\phi\rangle$ from the behaviours of the program C on each input $|\psi\rangle$, by decomposing $|\phi\rangle$ to a *linear combination* of *disentangled states*.[9] This generalizes the reasoning showcased in § 2.1.

Also, bare mixing over sum entails sum of bare mixing SUM-BIGMIX and sum enjoys the frame rule SUM-FRAME. However, the converses of these two rules do not hold in general and require careful side conditions. Technically, this comes from the freedom of the multiset bijection r or the way to match the elements in the model of sum $P + Q$. Later, we will discuss the side conditions for the two converses.

Remark 7 (Right adjoints of $\wedge$, $*$, $\oplus$ and $+$). As usual, usual and separating conjunctions $\wedge$, $*$ have the right adjoints $\rightarrow$ and $-\!*$ (magic wand). Interestingly, in RapunSL, mixing $\oplus$ and sum $+$ also have the right adjoints $-\!\oplus$ and $-\!+$. See § A.2 for the details.

Incompatibility and unambiguity. In order to ensure the uniqueness of the multiset bijection in the assertion sum $P + Q$, we introduce a notion of *incompatibility* and a derived notion of *unambiguity*, as in Fig. 16. First, the incompatibility $a \# b$ over resources $a, b \in Res$ is defined as having different values for some variable. Then we naturally lift that to the incompatibility over propositions $P \# Q$. We have natural proof rules for the incompatibility. If any pair of the assertions at different indices are incompatible, sum of mixing can be taken in parallel (BIGMIX-SUM), which is the converse of SUM-BIGMIX.

Using the incompatibility between resources, we also define the *unambiguity P : unambig* of an SL assertion P, meaning that for any multiset of the assertion, any two distinct elements of the multiset are incompatible with each other. Intuitively, this means all the branches have different stored values. Here, we introduce an auxiliary predicate P : nonnb, meaning that any multiset in P is not empty (i.e., contains some behaviours). The predicate satisfies natural proof rules (see § A.3 for the details). The rule SUM-UNFRAME has three side conditions: the precision of R, the behaviour non-emptiness (nonnb) of R, and the unambiguity of the assertions P and R. The precision of R is

---

[9] Here, the universal quantification $\forall |\psi\rangle$ can range just over some basis of a finite size.

$$(|\psi\rangle, S) \mathbin{\#} (|\phi\rangle, S') \ \triangleq \ \exists a \in \mathrm{dom}\, S \cap \mathrm{dom}\, S'. \ S[a] \neq S'[a]$$

$$P \mathbin{\#} Q \ \triangleq \ \forall m \in P. \ \forall m' \in Q. \ \forall a \in_m m. \ \forall b \in_m m'. \ a \mathbin{\#} b$$

$$\frac{v \neq w}{a \mapsto v \ \mathbin{\#} \ a \mapsto w} \qquad \frac{P \mathbin{\#} Q}{Q \mathbin{\#} P} \qquad \frac{P \mathbin{\#} Q \quad Q' \vdash Q}{P \mathbin{\#} Q'} \qquad \frac{P \mathbin{\#} Q}{P * R \mathbin{\#} Q} \qquad \frac{\forall x. \ (P_x \mathbin{\#} Q)}{\bigoplus_{x \in I} P_x \mathbin{\#} Q}$$

$$P : \mathsf{unambig} \ \triangleq \ \forall m \in P. \ \forall \{\!\{a, b\}\!\} \subseteq_m m. \ a \mathbin{\#} b \qquad P : \mathsf{nonnb} \ \triangleq \ \forall m \in P. \ m \neq \{\!\{\ \}\!\}$$

$$\mathsf{emp}, \ \bar{x} \mapsto |\psi\rangle, \ a \mapsto v : \mathsf{unambig} \qquad \frac{P : \mathsf{unambig} \quad Q \vdash P}{Q : \mathsf{unambig}} \qquad \frac{P, Q : \mathsf{unambig}}{P * Q, \ P + Q : \mathsf{unambig}}$$

BIGBMIX-UNAMBIG

$$\frac{\forall x \in I. \ (P_x : \mathsf{unambig}) \qquad \forall x, y \in I \ \text{s.t.} \ x \neq y. \ P_x \mathbin{\#} P_y}{\bigoplus_{x \in I} P_x : \mathsf{unambig}}$$

SUM-PRECISE

$$\frac{P, Q : \mathsf{precise} \qquad P : \mathsf{unambig}}{P + Q : \mathsf{precise}}$$

BIGBMIX-SUM

$$\frac{\forall x, y \in I \ \text{s.t.} \ x \neq y. \ P_x \mathbin{\#} Q_y}{\left(\bigoplus_{x \in I} P_x\right) + \left(\bigoplus_{x \in I} Q_x\right) \vdash \bigoplus_{x \in I} (P_x + Q_x)}$$

SUM-UNFRAME

$$\frac{P, R : \mathsf{unambig} \qquad R : \mathsf{precise}, \mathsf{nonnb}}{(P * R) + (Q * R) \vdash (P + Q) * R}$$

Fig. 16. Incompatibility and unambiguity.

$$P \ {}_0\!\oplus_1^\iota Q \ \triangleq \ (\iota \mapsto 0 * P) \oplus (\iota \mapsto 1 * Q) \qquad\qquad \bigoplus_{x \in I}^\iota P_x \ \triangleq \ \bigoplus_{x \in I} (\iota \mapsto x * P_x)$$

BIGMIX-SUM

$$\left(\bigoplus_{x \in I}^\iota P_x\right) + \left(\bigoplus_{x \in I}^\iota Q_x\right) \ \dashv\vdash \ \bigoplus_{x \in I}^\iota (P_x + Q_x)$$

BIGMIX-UNAMBIG

$$\frac{\forall x \in I. \ (P_x : \mathsf{unambig})}{\bigoplus_{x \in I}^\iota P_x : \mathsf{unambig}}$$

HOARE-MEASURE-MIX

$$\left\{ \bar{x} \mapsto |\psi\rangle \right\}^\iota M^\iota[\bar{x}] \left\{ \bigoplus_i^\iota \bar{x} \mapsto M^{(i)} |\psi\rangle \right\}$$

Fig. 17. Tagged mixing and its derived proof rules.

needed for the same reason as BIGBMIX-UNFRAME. The behaviour non-emptiness excludes a subtle corner case. For example, if $R = \mathsf{nb}$ (violating $R : \mathsf{nonnb}$), $P = a \mapsto 0$ and $Q = a \mapsto 1$, then the left-hand side is equivalent to nb, while the right-hand side entails the falsehood $\bot \triangleq \varnothing$ because $P + Q$ entails $\bot$. The key side condition is the unambiguity of P and R. For example, if $R = \alpha \oplus \beta$ (violating $R : \mathsf{unambig}$) and $P = Q = \mathsf{emp}$, the left-hand side contains $(\alpha + \beta) \oplus (\beta + \alpha)$, while the right-hand side does not. We similarly have unsoundness if both P and Q are ambiguous.[10] The rule SUM-PRECISE also requires the unambiguity for a similar reason. We carefully designed the side conditions to achieve both soundness and flexibility.

4.6 Tagged Mixing

Finally, we formally model the *tagged mixing*, introduced in the overview § 2.2. Figure 17 shows its definition and proof rules. Generalizing binary tagged mixing $P \ {}_0\!\oplus_1^\iota Q$, we also introduce tagged mixing $\bigoplus_{x \in I}^\iota P_x$ indexed over any set I. Tagged mixing is derived from bare mixing § 4.2 by tagging each argument with a classical points-to token $\iota \mapsto x$ whose value x allows for the

---

[10] For example, let us set $P = Q = \ulcorner \alpha \urcorner \oplus \ulcorner \beta \urcorner$ (neither $P : \mathsf{unambig}$ nor $Q : \mathsf{unambig}$ holds) and $R = \iota \mapsto 0 \oplus \iota \mapsto 1$. Then the left-hand side of SUM-UNFRAME contains $\iota \mapsto 0 * (\ulcorner \alpha + \beta \urcorner \oplus \ulcorner \beta + \alpha \urcorner) \oplus \iota \mapsto 1 * (\ulcorner \alpha + \alpha \urcorner \oplus \ulcorner \beta + \beta \urcorner)$ while the right-hand side does not.

$$P: \text{frameable} \quad \triangleq \quad P: \text{precise, unambig, nonnb}$$

$$\frac{P: \text{frameable} \quad Q \vdash P}{Q: \text{frameable}} \qquad \frac{P, Q: \text{frameable}}{P * Q, \ P + Q: \text{frameable}} \qquad \frac{I \neq \varnothing \quad \forall i \in I. \ P_i: \text{frameable}}{\bigoplus_{i \in I}^t P_i: \text{frameable}}$$

$$\text{Frameable subclass} \quad FProp \ni F \ ::= \ a \mapsto v \ \mid \ \mathsf{x} \mapsto |\psi\rangle \ \mid \ \forall x \in A. \ F_x \qquad \frac{F \in FProp}{F: \text{frameable}}$$
$$\mid \ F * F' \ \mid \ F + F' \ \mid \ \bigoplus_{i \in I(\neq \varnothing)}^t F_i$$

BIGMIX-FRAME-FRAMEABLE

$$\frac{Q: \text{frameable}}{\bigoplus_{x \in I} (P_x * Q) \ \dashv\vdash \ (\bigoplus_{x \in I} P_x) * Q}$$

SUM-FRAME-FRAMEABLE

$$\frac{P, R: \text{frameable}}{(P * R) + (Q * R) \ \dashv\vdash \ (P + Q) * R}$$

Fig. 18. Frameability.

identification of the outcomes. Tagged mixing is very well-behaved with respect to the other connectives. Remarkably, sum + can be taken over tagged mixing in parallel (BIGMIX-SUM), thanks to the classical points-to token automatically ensuring the incompatibility conditions of BIGMIX-SUM. This is one of the key ideas of RapunSL, as explained in § 2.2. Also, tagged mixing is unambiguous simply when the arguments are all unambiguous (BIGMIX-UNAMBIG). Notably, the Hoare rule for measurement (HOARE-MEASURE) can be reformulated using tagged mixing (HOARE-MEASURE-MIX). Tagged mixing also enjoys natural proof rules derived from rules on bare mixing; see § A.4 for the details. In summary, using tagged mixing, we can enjoy natural reasoning about mixed quantum states introduced by measurements.

4.7 Handling Complexity

Frameability. So far, we have introduced various proof rules to RapunSL. Our primary goal has been to identify the most general sound rules that achieve the desired modularity. It is unsurprising that some of them (e.g., SUM-UNFRAME) carry intricate side conditions, but one might worry that they are too complex for practical use.

In fact, these complex side conditions can be greatly simplified by focusing on a well-behaved class of SL assertions, which we call *frameable*. The rules are summarized in Fig. 18. Frameability frameable is simply the conjunction of precise, unambig and nonnb. As shown in Fig. 18, frameable assertions have nice closure properties. This enables us to define a handy class *FProp* of frameable SL assertions, which are obviously frameable by construction.[11] Notably, frameable assertions can be a frame for mixing and sum (BIGMIX-FRAME-FRAMEABLE, SUM-FRAME-FRAMEABLE), simplifying the unframing rules BIGMIX-UNFRAME and SUM-UNFRAME.

Abstraction. We can go further and provide *abstraction*, as discussed in § 2.4. We want to abstract away the complicated 'factor' $P_\oplus$ of the outcome of the program mCNOT, so that clients of mCNOT can safely forget the exact form of $P_\oplus$ and equate mCNOT with CNOT.

Frameability is a great match for this purpose. As long as clients know that $P_\oplus$ is frameable, they can treat it as a frame for mixing and sum to achieve the outcome- and basis-locality (we will explain this in more detail later in § 5.3).

For a detailed analysis of probability, we introduce the probability predicate $P: \text{prob } p$, shown in Fig. 19.[12] The predicate simply means that every state represented by P has the total probability p.

---

[11] For simplicity, we describe the class syntactically in BNF. The mixing here is tagged and runs over a non-empty domain.
[12] Recall that RapunSL can verify the termination probability, which is non-trivial for programs with loops, as discussed in Verifying loops generally, § 4.4.

$$P : \mathsf{prob}\, p \quad \triangleq \quad \forall m \in P. \sum_{(|\psi\rangle, S) \in_m m} \| |\psi\rangle \|^2 = p \qquad \frac{P : \mathsf{prob}\, p \quad Q : \mathsf{prob}\, q}{P * Q : \mathsf{prob}\, pq} \qquad \frac{P_i : \mathsf{prob}\, p_i}{\bigoplus_i P_i : \mathsf{prob}\, \sum_i p_i}$$

Fig. 19. Probability.
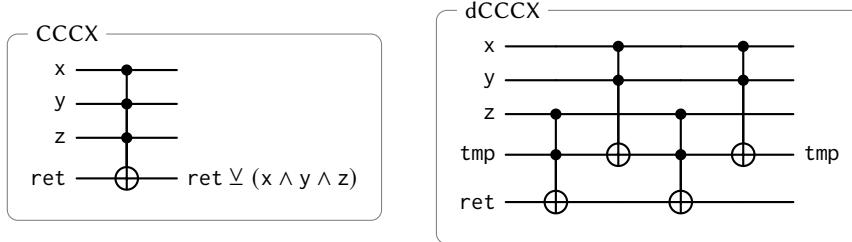
Fig. 20. A CCCX gate (left) and its encoding dCCCX using dirty qubit tmp (right).

A factor like mCNOT's $P_\oplus$ typically satisfies prob 1. So in a simple setting we can abstract such a factor as frameable and prob 1. The probability behaves well over separating conjunction $*$ and mixing $\bigoplus$. To reason about the probability of sums $+$, we can also introduce the inner product (and orthogonality as a special case) of SL assertions, naturally extending the usual vector calculus; see § A.7 for the details.

5 Case Studies

This section presents a number of practical examples of RapunSL verification of quantum programs from the literature, confirming that our logic:

G1: Enables modular reasoning by using the three locality principles;

G2: Effectively applies basis-locality to real-world programs involving measurements;

G3: Demonstrates that abstracting the global phase, as presented in § 2.4, simplifies the specification and makes the reasoning scalable;

G4: Supports standard probabilistic features of quantum programs, such as proving almost-sure termination of while loops.

We start with relatively simple case studies and gradually move to more complex ones. All the details of the proofs and additional examples are available in § B. Here, we sketch the key ideas of the proofs and highlight the significance of the examples.

Remark 8 (Utility tagged mixing notation). For brevity, we introduce the following shorthand for tagged mixing, abusing a classical variable ι to represent the value it stores: $\bigoplus^\iota P \triangleq \bigoplus^\iota_x P[x/\iota]$. Here, the domain of the value x should be properly inferred and is typically set to $\{0, 1\}$. Precisely speaking, we think of some syntax for propositions P to define the substitution $P[x/\iota]$ here. Also, we write $\bigoplus^{\iota_1, \dots, \iota_n} P$ for $\bigoplus^{\iota_1} \cdots \bigoplus^{\iota_n} P$.

5.1 Dirty Qubit: Implementation of CCCX by Toffoli Gates

The first example is an implementation of the CCCX gate using a dirty qubit, which is a concrete example of the program we mentioned in § 2.1. We show how RapunSL enables the context-independent identification of CCCX gate with its implementation (**G1**).

As depicted in Fig. 20 (left), the CCCX gate is a 4-qubit multi-controlled-NOT gate that computes ret $\leftarrow$ ret $\veebar$ (x $\wedge$ y $\wedge$ z), where $\veebar$ is the exclusive-OR operation. The CCCX gate can be implemented

using the CCX (called Toffoli) gate with a dirty qubit serving as the auxiliary qubit, as illustrated in Fig. 20 (right). Note that the CCX gate computes $\text{ret} \leftarrow \text{ret} \veebar (\text{x} \wedge \text{y})$.

It is important is that we must return a dirty qubit to its original state after its temporary use, because it may be required by another computation. Thus, the goal here is to verify that Fig. 20 (right) enjoys this condition and implements CCCX correctly; however, two challenges arise here. First, the state of tmp is unknown. Second, tmp may be entangled with x, y, z, or other qubits.

The idea for overcoming these challenges is to decompose tmp into a basis using HOARE-SUM. After the decomposition, tmp can be regarded as disentangled from the other qubits for each basis state. Consequently, the qubits other than x, y, z and tmp can later be composed by using the frame rule. Moreover, we can obtain reasoning that is independent of the initial state of tmp because the reasoning results for each basis state can be linearly combined.

In fact, this idea makes it possible to verify the correctness of Fig. 20 (right). First, the specification of Fig. 20 (right) for each basis can be expressed as

$$\left\{ (\text{x}, \text{y}, \text{z}) \mapsto |ijk\rangle * \text{tmp} \mapsto |\ell\rangle * \text{ret} \mapsto |m\rangle \right\} \ \text{dCCCX}[\text{x}, \text{y}, \text{z}, \text{tmp}, \text{ret}]$$

$$\left\{ (\text{x}, \text{y}, \text{z}) \mapsto |ijk\rangle * \text{tmp} \mapsto |\ell\rangle * \text{ret} \mapsto |m \veebar (i \wedge j \wedge k)\rangle \right\}.$$

Then, we scale the global phase (11) and sum up derivations (12) to prove for any entangled case:

$$\frac{\forall i, j, k, \ell, m \in \{0,1\}. \quad \begin{array}{l} \{ (\text{x}, \text{y}, \text{z}, \text{tmp}, \text{ret}) \mapsto |ijk\ell m\rangle \} \ \text{dCCCX} \\ \{ (\text{x}, \text{y}, \text{z}, \text{tmp}, \text{ret}) \mapsto (\text{CCCX}_{\text{x,y,z,ret}} \otimes \text{id}_{\text{tmp}}) |ijk\ell m\rangle \} \end{array}}{\forall i, j, k, \ell, m \in \{0,1\}.} \ (11)$$

$$\frac{\begin{array}{l} \{ (\text{x}, \text{y}, \text{z}, \text{tmp}, \text{ret}) \mapsto \alpha_{ijk\ell m} |ijk\ell m\rangle * \bar{\text{w}} \mapsto |\psi_{ijk\ell m}\rangle \} \ \text{dCCCX} \\ \{ (\text{x}, \text{y}, \text{z}, \text{tmp}, \text{ret}) \mapsto (\text{CCCX}_{\text{x,y,z,ret}} \otimes \text{id}_{\text{tmp}}) \alpha_{ijk\ell m} |ijk\ell m\rangle * \bar{\text{w}} \mapsto |\psi_{ijk\ell m}\rangle \} \end{array}}{\begin{array}{l} \{ (\text{x}, \text{y}, \text{z}, \text{tmp}, \text{ret}, \bar{\text{w}}) \mapsto \sum_{i,j,k,\ell,m} \alpha_{ijk\ell m} |ijk\ell m\rangle |\psi_{ijk\ell m}\rangle \} \ \text{dCCCX} \\ \{ (\text{x}, \text{y}, \text{z}, \text{tmp}, \text{ret}, \bar{\text{w}}) \mapsto (\text{CCCX}_{\text{x,y,z,ret}} \otimes \text{id}_{\text{tmp},\bar{\text{w}}}) \sum_{i,j,k,\ell,m} \alpha_{ijk\ell m} |ijk\ell m\rangle |\psi_{ijk\ell m}\rangle \} \end{array}} \ (12)$$

The state $\sum_{i,j,k,\ell,m} \alpha_{ijk\ell m} |ijk\ell m\rangle |\psi_{ijk\ell m}\rangle$ can represent any entangled state of the qubits, not only including the dirty qubit tmp, but also uninvolved qubits $\bar{\text{w}}$ introduced as a frame, proving that the dCCCX gate is indeed a valid implementation of CCCX regardless of the context.

5.2 Quantum Teleportation

The second example we demonstrate is the quantum teleportation protocol. In this protocol, there are two parties: Alice and Bob, and they communicate two bits of classical information to teleport one qubit of quantum information from Alice to Bob. Here, we prove the correctness of the protocol in a modular way (G1). That is, we analyse the behaviour of Alice and Bob separately and then combine the results to prove the correctness of the whole protocol. This protocol involves the use of a shared entangled state and measurements (G2).

The quantum teleportation protocol can be implemented as follows:

$$\begin{aligned} \text{Bell}[\text{x}, \text{y}] &\triangleq \text{H}[\text{x}]; \text{CX}[\text{x}, \text{y}] \\ \text{Alice}^{a,b}[\text{x}, \text{y}] &\triangleq \text{CX}[\text{x}, \text{y}]; \text{H}[\text{x}]; a \leftarrow \text{M}_\text{Z}[\text{x}]; b \leftarrow \text{M}_\text{Z}[\text{y}] \\ \text{Bob}^{a,b}[\text{z}] &\triangleq \text{if } a \text{ then } \text{X}[\text{z}]; \text{if } b \text{ then } \text{Z}[\text{z}] \\ \text{Teleport}^{a,b}[\text{x}, \text{y}, \text{z}] &\triangleq \text{Bell}[\text{y}, \text{z}]; \text{Alice}^{a,b}[\text{x}, \text{y}]; \text{Bob}^{a,b}[\text{z}]. \end{aligned}$$

The protocol begins by preparing the Bell state $|\text{Bell}\rangle \triangleq \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ shared between Alice and Bob by using Bell[y, z]. Although Alice's qubit y and Bob's qubit z are entangled, in RapunSL, we can reason about $\text{Alice}^{a,b}[\text{x}, \text{y}]$ and $\text{Bob}^{a,b}[\text{z}]$ separately by considering the case in which y and

z are in some classical state. Consequently, if we introduce a Boolean variable x to represent the basis of the qubit x, we obtain:[13]

$$\big\{ (x,y) \mapsto |xi\rangle \big\}^{a,b} \ \text{Alice}^{a,b}[x,y] \ \big\{ \tfrac{1}{\sqrt{2}} \bigoplus^{a,b} (-1)^{xa} \cdot \delta_{x \vee i, b} \cdot (x,y) \mapsto |ab\rangle \big\}$$

$$\big\{ z \mapsto |i\rangle \ast a \mapsto a \ast b \mapsto b \big\} \ \text{Bob}^{a,b}[z] \ \big\{ (-1)^{xa}(z \mapsto |b \vee i\rangle) \ast a \mapsto a \ast b \mapsto b \big\}$$

Now, we verify the correctness of the whole protocol.

$$\frac{\big\{ (x,y,z) \mapsto |xii\rangle \big\}^{a,b} \ \text{Alice}[x,y]; \text{Bob}^{a,b}[z] \ \big\{ \tfrac{1}{\sqrt{2}} \bigoplus^{a,b} \delta_{x \vee i, b} \ast (x,y,z) \mapsto |ab(b \vee i)\rangle \big\}}{\big\{ (x,y,z) \mapsto \sum_{i=0,1} \tfrac{1}{\sqrt{2}} |xii\rangle \big\}^{a,b} \ \text{Alice}[x,y]; \text{Bob}^{a,b}[z] \ \big\{ \tfrac{1}{2} \bigoplus^{a,b} (x,y,z) \mapsto |abx\rangle \big\}} \quad (13)$$

$$\frac{}{\big\{ x \mapsto |\psi\rangle \ast (y,z) \mapsto |\text{Bell}\rangle \big\}^{a,b} \ \text{Alice}[x,y]; \text{Bob}^{a,b}[z] \ \big\{ \tfrac{1}{2} \bigoplus^{a,b} (x,y) \mapsto |ab\rangle \ast z \mapsto |\psi\rangle \big\}} \quad (14)$$

$$\frac{}{\big\{ x \mapsto |\psi\rangle \ast (y,z) \mapsto |\text{Bell}\rangle \big\}^{a,b} \ \text{Alice}[x,y]; \text{Bob}^{a,b}[z] \ \big\{ z \mapsto |\psi\rangle \ast \tfrac{1}{2} \bigoplus^{a,b} (x,y) \mapsto |ab\rangle \big\}} \quad (15)$$

We frame $z \mapsto |i\rangle$ into Alice's Hoare triple and frame Alice's postcondition into Bob's Hoare triple to obtain the first line. At (13), we use HOARE-SUM to combine them to obtain the Bell state in the precondition. To generalize the input state $|x\rangle$ to $|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle$, we multiply by the scalar α_x and sum over x at (14). We use BIGBMIX-UNFRAME at (15) to separate off the state $P \triangleq \tfrac{1}{2} \bigoplus^{a,b} (x,y) \mapsto |ab\rangle$, which is frameable and prob 1.

5.3 Lattice Surgery: Implementation of CNOT with Measurements

As we sketched in §2.4, we can verify a sort of equivalence between the CNOT gate and its implementation using 2-qubit measurements in Fig. 2. This technique of replacing a CNOT gate with 1-qubit gates and 1- or 2-qubit measurements is the core idea of lattice surgery [Fowler and Gidney 2019], which can be used for fault-tolerant quantum computing. This example illustrates how RapunSL achieves the verification of the program with measurements (G2), and how we can hide the information of the global phase that depends on the measurement outcomes but does not affect the final state (G3).

The program we verify, mCNOT, is defined by

$$\text{mCNOT}^{\iota,\kappa,\lambda}[x,y,z] \quad \triangleq \quad \text{if } M^{\iota}_{XX}[y,z] \text{ then } Z[x];$$
$$\text{if } M^{\kappa}_{ZZ}[x,y] \text{ then } X[z]; \ H[y]; \ \text{if } M^{\lambda}_{Z}[y] \text{ then } Z[x].$$

We introduce shorthand if $M^{\iota}[\bar{x}]$ then $C \triangleq M^{\iota}[\bar{x}]$; if ι then C. Here, classical variables ι, κ and λ are used for storing the results of measurements. Using our logic, we can easily derive the following Hoare triple by step-by-step reasoning:

$$\forall a,b \in \{0,1\}. \quad \big\{ x \mapsto |a\rangle \ast y \mapsto |0\rangle \ast z \mapsto H |b\rangle \big\}^{\iota,\kappa,\lambda} \quad \text{mCNOT}^{\iota,\kappa,\lambda}[x,y,z]$$

$$\big\{ \tfrac{1}{2\sqrt{2}} \bigoplus^{\iota,\kappa,\lambda} (-1)^{ab+\iota\kappa+\kappa\lambda} \cdot (x,y) \mapsto |a\lambda\rangle \ast z \mapsto H |b\rangle \big\}$$

The postcondition can be simplified as follows:

$$\tfrac{1}{2\sqrt{2}} \bigoplus^{\iota,\kappa,\lambda} (-1)^{ab+\iota\kappa+\kappa\lambda} \cdot (x,y) \mapsto |a\lambda\rangle \ast z \mapsto H |b\rangle$$

$$\vdash \quad (-1)^{ab} \tfrac{1}{2\sqrt{2}} \cdot (x,z) \mapsto |a\rangle \otimes H |b\rangle \ast \bigoplus^{\iota,\kappa,\lambda} (-1)^{\iota\kappa+\kappa\lambda} \cdot y \mapsto |\lambda\rangle$$

$$\vdash \quad (x,z) \mapsto CX(|a\rangle \otimes H |b\rangle) \ast \tfrac{1}{2\sqrt{2}} \bigoplus^{\iota,\kappa,\lambda} (-1)^{\iota\kappa+\kappa\lambda} \cdot y \mapsto |\lambda\rangle .$$

---
[13] We write $\delta_{a,b}$ for the Kronecker delta, returning 1 if $a = b$ and 0 otherwise.
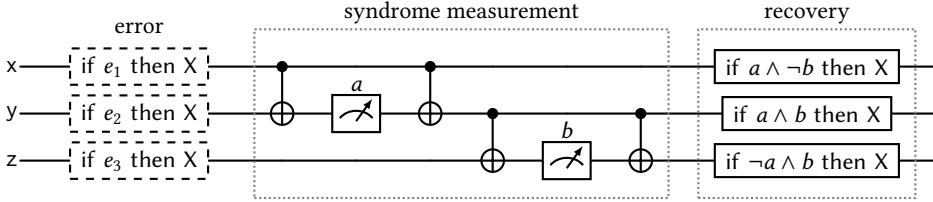
Fig. 21. Bit-flip code error correction procedure.

Now, let $P_\oplus \triangleq \frac{1}{2\sqrt{2}} \bigoplus^{\iota,\kappa,\lambda} (-1)^{\iota\kappa+\kappa\lambda} \cdot \mathsf{y} \mapsto |\lambda\rangle$. It satisfies the nice property $P_\oplus$: frameable, prob 1, i.e., it is frameable and its total probability is 1 (recall Figs. 18 and 19, § 4.5). With HOARE-SCALE and HOARE-SUM, we can derive the following general specification of mCNOT for any input state $|\psi\rangle$:

$$\forall |\psi\rangle . \ \left\{ (\mathsf{x}, \mathsf{z}) \mapsto |\psi\rangle * \mathsf{y} \mapsto |0\rangle \right\}^{\iota,\kappa,\lambda} \mathsf{mCNOT}^{\iota,\kappa,\lambda}[\mathsf{x}, \mathsf{y}, \mathsf{z}] \ \left\{ (\mathsf{x}, \mathsf{z}) \mapsto \mathsf{CX}|\psi\rangle * P_\oplus \right\}.$$

From this assertion, we can further derive the following, saying that mCNOT behaves like CNOT even if x and z are entangled with any other qubits:

$$\forall |\psi\rangle . \ \left\{ (\mathsf{x}, \mathsf{z}, \bar{\mathsf{w}}) \mapsto |\psi\rangle * \mathsf{y} \mapsto |0\rangle \right\}^{\iota,\kappa,\lambda} \mathsf{mCNOT}^{\iota,\kappa,\lambda}[\mathsf{x}, \mathsf{y}, \mathsf{z}] \ \left\{ (\mathsf{x}, \mathsf{z}, \bar{\mathsf{w}}) \mapsto \mathsf{CX}_{\mathsf{x},\mathsf{z}}|\psi\rangle * P_\oplus \right\}.$$

Remarkably, for the derivation (which uses HOARE-SUM), it suffices to know that $P_\oplus$ is *some* assertion satisfying frameable, without knowing the exact form of $P_\oplus$, thanks to the fact that frameable assertions can freely frame into and out of sum + (SUM-FRAME-FRAMEABLE). We can further enrich this for mixing $\bigoplus$ (using HOARE-BIGMIX and BIGMIX-FRAME-FRAMEABLE). This demonstrates the power of abstraction (G3). Note that, when we have some succeeding program C and want to prove the whole specification, just framing $P_\oplus$ suffices:

$$\frac{\dfrac{\left\{ (\mathsf{x}, \mathsf{z}, \bar{\mathsf{w}}) \mapsto \mathsf{CX}_{\mathsf{x},\mathsf{z}}|\psi\rangle \right\} C \left\{ Q \right\}}{\left\{ (\mathsf{x}, \mathsf{z}, \bar{\mathsf{w}}) \mapsto \mathsf{CX}_{\mathsf{x},\mathsf{z}}|\psi\rangle * P_\oplus \right\} C \left\{ Q * P_\oplus \right\}} \text{ HOARE-FRAME}}{\left\{ (\mathsf{x}, \mathsf{z}, \bar{\mathsf{w}}) \mapsto |\psi\rangle * \mathsf{y} \mapsto |0\rangle \right\}^{\iota,\kappa,\lambda} \mathsf{mCNOT}^{\iota,\kappa,\lambda}[\mathsf{x}, \mathsf{y}, \mathsf{z}]; C \left\{ Q * P_\oplus \right\}} \text{ HOARE-SEQ}$$

5.4 Error Correction: Bit-Flip Code

Fault tolerance is essential for achieving reliable quantum computation, and error correction lies at its core. From the perspective of program verification, verifying such quantum error-correcting codes (QECCs) is a crucial task that must be addressed. In this subsection and the next § 5.5, we present case studies of verifying two simple but fundamental QECCs using RapunSL, the *bit-flip code* and the *Shor code*, while highlighting how RapunSL achieves our goals (G1, G2, G3).

The *bit-flip code* is the simplest QECC that can correct a single bit-flip error (a.k.a. X error), i.e., a quantum error that flips one qubit flips from $|0\rangle$ to $|1\rangle$ or vice versa. In order to make quantum information tolerant to such errors, we encode a single qubit of quantum information into three actual qubits. Concretely, we encode the state $|0\rangle$ as $|000\rangle$, and $|1\rangle$ as $|111\rangle$. The quantum information we want to protect is called a *logical qubit*, and the three qubits that store the logical qubit are called *physical qubits*. The state of the physical qubits has some robustness, and such robustness enables us to detect and correct the bit-flip error.

The circuit for the bit-flip code error correction procedure is shown in Fig. 21. The left-most column models the occurrence of bit-flip errors, where each if e_i then X applies an X gate to the qubit if the error $e_i \in \{0, 1\}$ occurs. We assume $e_1 + e_2 + e_3 \le 1$ to ensure that at most one qubit is flipped. The procedure of error correction is as follows: we first measure the *syndrome*—the effect caused by the error—and then use the measurement outcomes to recover the logical qubit. After

the recovery procedure, the syndrome affects only the global phase of the logical qubit, and thus we can ignore it.

As we did in the previous example, we can specify and verify the correctness of the bit-flip code error correction procedure BitEC, using the power of RapunSL:

$$\exists P: \text{frameable, prob 1.} \quad \forall \alpha, \beta. \quad \left\{ (x, y, z) \mapsto X^{e_1} X^{e_2} X^{e_3} (\alpha \ket{000} + \beta \ket{111}) \right\}^{a,b}$$
$$\text{BitEC}^{a,b}[x, y, z] \quad \left\{ (x, y, z) \mapsto (\alpha \ket{000} + \beta \ket{111}) * P \right\}.$$

The postcondition $(x, y, z) \mapsto (\alpha \ket{000} + \beta \ket{111}) * P$ describes the state of the system after the error correction procedure. Here, $(x, y, z) \mapsto (\alpha \ket{000} + \beta \ket{111})$ captures the restored logical state of the qubits, while P abstracts the effect of the measured syndrome—that contains the difference of global phase induced by measurement outcomes (G3).

Crucially, the use of separating conjunction $*$ makes the distinction between the logical state and the syndrome information explicit and formal. This separation reflects the fact that the syndrome affects only the global phase or auxiliary context, and not the logical content of the qubits. Such a formulation not only captures the correctness of the procedure but also demonstrates how separation in the logic naturally mirrors separation in the underlying physical system.

We can also think of a *phase-flip error*, applying the Z gate instead of X. A phase-flip error-correction code can be obtained immediately from the bit-flip code. Instead of the Z basis $\langle \ket{0}, \ket{1} \rangle$, the new code uses the X basis $\langle \ket{+}, \ket{-} \rangle$, encoding a qubit $\ket{+}$ as $\ket{+++}$ and $\ket{-}$ as $\ket{---}$. The error correction procedure PhaseEC can be obtained by adding Hadamard H gates before and after BitEC for the basis transformation. Its specification can be naturally derived from that of BitEC.

5.5 Error Correction: Shor's Code

The *Shor code* [Shor 1995] is a QECC that encodes one logical qubit with nine physical qubits and can correct any single-qubit unitary error. In general, if a QECC can correct both a single X error and a single phase-flip error (a.k.a. Z error), it can correct any single-qubit error U, because U can be expressed as a linear combination of id, X, Z, and XZ. Shor's code achieves the correction of both X and Z errors by concatenating a 3-qubit bit-flip code with a 3-qubit phase-flip code: first it encodes three Z-error-tolerant qubits using the 3-qubit phase-flip code, and then applies the 3-qubit bit-flip code to those encoded qubits to obtain a single logical qubit. Let us denote the logical qubit state that encodes the state $\ket{\psi}$ in the phase-flip code as $\ket{\psi_L}$. Concretely, the Shor code encodes the logical qubit state $\alpha \ket{0} + \beta \ket{1}$ into

$$\alpha \ket{0_L 0_L 0_L} + \beta \ket{1_L 1_L 1_L}.$$

As a result, any X error on the logical qubit can be corrected by the top-level bit-flip code, and any Z error can be corrected by the bottom-level phase-flip code.

The error correction procedure of the Shor code is roughly given as follows, where PhaseEC is the Z error-correction procedure defined above and BitEC_L is a variant of the X error-correction procedure BitEC that works on logical qubits (we omit classical variables here for simplicity):

$$\text{ShorEC}[\bar{x}, \bar{y}, \bar{z}] \quad \triangleq \quad \text{PhaseEC}[\bar{x}]; \ \text{PhaseEC}[\bar{y}]; \ \text{PhaseEC}[\bar{z}]; \ \text{BitEC}_L[\bar{x}, \bar{y}, \bar{z}].$$

The Shor code error correction procedure can be proved by combining the specifications of PhaseEC and BitEC_L (G1). In each error correction procedure, we perform syndrome measurement and accumulate some global phase information. However, framing such a syndrome by separating conjunction allows us to abstract the global phase information, enabling scalable reasoning without any blow-up in the size of the formula (G3).

5.6 Probabilistic Choice and Almost Sure Termination

Finally, we present a case study of verifying a quantum program that involves probabilistic choice and almost sure termination (G4).

Since we can use the norm of a quantum state as a probability, we can define the probabilistic choice connective $P \oplus_p Q$ by $\sqrt{p}\, P \oplus \sqrt{1-p}\, Q$, choosing the state P with probability p and Q with probability $1-p$ for $p \in [0, 1]$. This $\oplus_p$ satisfies the following as expected:

$$P \oplus_p Q \dashv\vdash Q \oplus_{1-p} P \qquad\qquad (P \oplus_p Q) \oplus_q R \dashv\vdash P \oplus_{pq} (Q \oplus_{\frac{q}{1-pq}} R)$$

$$\frac{P \vdash Q \qquad P' \vdash Q'}{P \oplus_p P' \vdash Q \oplus_p Q'} \qquad\qquad \frac{\{P\}\, C\, \{Q\} \qquad \{P'\}\, C\, \{Q'\}}{\{P \oplus_p P'\}\, C\, \{Q \oplus_p Q'\}}$$

Also, the probabilistic choice program $C \oplus_p^a C'$ can be defined as follows. First, we introduce a primitive coin_p^a that stores 0 or 1 to a with probability p or $1-p$, respectively.[14] We have

$$\big\{\, \mathrm{emp}\, \big\}^a\ \mathrm{coin}_p^a\ \big\{\, a \mapsto 0 \oplus_p a \mapsto 1\, \big\}.$$

From this, we can derive the probabilistic choice by $C \oplus_p^a C' \triangleq \mathrm{coin}_p^a;\ \mathrm{if}\ a\ \mathrm{then}\ C'\ \mathrm{else}\ C$, for which the following rule can be derived:

$$\frac{\{P\}\, C\, \{Q\} \qquad \{P\}\, C'\, \{R\}}{\big\{\, P\, \big\}^a\ C \oplus_p^a C'\ \big\{\, a \mapsto 0 * Q \oplus_p a \mapsto 1 * R\, \big\}}$$

For example, let us think of the following repeat-until-success program:

$$\mathrm{cointoss}_p\ \triangleq\ \mathrm{coin}_p^a;\ \mathrm{while}\ a\ \mathrm{do}\ \big(\mathrm{coin}_p^a;\ c \leftarrow c + 1\big).$$

Intuitively, the program repeats tossing a coin that returns 0 with probability $p \neq 0$ until the coin finally returns 0. The counter variable c stores the number of iterations. Our logic can prove that this program almost surely terminates. More specifically, we can prove the following, abstracting over an assertion P that is frameable and has the probability 1:

$$\exists\, P: \mathrm{frameable}, \mathrm{prob}\, 1.\ \big\{\, c \mapsto 0\, \big\}^a\ \mathrm{cointoss}_p\ \big\{\, a \mapsto 0 * P\, \big\}.$$

6 Discussion

Automation. One point worth noting about the case studies in the previous section is that most of the proofs follow a uniform three-step pattern: (i) prove the specification on the computational basis (or any chosen basis) via simple symbolic execution; (ii) apply HOARE-SCALE and HOARE-FRAME; and (iii) sum up the derivations with the HOARE-SUM to lift the result to general preconditions.

This observation also suggests a route to automation. Step (i) is essentially a standard symbolic-execution pass augmented with basic complex number arithmetic; steps (ii)–(iii), despite applying rules with intricate side conditions, can still be automated using the frameability notion of Fig. 18, §4.7. This kind of pattern appears in recent work on automated verification of quantum programs, such as AutoQ 2.0 [Chen et al. 2025]; see §7 for the details.

Density matrices. A limitation of such a simple methodology is that the size of the propositions can grow exponentially with the number of measurements. As we explained in §2.4, the abstraction method can help mitigate the growth, or we typically find compact representations (e.g., $\bigoplus_{i=0,1}(-1)^i \cdot \mathsf{x} \mapsto |a \veebar \iota\rangle$) as shown in the case studies considered so far. This method

---

[14] Technically, we do not need to extend our program language, because we can mathematically encode coin_p^a as $\mathsf{M}_p^a[]$, where M_p is the 0-qubit measurement that applies $\sqrt{p}\,\mathrm{id}$ returning 0 or $\sqrt{1-p}\,\mathrm{id}$ returning 1.

should works for any Clifford circuits, but is not guaranteed to work for general programs. In such situations where an explosion occurs, it may be preferable to work with *density matrices* instead.

Although we adopt a global-phase-sensitive logic rather than work with density matrices, this choice does not preclude encoding density matrices in RapunSL. One such encoding is:

$$\mathsf{x} \mapsto^\iota \rho \quad \triangleq \quad \exists I. \ \exists \{ |\psi_i\rangle \}_{i \in I}. \ \ \rho = \sum\nolimits_{i \in I} |\psi_i\rangle\langle\psi_i| \ * \ \bigoplus\nolimits_{i \in I}^\iota \mathsf{x} \mapsto |\psi_i\rangle \, .$$

We can switch between the usual vector representation and this density-matrix representation in RapunSL. Where basis-locality is not necessary, one can use the density matrix representation $\mathsf{x} \mapsto^\iota \rho$. This roughly amounts to the existing approach of Deng et al. [2024] and enjoys entanglement- and outcome-locality. Still, the existential quantification in $\mathsf{x} \mapsto^\iota \rho$ makes the assertion non-precise, which is at odds with some advanced proof rules (e.g., BIGBMIX-UNFRAME).

7 Related Work

Quantum Hoare logics. The pioneering work on Hoare logic [Ying 2012; Zhou et al. 2019; Unruh 2021; Lewis et al. 2023] established a good foundation for deductive verification of quantum programs. In particular, Li and Ying [2017] has proposed a Hoare logic that can specify almost sure termination of quantum programs, as we also do. However, as in the classical case, Hoare logic does not scale for more complex programs, due to its very global style of specification. This problem has been attacked by developing separation logics, as discussed below.

Quantum separation logics. To bring local reasoning to quantum verification, disentanglement has been proposed as a suitable notion of separation, obtaining Quantum Separation Logic (QSL) [Zhou et al. 2021; Le et al. 2022; Deng et al. 2024]. As we remarked in §2, none of the existing QSLs have achieved all three locality principles we proposed (Fig. 1, §1) in one logic.

Of particular note is Deng et al. [2024]'s logic, which is the only existing QSL that supports a probabilistic mixture of assertions ⊕. Unlike our mixing ⊕, modelled as multiset sum, their model of ⊕ is the sum of density matrices, insensitive to global phases and the branching structure introduced by measurements. Consequently, as discussed in §2.2, their approach to ⊕ is incompatible with our basis-locality principle, and a simple extension of their logic cannot achieve that locality.

Also, Su et al. [2024] have recently proposed a separation logic for the modular verification of algorithms that utilize dirty qubits. Their model is global-phase-insensitive, and thus does not support our locality principles. In particular, because of the lack of basis-locality, their framework does not handle cases where dirty qubits are entangled with the input, unlike RapunSL.

Automated verification of quantum programs. Automated verification of quantum programs is a highly important yet challenging research problem, and various approaches have been explored. Qbricks [Chareton et al. 2021] demonstrated some degree of automation by employing the path-sum representation together with Why3 [Filliâtre and Paskevich 2013], a semi-automated verification platform for functional programs. Qafny [Li et al. 2024b] automatically verifies annotated quantum programs by translating them into classical separation logic for arrays implemented in Dafny [Leino and Moskal 2014].[15] Fang and Ying [2024] proposed a symbolic execution framework for quantum programs and applied it for automated verification of quantum error-correction codes.

Particularly relevant to our work is AutoQ 2.0 [Chen et al. 2025]. It is a framework for verifying Hoare-style assertions on quantum programs with measurement, branching, and loop constructs, with automation using level-synchronized tree automata [Abdulla et al. 2025]. Much like our

---

[15] The work by Li et al. [2024b] was not cited in the published POPL version, as we became aware of it only after the version had been finalized.

treatment, it represents post-measurement states as a map of non-normalized vectors and employs reasoning principles similar to RapunSL's outcome- and basis-locality.

Quantum relational logics. Quantum relational logics [Barthe et al. 2019b; Unruh 2019] verify relations between program behaviours rather than functional specifications of single programs. For modularity, relational *separation* logics have been studied in the classical probabilistic setting [Bao et al. 2025], but not yet in the quantum setting.

Outcome logics. Notably, the need to talk about a collection of outcomes at once has emerged before in a completely different context: Outcome Logic [Zilberstein et al. 2023] studied general reasoning principles about branching effects—i.e., non-deterministic and probabilistic (but not quantum) computation—when assertions are over the whole collection of possible outcomes. What we need here is a similar jump, conceptually. Technically, however, there are serious challenges, as Outcome Logic only considers one layer of locality, while we have to handle three, deeply interacting, layers. Outcome Separation Logic [Zilberstein et al. 2024] makes some first steps in incorporating separation of heaps in a language with probabilistic branching, but it uses a very syntactic approach to framing and does not hint at design principles that might apply more generally. In RapunSL, we harmonize general framing with the other two connectives. To make the logic work, in particular, the connective that composes outcomes, mixing, must be non-commutative and satisfy a strong interchange law with sum.

Quantum programming languages. Quantum control has been actively studied in the form of qif and symmetric pattern matching [Sabry et al. 2018], and adopted by several quantum languages [Grattage and Altenkirch 2005; Svore et al. 2018; Bichsel et al. 2020; Hirata and Heunen 2025; Heunen et al. 2026]. Roughly speaking, a program qif e then C_1 else C_0 executes both branches C_0 and C_1 under negative or positive quantum control over the qubit e. We expect that programs with such quantum control can be effectively verified using basis-locality of RapunSL.

Also, in light of the no-cloning theorem, some recent quantum languages [Koch et al. 2025; Hirata and Heunen 2025] adopt ownership types, popularized by Rust [Matsakis and Klock 2014]. We expect that RapunSL can serve as a semantic foundation for such ownership types.

8 Conclusion and Future Work

We proposed RapunSL, the first quantum program logic to unify the three locality principles—entanglement-locality, outcome-locality, and basis-locality. We proved its soundness and demonstrated its power by verifying several practical quantum programs.

Although we have provided a pen-and-paper proof of the soundness of our logic, it would be better to mechanize the proof, as has been done for some quantum program logics [Zhou et al. 2023; Wu et al. 2025]. It would help explore advanced features without the fear of making mistakes.

One important direction for future work is the application of RapunSL for automated verification. We have started exploring this direction, as discussed in § 6. More specifically, we expect that the insights from RapunSL can be used to extend AutoQ 2.0 [Chen et al. 2025], discussed in § 7, with the entanglement-locality principle for more scalable automation.

We are also interested in extending our logic to relational verification. A possible direction for future work is to adopt the perspective of Bluebell [Bao et al. 2025], which unified unary and relational reasoning in a single separation logic for probabilistic programs.

Acknowledgments

References

Parosh Aziz Abdulla, Yo-Ga Chen, Yu-Fang Chen, Lukáš Holík, Ondřej Lengál, Jyun-Ao Lin, Fang-Yi Lo, and Wei-Lun Tsai. 2025. Verifying Quantum Circuits with Level-Synchronized Tree Automata. *Proc. ACM Program. Lang.* 9, POPL, Article 32 (Jan. 2025), 31 pages. https://doi.org/10.1145/3704868

Jialu Bao, Simon Docherty, Justin Hsu, and Alexandra Silva. 2021. A Bunched Logic for Conditional Independence. In *Proceedings of the 36th Annual ACM/IEEE Symposium on Logic in Computer Science* (Rome, Italy) *(LICS '21)*. IEEE Press, Article 13, 14 pages. https://doi.org/10.1109/LICS52264.2021.9470712

Jialu Bao, Emanuele D'Osualdo, and Azadeh Farzan. 2025. Bluebell: An Alliance of Relational Lifting and Independence for Probabilistic Reasoning. *Proc. ACM Program. Lang.* 9, POPL, Article 58 (Jan. 2025), 31 pages. https://doi.org/10.1145/3704894

Gilles Barthe, Justin Hsu, and Kevin Liao. 2019a. A Probabilistic Separation Logic. *Proc. ACM Program. Lang.* 4, POPL, Article 55 (Dec. 2019), 30 pages. https://doi.org/10.1145/3371123

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

Benjamin Bichsel, Maximilian Baader, Timon Gehr, and Martin Vechev. 2020. Silq: A High-Level Quantum Language with Safe Uncomputation and Intuitive Semantics. In *Proceedings of the 41st ACM SIGPLAN Conference on Programming Language Design and Implementation* (London, UK) *(PLDI 2020)*. Association for Computing Machinery, New York, NY, USA, 286–300. https://doi.org/10.1145/3385412.3386007

Stephen Brookes. 2007. A Semantics for Concurrent Separation Logic. *Theor. Comput. Sci.* 375, 1–3 (April 2007), 227–270. https://doi.org/10.1016/j.tcs.2006.12.034

Stephen Brookes and Peter W. O'Hearn. 2016. Concurrent Separation Logic. *ACM SIGLOG News* 3, 3 (Aug. 2016), 47–65. https://doi.org/10.1145/2984450.2984457

Christophe Chareton, Sébastien Bardin, François Bobot, Valentin Perrelle, and Benoît Valiron. 2021. An Automated Deductive Verification Framework for Circuit-building Quantum Programs. In *Programming Languages and Systems: 30th European Symposium on Programming, ESOP 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 – April 1, 2021, Proceedings* (Luxembourg City, Luxembourg). Springer-Verlag, Berlin, Heidelberg, 148–177. https://doi.org/10.1007/978-3-030-72019-3_6

Yu-Fang Chen, Kai-Min Chung, Min-Hsiu Hsieh, Wei-Jia Huang, Ondřej Lengál, Jyun-Ao Lin, and Wei-Lun Tsai. 2025. AutoQ 2.0: From Verification of Quantum Circuits to Verification of Quantum Programs. In *Tools and Algorithms for the Construction and Analysis of Systems: 31st International Conference, TACAS 2025, Held as Part of the International Joint Conferences on Theory and Practice of Software, ETAPS 2025, Hamilton, ON, Canada, May 3–8, 2025, Proceedings, Part III* (Hamilton, ON, Canada). Springer-Verlag, Berlin, Heidelberg, 87–108. https://doi.org/10.1007/978-3-031-90660-2_5

Yuxin Deng, Huiling Wu, and Ming Xu. 2024. Local Reasoning About Probabilistic Behaviour for Classical–Quantum Programs. In *Verification, Model Checking, and Abstract Interpretation: 25th International Conference, VMCAI 2024, London, United Kingdom, January 15–16, 2024, Proceedings, Part II* (London, United Kingdom). Springer-Verlag, Berlin, Heidelberg, 163–184. https://doi.org/10.1007/978-3-031-50521-8_8

Eric Dennis, Alexei Kitaev, Andrew Landahl, and John Preskill. 2002. Topological quantum memory. *J. Math. Phys.* 43, 9 (09 2002), 4452–4505. https://doi.org/10.1063/1.1499754

Wang Fang and Mingsheng Ying. 2024. Symbolic Execution for Quantum Error Correction Programs. *Proc. ACM Program. Lang.* 8, PLDI, Article 189 (June 2024), 26 pages. https://doi.org/10.1145/3656419

Jean-Christophe Filliâtre and Andrei Paskevich. 2013. Why3 — Where Programs Meet Provers. In *Proceedings of the 22nd European Conference on Programming Languages and Systems* (Rome, Italy) *(ESOP'13)*. Springer-Verlag, Berlin, Heidelberg, 125–128. https://doi.org/10.1007/978-3-642-37036-6_8

Austin G. Fowler and Craig Gidney. 2019. Low Overhead Quantum Computation Using Lattice Surgery. *CoRR* (Aug. 2019). arXiv:1808.06709 [quant-ph] https://arxiv.org/abs/1808.06709

Craig Gidney. 2018. Halving the Cost of Quantum Addition. *Quantum* 2 (June 2018), 74. https://doi.org/10.22331/q-2018-06-18-74

Jonathan Grattage and Thorsten Altenkirch. 2005. A Functional Quantum Programming Language. In *Proceedings. 20th Annual IEEE Symposium on Logic in Computer Science*. IEEE Computer Society, Los Alamitos, CA, USA, 249–258. https://doi.org/10.1109/LICS.2005.1

Thomas Häner, Martin Roetteler, and Krysta M. Svore. 2017. Factoring Using $2n + 2$ Qubits with Toffoli Based Modular Multiplication. *Quantum Info. Comput.* 17, 7-8 (June 2017), 673–684.

Chris Heunen, Louis Lemonnier, Christopher McNally, and Alex Rice. 2026. Quantum circuits are just a phase. *Proc. ACM Program. Lang.* 10, POPL, Article 89 (Jan. 2026). https://doi.org/10.1145/3776731

Kengo Hirata and Chris Heunen. 2025. Qurts: Automatic Quantum Uncomputation by Affine Types with Lifetime. *Proc. ACM Program. Lang.* 9, POPL, Article 6 (Jan. 2025), 28 pages. https://doi.org/10.1145/3704842

Dominic Horsman, Austin G. Fowler, Simon Devitt, and Rodney Van Meter. 2012. Surface code quantum computing by lattice surgery. *New Journal of Physics* 14, 12 (dec 2012), 123011. https://doi.org/10.1088/1367-2630/14/12/123011

Samin S. Ishtiaq and Peter W. O'Hearn. 2001. BI as an Assertion Language for Mutable Data Structures. In *Proceedings of the 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages* (London, United Kingdom) *(POPL '01)*. Association for Computing Machinery, New York, NY, USA, 14–26. https://doi.org/10.1145/360204.375719

Mark Koch, Agustín Borgna, Craig Roy, Alan Lawrence, Kartik Singhal, Seyon Sivarajah, and Ross Duncan. 2025. Imperative Quantum Programming with Ownership and Borrowing in Guppy. *CoRR* (Oct. 2025). arXiv:2510.13082 [cs.PL] https://arxiv.org/abs/2510.13082

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

K. Rustan M. Leino and Michał Moskal. 2014. Co-induction Simply. In *FM 2014: Formal Methods*, Cliff Jones, Pekka Pihlajasaari, and Jun Sun (Eds.). Springer International Publishing, Cham, 382–398.

Marco Lewis, Sadegh Soudjani, and Paolo Zuliani. 2023. Formal Verification of Quantum Programs: Theory, Tools, and Challenges. *ACM Transactions on Quantum Computing* 5, 1, Article 1 (Dec. 2023), 35 pages. https://doi.org/10.1145/3624483

John M. Li, Amal Ahmed, and Steven Holtzen. 2023. Lilac: A Modal Separation Logic for Conditional Probability. *Proc. ACM Program. Lang.* 7, PLDI, Article 112 (June 2023), 24 pages. https://doi.org/10.1145/3591226

John M. Li, Jon Aytac, Philip Johnson-Freyd, Amal Ahmed, and Steven Holtzen. 2024a. A Nominal Approach to Probabilistic Separation Logic. In *Proceedings of the 39th Annual ACM/IEEE Symposium on Logic in Computer Science* (Tallinn, Estonia) *(LICS '24)*. Association for Computing Machinery, New York, NY, USA, Article 55, 14 pages. https://doi.org/10.1145/3661814.3662135

Liyi Li, Mingwei Zhu, Rance Cleaveland, Alexander Nicolellis, Yi Lee, Le Chang, and Xiaodi Wu. 2024b. Qafny: A Quantum-Program Verifier. In *38th European Conference on Object-Oriented Programming, ECOOP 2024, Vienna, Austria, September 16-20, 2024 (LIPIcs, Vol. 313)*, Jonathan Aldrich and Guido Salvaneschi (Eds.). Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 24:1–24:31. https://doi.org/10.4230/LIPICS.ECOOP.2024.24

Yangjia Li and Mingsheng Ying. 2017. Algorithmic Analysis of Termination Problems for Quantum Programs. *Proc. ACM Program. Lang.* 2, POPL, Article 35 (Dec. 2017), 29 pages. https://doi.org/10.1145/3158123

Nicholas D. Matsakis and Felix S. Klock, II. 2014. The Rust Language. In *Proceedings of the 2014 ACM SIGAda annual conference on High integrity language technology* (Portland, Oregon, USA) *(HILT '14)*. Association for Computing Machinery, New York, NY, USA, 103–104. https://doi.org/10.1145/2663171.2663188

Junhong Nie, Wei Zi, and Xiaoming Sun. 2024. Quantum Circuit for Multi-Qubit Toffoli Gate with Optimal Resource. *CoRR* (Feb. 2024). arXiv:2402.05053 [quant-ph] https://arxiv.org/abs/2402.05053

Peter W. O'Hearn. 2007. Resources, Concurrency and Local Reasoning. *Theor. Comput. Sci.* 375, 1–3 (April 2007), 271–307. https://doi.org/10.1016/j.tcs.2006.12.035

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

Peter W. O'Hearn, John C. Reynolds, and Hongseok Yang. 2001. Local Reasoning about Programs that Alter Data Structures. In *Proceedings of the 15th International Workshop on Computer Science Logic (CSL '01)*. Springer-Verlag, Berlin, Heidelberg, 1–19.

John C. Reynolds. 2002. Separation Logic: A Logic for Shared Mutable Data Structures. In *Proceedings of the 17th Annual IEEE Symposium on Logic in Computer Science (LICS '02)*. IEEE Computer Society, USA, 55–74.

Amr Sabry, Benoît Valiron, and Juliana Kaizer Vizzotto. 2018. From Symmetric Pattern-Matching to Quantum Control. In *Foundations of Software Science and Computation Structures*, Christel Baier and Ugo Dal Lago (Eds.). Springer International Publishing, Cham, 348–364.

Peter W. Shor. 1995. Scheme for Reducing Decoherence in Quantum Computer Memory. *Physical Review A* 52, 4 (Oct. 1995), R2493–R2496. https://doi.org/10.1103/PhysRevA.52.R2493

Bonan Su, Li Zhou, Yuan Feng, and Mingsheng Ying. 2024. BI-based Reasoning about Quantum Programs with Heap Manipulations. *CoRR* (Sept. 2024). arXiv:2409.10153 [quant-ph] https://arxiv.org/abs/2409.10153

Krysta Svore, Alan Geller, Matthias Troyer, John Azariah, Christopher Granade, Bettina Heim, Vadym Kliuchnikov, Mariia Mykhailova, Andres Paz, and Martin Roetteler. 2018. Q#: Enabling Scalable Quantum Computing and Development with a High-level DSL. In *Proceedings of the Real World Domain Specific Languages Workshop 2018* (Vienna, Austria) *(RWDSL2018)*. Association for Computing Machinery, New York, NY, USA, Article 7, 10 pages. https://doi.org/10.1145/3183895.3183901

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

Dominique Unruh. 2021. Quantum Hoare Logic with Ghost Variables. In *Proceedings of the 34th Annual ACM/IEEE Symposium on Logic in Computer Science* (Vancouver, Canada) *(LICS '19)*. IEEE Press, Article 47, 13 pages.

Huiling Wu, Yuxin Deng, and Ming Xu. 2025. Local Reasoning about Probabilistic Behaviour for Classical-Quantum Programs. *CoRR* (Feb. 2025). arXiv:2308.04741 [cs.PL] https://arxiv.org/abs/2308.04741

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

Li Zhou, Gilles Barthe, Justin Hsu, Mingsheng Ying, and Nengkun Yu. 2021. A Quantum Interpretation of Bunched Logic & Quantum Separation Logic. In *Proceedings of the 36th Annual ACM/IEEE Symposium on Logic in Computer Science* (Rome, Italy) *(LICS '21)*. IEEE Press, Article 75, 14 pages. https://doi.org/10.1109/LICS52264.2021.9470673

Li Zhou, Gilles Barthe, Pierre-Yves Strub, Junyi Liu, and Mingsheng Ying. 2023. CoqQ: Foundational Verification of Quantum Programs. *Proc. ACM Program. Lang.* 7, POPL, Article 29 (Jan. 2023), 33 pages. https://doi.org/10.1145/3571222

Li Zhou, Nengkun Yu, and Mingsheng Ying. 2019. An Applied Quantum Hoare Logic. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation* (Phoenix, AZ, USA) *(PLDI 2019)*. Association for Computing Machinery, New York, NY, USA, 1149–1162. https://doi.org/10.1145/3314221.3314584

Noam Zilberstein, Derek Dreyer, and Alexandra Silva. 2023. Outcome Logic: A Unifying Foundation for Correctness and Incorrectness Reasoning. *Proc. ACM Program. Lang.* 7, OOPSLA1, Article 93 (April 2023), 29 pages. https://doi.org/10.1145/3586045

Noam Zilberstein, Angelina Saliling, and Alexandra Silva. 2024. Outcome Separation Logic: Local Reasoning for Correctness and Incorrectness with Computational Effects. *Proc. ACM Program. Lang.* 8, OOPSLA1, Article 104 (April 2024), 29 pages. https://doi.org/10.1145/3649821

Appendix

A More on the Logic

A.1 Standard Connectives

We have the following standard connectives. They satisfy the usual proof rules, presented below:

$$\forall x \in A. P_x \triangleq \bigcap_{x \in A} P_x \qquad \exists x \in A. P_x \triangleq \bigcup_{x \in A} P_x$$

$$P_0 \wedge P_1 \triangleq \forall i \in \{0, 1\}. P_i \qquad P_0 \vee P_1 \triangleq \exists i \in \{0, 1\}. P_i$$

$$\neg P \triangleq P^\complement \qquad \ulcorner \varphi \urcorner \triangleq \{ m \mid \varphi \}$$

$$\frac{\forall x \in A. (P \vdash Q_x)}{P \vdash \forall x \in A. Q_x} \qquad \frac{\forall x \in A. (P_x \vdash Q)}{(\exists x \in A. P_x) \vdash Q}$$

$$P \wedge \neg P \vdash Q \qquad Q \vdash P \vee \neg P \qquad \frac{\varphi \rightarrow (P \vdash Q)}{\ulcorner \varphi \urcorner \wedge P \vdash Q}$$

A.2 Right Adjoint of SL Connectives

In Remark 7, we mentioned that the connectives $\wedge$, $*$, $\oplus$ and $+$ all have the right adjoint, written as $\rightarrow$, $-\!*$, $-\!\oplus$ and $-\!+$.

The following give the models for these connectives:[16]

$$P \rightarrow Q \triangleq P^\complement \cup Q \qquad P -\!* Q \triangleq \{ m \mid \forall m' \in P \text{ s.t. } m \cdot m' \downarrow. \ m \cdot m' \in Q \}$$

$$P -\!\oplus Q \triangleq \{ m \mid \forall m' \in P. \ m \uplus m' \in Q \}$$

$$P -\!+ Q \triangleq \{ m \mid \forall m' \in P. \ \forall r : m \cong_{\mathsf{m}} m' \text{ s.t. } m +_r m' \downarrow. \ m +_r m' \in Q \}$$

As expected, we have the following adjunction rules.

$$\frac{P \wedge Q \vdash R}{P \vdash Q \rightarrow R} \qquad \frac{P * Q \vdash R}{P \vdash Q -\!* R} \qquad \frac{P \oplus Q \vdash R}{P \vdash Q -\!\oplus R} \qquad \frac{P + Q \vdash R}{P \vdash Q -\!+ R}$$

A.3 Additional Proof Rules

We have the following proof rules for the quantum and classical points-to tokens.

QPOINTS-SWAP
$$(\bar{z}, x, y, \bar{z}') \mapsto |\psi\rangle \ \dashv\vdash\ (\bar{z}, y, x, \bar{z}') \mapsto \mathsf{SWAP}_{x,y} |\psi\rangle$$

QPOINTS-DISTINCT
$$\bar{x} \mapsto |\psi\rangle \vdash \mathrm{distinct}(\bar{x})$$

CPOINTS-DISJ
$$a \mapsto v * b \mapsto w \vdash a \neq b$$

The rule QPOINTS-SWAP is for permuting the qubits of a quantum points-to token. Here, SWAP is the unitary operator that maps $|ij\rangle$ to $|ji\rangle$ for each $i, j \in \{0, 1\}$.

We have the following proof rules for $P : \mathsf{nonnb}$, defined in Fig. 16.

$$\mathsf{emp}, \bar{x} \mapsto |\psi\rangle, a \mapsto v : \mathsf{nonnb} \qquad \frac{Q : \mathsf{nonnb} \quad P \vdash Q}{P : \mathsf{nonnb}}$$

$$\frac{\forall x \in I. (P_x : \mathsf{nonnb}) \quad I \neq \varnothing}{\bigoplus_{x \in I} P_x : \mathsf{nonnb}} \qquad \frac{P, Q : \mathsf{nonnb}}{P * Q : \mathsf{nonnb}} \qquad \frac{P : \mathsf{nonnb}}{P + Q : \mathsf{nonnb}}$$

---

[16] We use $\downarrow$ to denote definedness (e.g., $m \cdot m' \downarrow$ means that $m \cdot m'$ is defined).

We have the following proof rules for P : prob p, defined in Fig. 19.

$$\mathsf{emp}, a \mapsto v : \mathsf{prob}\, 1 \qquad \bar{\mathsf{x}} \mapsto |\psi\rangle : \mathsf{prob}\, \||\psi\rangle\|^2 \qquad \frac{Q : \mathsf{prob}\, p \qquad P \vdash Q}{P : \mathsf{prob}\, p}$$

$$\frac{\forall x.\, (P_x : \mathsf{prob}\, p_x)}{\bigoplus_{x \in I} P_x : \mathsf{prob}\, (\sum_{x \in I} p_x)} \qquad \frac{P : \mathsf{prob}\, p \qquad Q : \mathsf{prob}\, q}{P * Q : \mathsf{prob}\, pq}$$

A.4 More on Tagged Mixing

We have the following derived rules for tagged mixing, other than those in Fig. 17:

BIGMIX-MONO
$$\frac{\forall x \in I.\, (P_x \vdash Q_x)}{\bigoplus_{x \in I}^{\iota} P_x \vdash \bigoplus_{x \in I}^{\iota} Q_x}$$

BIGMIX-COMM
$$\frac{f : I \to J \text{ is a bijection}}{\bigoplus_{x \in I}^{\iota} P_{fx} \dashv\vdash \bigoplus_{y \in J}^{\iota} P_y}$$

BIGMIX-BIGMIX
$$\bigoplus_{x \in I}^{\iota} \bigoplus_{y \in J}^{\kappa} P_{x,y} \dashv\vdash \bigoplus_{y \in J}^{\kappa} \bigoplus_{x \in I}^{\iota} P_{x,y}$$

NB-BIGMIX
$$\mathsf{nb} \dashv\vdash \bigoplus_{\_ \in \varnothing}^{\iota}$$

MIX-BIGMIX
$$P_0\, {}_0\oplus_1^{\iota}\, P_1 \dashv\vdash \bigoplus_{i \in \{0,1\}}^{\iota} P_i$$

BIGMIX-SCALE
$$\alpha\, (\bigoplus_{x \in I}^{\iota} P_x) \dashv\vdash \bigoplus_{x \in I}^{\iota} \alpha\, P_x$$

$$\frac{\forall x.\, (P_x : \mathsf{precise})}{\bigoplus_{x \in I}^{\iota} P_x : \mathsf{precise}}$$

HOARE-BIGMIX
$$\frac{\forall x \in I.\, \{P_x\}\, C\, \{Q_x\}}{\{\bigoplus_{x \in I}^{\iota} P_x\}\, C\, \{\bigoplus_{x \in I}^{\iota} Q_x\}}$$

BIGMIX-FRAME
$$(\bigoplus_{x \in I}^{\iota} P_x) * Q \vdash \bigoplus_{x \in I}^{\iota} (P_x * Q)$$

BIGMIX-UNFRAME
$$\frac{Q : \mathsf{precise}}{\bigoplus_{x \in I}^{\iota} (P_x * Q) \vdash (\bigoplus_{x \in I}^{\iota} P_x) * Q}$$

BIGMIX-FRAME-FRAMEABLE
$$\frac{Q : \mathsf{frameable}}{\bigoplus_{x \in I}^{\iota} (P_x * Q) \dashv\vdash (\bigoplus_{x \in I}^{\iota} P_x) * Q}$$

$$\frac{\forall x \in I.\, (P_x : \mathsf{nonnb}) \quad I \neq \varnothing}{\bigoplus_{x \in I}^{\iota} P_x : \mathsf{nonnb}}$$

$$\frac{\forall x.\, (P_x : \mathsf{prob}\, p_x)}{\bigoplus_{x \in I}^{\iota} P_x : \mathsf{prob}\, (\sum_{x \in I} p_x)}$$

A.5 More on Incompatibility Relation

In Fig. 16, we defined the incompatibility relation of resources by

$$(|\psi\rangle, S) \,\#\, (|\phi\rangle, S') \;\triangleq\; \exists a \in \mathrm{dom}\, S \cap \mathrm{dom}\, S'.\, S[a] \neq S'[a].$$

We state the following property that characterizes it algebraically.

Lemma 9 (Algebraic characterization of incompatibility). *For any x, y in the resource ring $R = \mathit{Res}$, the following three conditions are equivalent:*

(1) $x \,\#\, y$,

(2) $\forall k, h \in R.\, k \cdot x + h \cdot y = \bot$,

(3) $\forall z \in R.\, (\forall k \in R.\, k \cdot x + z = \bot) \lor (\forall h \in R.\, h \cdot y + z = \bot)$.

PROOF. For simplicity, we denote $x[a]$ to mean $S[a]$ for $x = (|\psi\rangle, S)$.

(1) $\Rightarrow$ (2). If there exists a such that $x[a] \neq y[a]$ and both $k \cdot x$ and $h \cdot y$ are defined, then $(k \cdot x)[a] = x[a] \neq y[a] = (h \cdot y)[a]$. So $k \cdot x + h \cdot y$ is undefined, that is, it equals $\bot$.

(2) $\Rightarrow$ (3). In Res, if $x + y \neq \bot$, then $x + z = \bot$ if and only if $y + z = \bot$ for any z. Therefore, if $k \cdot x + z \neq \bot$ and $k \cdot x + h \cdot y = \bot$, then $z + h \cdot y = \bot$.

(3) $\Rightarrow$ (1). We prove the contrapositive. Let $x = (\mathsf{X} \mapsto |\psi\rangle, S)$ and $y = (\mathsf{Y} \mapsto |\phi\rangle, S')$. Since there is no $a \in \mathit{Store}$ such that $S[a] \neq S'[a]$, we can take the supremum $S'' \triangleq \sup(S, S')$. Let us define

$Z \triangleq X \cup Y$ and $z \triangleq (Z \mapsto 0, S'')$. Such z satisfies $z = z + z = x \cdot (Z \setminus X \mapsto 0, S'' \setminus S) = y \cdot (Z \setminus Y \mapsto 0, S'' \setminus S')$ thus it contradicts (3). □

Lemma 10 (Frame on incompatibility). *If $x, y \in R$ satisfy $x \# y$, then $x + y = \bot$. In particular, $x \neq y$. Also, for any $z \in R$, $x \cdot z \# y$ and $x + z \# y$ if the left-hand side is defined.*

PROOF. Immediate from (2). □

Remark 11. In this paper, we defined the incompatibility relation only for the specific resource ring *Res*. This definition extends to any resource ring R by adopting (2) $\wedge$ (3) as axioms. These conditions (2) and (3) are not equivalent in general, so both are required. Lemma 10 holds for arbitrary R, and all soundness proof in this subsection can be done with this general definition. We note that the condition (2) captures the essence of incompatibility, whereas (3) is required for the soundness of SUM-PRECISE and SUM-UNFRAME.

A.6 Soundness Proof

We give proofs of the correctness of some non-trivial proof rules.

PROOF OF BIGBMIX-FRAME, FIG. 10. All elements on the left-hand side are of the form $(\biguplus_x m_x) \cdot m'$ where $m_x \in P_x$ and $m' \in Q$. By an easy calculation, this is equivalent to $\biguplus_x m_1 \cdot m'$, which is on the right-hand side.[17] □

PROOF OF BIGBMIX-UNFRAME, FIG. 10. If Q is empty, then both sides are empty. If Q is a singleton $\{m\}$, then the left-hand side is $\left\{ \biguplus_x m_x \cdot m \mid m_x \in P_x \right\}$. Since $\biguplus_x m_x \cdot m = (\biguplus_x m_x) \cdot m$, this is included in the right-hand side. □

PROOF OF SUM-ASSOC, FIG. 15. All elements on the left-hand side are of the form $(m_1 +_r m_2) +_s m_3$ where $m_1 \in P$, $m_2 \in Q$, $m_3 \in R$, with multiset bijections $r : m_1 \cong_{\mathrm{m}} m_2$ and $s : (m_1 + m_2) \cong_{\mathrm{m}} m_3$. Since multiset bijections compose and there are canonical multiset bijections $m \cong_{\mathrm{m}} m' \cong_{\mathrm{m}} m +_h m'$ when $m +_h m'$ is defined, we have a canonical multiset bijection $m_1 \cong_{\mathrm{m}} m_2 \cong_{\mathrm{m}} m_3$. Along these bijections, we can define $m_1 +_{r'} (m_2 +_{s'} m_3)$ with appropriate r' and s', satisfying $m_1 +_{r'} (m_2 +_{s'} m_3) = (m_1 +_r m_2) +_s m_3$. □

PROOF OF SUM-FRAME, FIG. 15. All elements on the left-hand side are of the form $(m_1 +_r m_2) \cdot m'$ where $m_1 \in P$, $m_2 \in Q$, and $m' \in R$, with a multiset bijection $r : m_1 \cong_{\mathrm{m}} m_2$. Then,

$$
\begin{aligned}
(m_1 +_r m_2) \cdot m' &= \{\!| (x + x') \cdot y \mid (x, x') \in_{\mathrm{m}} r, y \in_{\mathrm{m}} m' |\!\} \\
&= \{\!| (x \cdot y) + (x' \cdot y) \mid (x, x') \in_{\mathrm{m}} r, y \in_{\mathrm{m}} m' |\!\} \\
&= (m_1 \cdot m') +_{r'} (m_2 \cdot m') \quad \in \mathrm{RHS},
\end{aligned}
$$

where r' is the extended multiset bijection of r that relates $x \cdot y$ and $x' \cdot y$ for each $(x, x') \in_{\mathrm{m}} r$.[18] □

PROOF OF SUM-BIGBMIX, FIG. 15. All elements on the left-hand side are of the form $\biguplus_x m_x +_{r_x} m'_x$ where $m_x \in P_x$ and $m'_x \in Q_x$, with multiset bijections $r_x : m_x \cong_{\mathrm{m}} m'_x$. Let $\biguplus_x r_x : (\biguplus_x m_x) \cong_{\mathrm{m}} (\biguplus_x m'_x)$ be the multiset bijection that is the union of all the r_x, i.e., if $m_x, m'_x, r_x : J_x \to \textit{Res}$,

$$
\left(\biguplus_x r_x \right) : \bigsqcup_{x \in I} J_x \longrightarrow \textit{Res}; \quad (x \in I, y \in J_x) \longmapsto (a, b) = r_x(y).
$$

---

[17] The opposite direction is not generally true when Q is not a singleton set.

[18] The opposite direction is not generally true because not all the multiset bijections between $m_1 \cdot m'$ and $m_2 \cdot m'$ are of the form r'.

Then, we have $\biguplus_x m_x +_{r_x} m'_x = \{\!| a + b \mid x \in I, (a, b) \in_\mathsf{m} r_x |\!\} = \{\!| a + b \mid (a, b) \in_\mathsf{m} \biguplus_x r_x |\!\} \in$ RHS.[19] $\square$

PROOF OF SUM-PRECISE, FIG. 16. Let $P = \{m\}$ and $Q = \{m'\}$. Assume that we have $r, s \colon m \cong_\mathsf{m} m'$ such that $m +_r m'$ and $m +_s m'$. If $(a, b) \in_\mathsf{m} r$ and $(a', b) \in_\mathsf{m} s$, then $a + b \downarrow$ and $a' + b \downarrow$ hold. From (3) in Lemma 9, $\neg(a \# a')$. Since m is unambiguous, $a = a'$. Therefore, $r = s$. $\square$

PROOF OF BIGMIX-SUM, FIG. 16. All elements on the left-hand side are of the form $(\biguplus_{x \in I} m_x) +_r (\biguplus_{x \in I} m'_x)$ where $m_x \in P_x$, $m'_x \in Q_x$, and $r \colon (\biguplus_x m_x) \cong_\mathsf{m} (\biguplus_x m'_x)$. Let $m \triangleq \biguplus_x m_x$ and $m' \triangleq \biguplus_x m'_x$. This r can be regarded as a bijection of the set of indices $\bigsqcup_{x \in I} J_x$ and $\bigsqcup_{x \in I} J'_x$ where $m_x \colon J_x \to Res$ and $m'_x \colon J'_x \to Res$. For each $(x, j) \in \bigsqcup_{x \in I} J_x$, let $(y, j') \triangleq r(x, j)$. Since the sum $m(x, j) + m'(r(x, j)) = m_x(j) + m'_y(j')$ is defined, $x = y$ because of the assumption $x \neq y \Rightarrow P_x \# Q_y$ and Lemma 10. Therefore, for each $i \in I$, the image of r restricted to J_i is included in J'_i. The bijection r must be of the form $\biguplus_{x \in I} r_x \colon \biguplus_{x \in I} m_x \to \biguplus_{x \in I} m'_x$ with some multiset bijections $r_x \colon m_x \cong_\mathsf{m} m'_x$. Therefore, $(\biguplus_x m_x) +_r (\biguplus_x m'_x)$ can now be rewritten as $\biguplus_x m_x +_{r_x} m'_x$, which is in the right-hand side. $\square$

PROOF OF SUM-UNFRAME, FIG. 16. If R is empty, then both sides are empty. Let R be $\{m\}$. All elements on the left-hand side are of the form $(m_1 \cdot m) +_r (m_2 \cdot m)$ where $m_1 \in P$, $m_2 \in Q$, and $r \colon m_1 \cdot m \cong_\mathsf{m} m_2 \cdot m$. Because of the unambiguity of P and R, all the multiplicities of m_1 and m are either 0 or 1, so the multisets m_1 and m can be regarded as mere sets of resources. The multiset bijection r can now be regarded as a bijection of the sets $m_1 \times m$ and $I \times m$, where I is the set of indices of $m_2 \colon I \to Res$. For each $x \in m_1$ and $z \in m$, if $(i, z') \triangleq r(x, z)$, then $x \cdot z + m_2(i) \cdot z' \neq \bot$ follows because $(m_1 \cdot m) +_r (m_2 \cdot m) \downarrow$. Because of the unambiguity of R, $z = z'$ follows from (2) in Lemma 9. Therefore, for each $z \in m$, there is a bijection $r_z \triangleq \mathsf{fst} \circ r(-, z) \colon m_1 \to I$ that satisfies $r(x, z) = (r_z(x), z)$. We prove that this bijection r_z does not depend on z. Since m is non-empty because of the assumption $R \colon$ nonnb, we choose one $z_0 \in m$. For each $z \in m$ and $i = r_{z_0}(x) = r_z(x')$, both $x \cdot z + m_2(i) \cdot z$ and $x' \cdot z + m_2(i) \cdot z$ are defined. Therefore, from (3) in Lemma 9, $\neg(x \# x')$. Because P is unambiguous, x and x' must be equal, which proves $r_z^{-1} \circ r_{z_0} = \mathrm{id}_{m_1} (\Leftrightarrow r_{z_0} = r_z)$. We can now rewrite $(m_1 \cdot m) +_r (m_2 \cdot m)$ as

$$
\begin{aligned}
(m_1 \cdot m) +_r (m_2 \cdot m) &= \{\!| x \cdot z + y \cdot z' \mid ((x, z), (y, z')) \in_\mathsf{m} r |\!\} \\
&= \{\!| x \cdot z + m_2(i) \cdot z' \mid x \in m_1, z \in m, (i, z') = r(x, z) |\!\} \\
&= \{\!| (x + m_2(r_{z_0}(x))) \cdot z \mid x \in m_1, z \in m |\!\} \\
&= (m_1 +_{r_{z_0}} m_2) \cdot m \qquad \in \text{RHS}. \qquad\qquad \square
\end{aligned}
$$

PROOF OF THE REST OF THE RULES IN FIG. 16. Most rules are straightforward from the definition and Lemma 10. We prove $P, Q \colon$ unambig $\Rightarrow P * Q \colon$ unambig as an example. Let $m \cdot m' \in P * Q$ where $m \in P$ and $m' \in Q$, and $\{\!| x \cdot x', y \cdot y' |\!\} \subseteq_\mathsf{m} m \cdot m'$ where $x, y \in_\mathsf{m} m$ and $x', y' \in_\mathsf{m} m'$. The indices of $x \cdot x'$ and $y \cdot y'$ are not the same, thus $\{\!| x, y |\!\} \subseteq_\mathsf{m} m$ or $\{\!| x', y' |\!\} \subseteq_\mathsf{m} m'$ from the definition of $m \cdot m'$. Without loss of generality, we assume $\{\!| x, y |\!\} \subseteq_\mathsf{m} m$. Then, from Lemma 10, $x \cdot x' \# y$ and $x \cdot x' \# y \cdot y'$. $\square$

The remaining rules to be shown are the ones regarding Hoare triples. From here on, we denote $c \Rightarrow t$ to mean $c = (\bar{C}, |\psi\rangle, S)$ and $t = [\![\bar{C}]\!](|\psi\rangle, S)$.

---

[19] The opposite direction is not generally true because not all the multiset bijections between $\biguplus_x m_x$ and $\biguplus_x m'_x$ are of the form $\biguplus_x r_x$.

Lemma 12 (Frame on denotations). *Let $c = (\bar{C}, a)$ be a configuration and t be a tree such that $c \Rightarrow t$. If $x \in Res$ satisfies $(\bar{C}, a \cdot_{Res} x) \Rightarrow t'$, then the tree t' is obtained from t by replacing all the leaves* $\mathsf{Leaf}(b)$ *with* $\mathsf{Leaf}(b \cdot_{Res} x)$.

PROOF. By induction on the definition of the denotational semantics. Note that no commands change the domain of resources, i.e., if c' appears as a leaf of $[\![\bar{C}]\!](c)$ where $c = (\mathsf{X} \mapsto |\psi\rangle, S)$ and $c' = (\mathsf{Y} \mapsto |\psi\rangle, S')$, then $\mathsf{X} = \mathsf{Y}$ and $\mathrm{dom}(S) = \mathrm{dom}(S')$. □

PROOF OF HOARE-FRAME, FIG. 13. Follows from Lemma 12. □

PROOF OF HOARE-BIGBMIX, FIG. 13. Straightforward from the definition of weakest precondition. □

Lemma 13 (Adding denotations). *Let c_0 and c_1 be two configurations such that $c_i = (\bar{C}, \mathsf{X} \mapsto |\psi_i\rangle, S)$. If $c_i \Rightarrow t_i$ holds for $i \in \{0, 1\}$, then $c_0 + c_1 \Rightarrow t_0 + t_1$ holds, where $c_0 + c_1 \triangleq (\bar{C}, (\mathsf{X} \mapsto |\psi_0\rangle + |\psi_1\rangle), S)$ and $t_0 + t_1$ is the tree of the same structure as t_0 and t_1 (the two agree in the structure) whose leaves are* $\mathsf{Leaf}(a_0 +_{Res} a_1)$ *letting* $\mathsf{Leaf}(a_i)$ *be the leaf of t_i at the same position.*

PROOF. By induction on the definition of the denotational semantics. This can also be proven using Theorem 1 and the linearity of the operational semantics. □

PROOF OF HOARE-SUM, FIG. 15. Follows from Lemma 13 and the fact that there is a canonical multiset bijection between $\mathsf{Leaves}(t)$ and $\mathsf{Leaves}(t')$ whenever t and t' have the same shape. □

PROOF OF HOARE-SEQ, FIG. 13. Let $\{\!\{a_i \in Res \mid i \in I\}\!\} \in P$ and $(C, a_i) \Rightarrow t_i$. Then, we have $\biguplus_{i \in I} \mathsf{Leaves}(t_i) \in Q$. We also know that for each $b_{ij} \in_{\mathrm{m}} \mathsf{Leaves}(t_i), \exists t_{ij}. (C', b_{ij}) \Rightarrow t_{ij}$, satisfying $\biguplus_i \biguplus_j \mathsf{Leaves}(t_{ij}) \in R$. Now, from the definition of the denotational semantics, we have $(C; C', a_i) \Rightarrow t_i[t_{ij}/\mathsf{Leaf}(b_{ij})]$. The multiset of leaves of the tree $t_i[t_{ij}/\mathsf{Leaf}(b_{ij})]$ is $\biguplus_j \mathsf{Leaves}(t_{ij})$. Therefore, $\biguplus_i \mathsf{Leaves}(t_i[t_{ij}/\mathsf{Leaf}(b_{ij})]) = \biguplus_i \biguplus_j \mathsf{Leaves}(t_{ij}) \in R$. □

PROOF OF HOARE-WHILE, FIG. 13. From the definition of the denotation of the while loop, the following holds.

$$[\![\mathsf{while}\ e\ \mathsf{do}\ C]\!] = [\![\mathsf{if}\ e\ \mathsf{then}\ (C;\ \mathsf{while}\ e\ \mathsf{do}\ C)]\!]$$

Since the denotation of the while loop is defined by the least fixed point, this can be rewritten as

$$[\![\mathsf{while}\ e\ \mathsf{do}\ C]\!] = \sup_{n \in \mathbb{N}} [\![\mathsf{if}\ e\ \mathsf{then}\ (C;\ \ldots \mathsf{if}\ e\ \mathsf{then}\ (C;\ \mathsf{if}\ e\ \mathsf{then}\ \mathsf{loop})\ldots)]\!]$$

$$= \sup_{n \in \mathbb{N}} [\![(\mathsf{if}\ e\ \mathsf{then}\ C)^n,\ \mathsf{if}\ e\ \mathsf{then}\ \mathsf{loop}]\!]$$

where loop is the infinite loop whose denotation is $\lambda\_.\mathsf{Nil}$, and $\mathsf{iter}_n \triangleq (\mathsf{if}\ e\ \mathsf{then}\ C)^n$ denotes that the command $(\mathsf{if}\ e\ \mathsf{then}\ C)$ is executed n times in sequence. Here, we used the following equation and the compositionality of the denotational semantics.

$$[\![\mathsf{if}\ e\ \mathsf{then}\ (C;\ \mathsf{if}\ e\ \mathsf{then}\ C')]\!](|\psi\rangle, S)$$

$$= \begin{cases} [\![C;\ \mathsf{if}\ e\ \mathsf{then}\ C']\!](|\psi\rangle, S) & \text{if } [\![e]\!]_S = 1 \\ [\![\mathsf{skip}]\!](|\psi\rangle, S) & \text{otherwise} \end{cases}$$

$$= \begin{cases} [\![C,\ \mathsf{if}\ e\ \mathsf{then}\ C']\!](|\psi\rangle, S) & \text{if } [\![e]\!]_S = 1 \\ [\![\mathsf{skip},\ \mathsf{if}\ e\ \mathsf{then}\ C']\!](|\psi\rangle, S) & \text{otherwise} \end{cases}$$

$$= [\![\mathsf{if}\ e\ \mathsf{then}\ C,\ \mathsf{if}\ e\ \mathsf{then}\ C']\!](|\psi\rangle, S).$$

Let $\{\!\{a_i \in Res \mid i \in I\}\!\}\in P_0$, and $(\mathsf{while}\ e\ \mathsf{do}\ C, a_i) \Rightarrow t_{i,\infty}$. Then, there exists a sequence of trees $t_{i,n}$ such that $(\mathsf{iter}_n, a_i) \Rightarrow t_{i,n}$. We define multisets from such trees.

$$m_{i,0}^P \triangleq \{\!\{a_i\}\!\}, \qquad m_{i,n+1}^P \triangleq \{\!\{b \mid a \in_\mathsf{m} m_{i,n}^R, b \in_\mathsf{m} \mathsf{Leaves}(\llbracket C \rrbracket(a))\}\!\}$$
$$m_{i,n}^Q \triangleq \{\!\{(|\psi\rangle, S) \mid (|\psi\rangle, S) \in_\mathsf{m} m_{i,n}^P, \llbracket e \rrbracket_S = 0\}\!\}$$
$$m_{i,n}^R \triangleq \{\!\{(|\psi\rangle, S) \mid (|\psi\rangle, S) \in_\mathsf{m} \mathsf{Leaves}(t_{i,n}), \llbracket e \rrbracket_S = 1\}\!\}$$

Note that this is not a recursive definition: the definition of m^P and m^Q depends on m^R, but m^R depends only on $t_{i,n}$. We first prove that $\biguplus_i m_{i,n}^R \in R_n$ by induction on n. For the base case $n = 0$, since $\mathsf{iter}_0 = \mathsf{skip}$, the tree $t_{i,0}$ is $\mathsf{Leaf}(a_i)$. So, $\biguplus_i m_{i,0}^R = \{\!\{a_i \mid i \in I, a_i = (|\psi\rangle, S), \llbracket e \rrbracket_S = 1\}\!\} \in R_0$ since $\{\!\{a_i \mid i \in I\}\!\} \in P_0$. For the step case,

$$\begin{aligned}
\biguplus_i m_{i,n+1}^R &= \biguplus_i \{\!\{(|\psi\rangle, S) \mid (|\psi\rangle, S) \in_\mathsf{m} \mathsf{Leaves}(t_{i,n+1}), \llbracket e \rrbracket_S = 1\}\!\} \\
&= \biguplus_i \{\!\{(|\psi'\rangle, S') \mid (|\psi\rangle, S) \in_\mathsf{m} \mathsf{Leaves}(t_{i,n}), \llbracket e \rrbracket_S = 1, \\
&\qquad\qquad (|\psi'\rangle, S') \in_\mathsf{m} \mathsf{Leaves}(\llbracket C \rrbracket(|\psi\rangle, S)), \llbracket e \rrbracket_{S'} = 1\}\!\} \\
&= \biguplus_i \{\!\{(|\psi'\rangle, S') \mid a \in_\mathsf{m} m_{i,n}^R, (|\psi'\rangle, S') \in_\mathsf{m} \mathsf{Leaves}(\llbracket C \rrbracket(a)), \llbracket e \rrbracket_{S'} = 1\}\!\} \\
&= \biguplus_i \biguplus_{a \in_\mathsf{m} m_{i,n}^R} \{\!\{(|\psi'\rangle, S') \mid (|\psi'\rangle, S') \in_\mathsf{m} \mathsf{Leaves}(\llbracket C \rrbracket(a)), \llbracket e \rrbracket_{S'} = 1\}\!\} \\
&= \biguplus_{a \in_\mathsf{m} \biguplus_i m_{i,n}^R} \{\!\{(|\psi'\rangle, S') \mid (|\psi'\rangle, S') \in_\mathsf{m} \mathsf{Leaves}(\llbracket C \rrbracket(a)), \llbracket e \rrbracket_{S'} = 1\}\!\} \\
&\in R_{n+1}.
\end{aligned}$$

The statements $\biguplus_i m_{i,n}^P \in P_n$ and $\biguplus_i m_{i,n}^Q \in Q_n$ follow from this.

By an easy induction, one can prove that the leaves of $t_{i,n}$ can be represented as $m_{i,n}^P \uplus \biguplus_{k<n} m_{i,k}^Q = m_{i,n}^R \uplus \biguplus_{k \le n} m_{i,k}^Q$. Therefore, the leaves of $\llbracket(\mathsf{iter}_n; \mathsf{if}\ e\ \mathsf{then}\ \mathsf{loop})\rrbracket(a_i)$ are $\biguplus_{k \le n} m_{i,k}^Q$. Taking the supremum of these trees, we conclude that

$$\biguplus_{i \in I} \mathsf{Leaves}(t_{i,\infty}) = \biguplus_{i \in I} \sup_{n \in \mathbb{N}} \llbracket(\mathsf{iter}_n; \mathsf{if}\ e\ \mathsf{then}\ \mathsf{loop})\rrbracket(a_i) = \biguplus_{i \in I, n \in \mathbb{N}} m_{i,n}^Q \in \bigoplus_{n \in \mathbb{N}} Q_n. \qquad \square$$

A.7 Inner Product and Orthogonality

Finally, to reason about the probability of the sum, we introduce the inner product and orthogonality, based on elementary linear algebra.

Definition 14 (Inner product). We define the *inner product* of resources $\langle\,,\,\rangle: A \times A \rightharpoonup \mathbb{C}$ for each PCM $A \in \{Qstate, Store, Res\}$ as follows.

$$\langle \mathsf{X} \mapsto |\psi\rangle, \mathsf{X} \mapsto |\phi\rangle \rangle_{Qstate} \triangleq \langle |\psi\rangle, |\phi\rangle \rangle = \langle\psi|\phi\rangle \qquad\qquad \langle S, S \rangle_{Store} \triangleq 1$$
$$\langle (\mathsf{X} \mapsto |\psi\rangle, S), (\mathsf{Y} \mapsto |\phi\rangle, S') \rangle_{Res} \triangleq \langle \mathsf{X} \mapsto |\psi\rangle, \mathsf{Y} \mapsto |\phi\rangle \rangle_{Qstate} \cdot \langle S, S' \rangle_{Store}$$

The inner product is just the usual one on vectors whenever the classical information (including the set of owned qubits) agrees, and is undefined otherwise.

Just like for the sum + (Fig. 15, § 4.5), we define the inner product on multisets via multiset bijections r and lift it to propositions by collecting the resulting values.

$$\langle m, m' \rangle_r \triangleq \sum_{(a,b)\in_\mathsf{m} r} \langle a, b \rangle \qquad\qquad \langle P, Q \rangle \triangleq \left\{ \langle m, m' \rangle_r \mid m \in P, m' \in Q, r: m \cong_\mathsf{m} m' \right\}$$

For utility, we also introduce the predicate $\langle P, Q \rangle : \alpha$ to assert that the inner product evaluates to α whenever defined:

$$\langle P, Q \rangle : \alpha \quad \triangleq \quad \langle P, Q \rangle \subseteq \{\alpha\}. \qquad \qquad \square$$

Remark 15. The probability predicate can be characterized through the inner product as

$$P : \mathrm{prob}\, p \quad \Longleftrightarrow \quad \forall m \in P.\, \langle m, m \rangle_{\mathrm{id}} = p,$$

where id is the canonical identity multiset bijection. Also, under the constraint P : precise, unambig, P : prob p is equivalent to $\langle P, P \rangle : p$.

Definition 16 (Orthogonality). The orthogonality $P \perp Q$ of SL assertions is defined as follows:

$$P \perp Q \quad \triangleq \quad \langle P, Q \rangle : 0. \qquad \qquad \square$$

As an auxiliary notion for reasoning about the inner product and orthogonality, we also introduce the *coherence*.

Definition 17 (Coherence). We say $a, b \in Res$ are *coherent* and write $a \frown b$ if their inner product is defined, i.e., $\langle a, b \rangle \downarrow$. This is equivalent to $a + b \downarrow$. We say multisets $m, m' \in \mathcal{M}(Res)$ are *coherent* if their inner product is defined for some multiset bijection r, i.e., $\exists r : m \cong_{\mathrm{m}} m'.\, \langle m, m' \rangle_r \downarrow$.

The coherence relation $P \frown Q$ for SL assertions is defined as follows:

$$P \frown Q \quad \triangleq \quad \forall m \in P.\, \forall m' \in Q.\, m \frown m' \qquad \qquad \square$$

We have the following rules for the coherence $\frown$, whose soundness can be proved easily:

$$\bar{\mathrm{x}} \mapsto |\psi\rangle \frown \bar{\mathrm{x}} \mapsto |\phi\rangle \qquad \frac{P : \text{precise}}{P \frown P} \qquad \frac{P \frown Q}{Q \frown P} \qquad \frac{P \frown Q \quad P' \vdash P}{P' \frown Q} \qquad \frac{\forall x \in I.\, (P_x \frown Q)}{(\exists x \in I.\, P_x) \frown Q}$$

$$\frac{P \frown Q \quad P' \frown Q'}{P * P' \frown Q * Q'} \qquad \frac{P \frown Q \quad P' \frown Q'}{P + P' \frown Q + Q'} \qquad \frac{\forall x \in I.\, P_x \frown Q_x}{\bigoplus_{x \in I} P_x \frown \bigoplus_{x \in I} Q_x}$$

Now we list the rules for the inner product and orthogonality.

INNER-PROD-UNIQUE

$$\langle \bar{\mathrm{x}} \mapsto |\psi\rangle, \bar{\mathrm{x}} \mapsto |\phi\rangle \rangle : \langle \psi | \phi \rangle \qquad \langle P, Q \rangle = \overline{\langle Q, P \rangle} \qquad \frac{P, Q : \text{precise} \qquad P : \text{unambig}}{\exists \alpha \in \mathbb{C}.\, \langle P, Q \rangle : \alpha}$$

INNER-PROD-SUM-EXACT

$$\frac{\alpha \in \mathbb{C}}{\langle P, \alpha Q \rangle = \alpha \langle P, Q \rangle} \qquad \text{INNER-PROD-SUM} \qquad \frac{P : \text{precise}}{\langle P, Q + R \rangle \subseteq \langle P, Q \rangle + \langle P, R \rangle} \qquad \frac{P : \text{precise}}{\langle P, Q + R \rangle = \langle P, Q \rangle + \langle P, R \rangle}$$

INNER-PROD-BIGMIX-EXACT

INNER-PROD-BIGMIX

$$\sum_{x \in I} \langle P_x, Q_x \rangle \subseteq \left\langle \bigoplus_{x \in I} P_x, \bigoplus_{x \in I} Q_x \right\rangle \qquad \frac{\forall x, y \in I \text{ s.t. } x \neq y.\, P_x \# Q_y}{\sum_{x \in I} \langle P_x, Q_x \rangle = \left\langle \bigoplus_{x \in I} P_x, \bigoplus_{x \in I} Q_x \right\rangle}$$

INNER-PROD-FRAME   INNER-PROD-UNFRAME

$$\frac{\forall m \in P, m' \in P'.\, m \cdot m' \downarrow}{\langle P, Q \rangle \cdot \langle P', Q' \rangle \subseteq \langle P * P', Q * Q' \rangle} \qquad \frac{P : \text{unambig} \qquad P \frown Q \qquad \langle R, R' \rangle : \alpha}{\langle P * R, Q * R' \rangle \subseteq \alpha \langle P, Q \rangle}$$

ORTH-PROB

$$\frac{P, Q : \text{prob } p \qquad P \perp Q \qquad \alpha, \beta \in \mathbb{C} \qquad |\alpha|^2 + |\beta|^2 = 1}{\alpha P + \beta Q : \text{prob } p}$$

ORTH-SUM

$$\frac{P \perp R \qquad Q \perp R}{P + Q \perp R}$$

ORTH-BIGMIX

$$\frac{\forall x \in I. P_x \perp Q_x \qquad \forall x, y \in I \text{ s.t. } x \neq y. P_x \# Q_y}{\bigoplus_{x \in I} P_x \perp \bigoplus_{x \in I} Q_x}$$

ORTH-FRAME

$$\frac{P, R : \text{unambig} \qquad P \supset Q \qquad P \perp Q}{P * R \perp Q * R'}$$

We have the coherence side condition $P \supset Q$ for the rules INNER-PROD-UNFRAME and ORTH-FRAME. Some of the above rules are used in the case study of the Shor code in §B.6.

Before proving the soundness of these rules, we discuss useful properties.

Lemma 18 (Basic properties of the inner product on *Res*). *The following hold for any $a, b, c, d \in Res$:*

$$\langle a, b \rangle = \overline{\langle b, a \rangle} \qquad\qquad \langle 1, \alpha \rangle = \alpha$$

$$\langle a + b, c \rangle = \langle a, c \rangle + \langle b, c \rangle \qquad\qquad \langle c, a + b \rangle = \langle c, a \rangle + \langle c, b \rangle$$

$$a \supset b \implies \langle a \cdot c, b \cdot d \rangle = \langle a, b \rangle \cdot \langle c, d \rangle.$$

PROOF. Immediately derived from the usual vector calculus. □

By combining these rules, we can, for example, prove the anti-linearity of the first argument as $\langle \alpha a + \beta b, c \rangle = \langle \alpha a, c \rangle + \langle \beta b, c \rangle = \langle \alpha \cdot a, 1 \cdot c \rangle + \langle \beta \cdot b, 1 \cdot c \rangle = \langle \alpha, 1 \rangle \cdot \langle a, c \rangle + \langle \beta, 1 \rangle \cdot \langle b, c \rangle = \overline{\langle 1, \alpha \rangle} \cdot \langle a, c \rangle + \overline{\langle 1, \beta \rangle} \cdot \langle b, c \rangle = \bar{\alpha}\langle a, c \rangle + \bar{\beta}\langle b, c \rangle.$

Lemma 19 (More properties on *Res* and $\mathcal{M}(Res)$). *The following hold for any $a, b, c \in Res$ and $m_1, m_2, m_3 \in \mathcal{M}(Res)$.*

- *If $a \supset b$ and $a \cdot c \downarrow$, then $(a + b) \cdot c \downarrow$.*
- *If $a \supset b$ and $a \# c$, then $b \# c$.*
- *If $m_1 +_r m_2 \downarrow$ and $m_1 \cdot m_3 \downarrow$, then $m_2 \cdot m_3 \downarrow$.*

PROOF. Straightforward. □

Now we prove the soundness of the proof rules for the inner product and orthogonality.

PROOF OF INNER-PROD-UNIQUE. This can be proven similarly to SUM-PRECISE. □

PROOF OF INNER-PROD-SUM. An element in the left-hand side can be written as $\langle m, m_1 +_r m_2 \rangle_{r'}$ where $m \in P$, $m_1 \in Q$, and $m_2 \in R$ with multiset bijections $r : m_1 \cong_m m_2$ and $r' : m \cong_m (m_1 +_r m_2)$. From these mutiset bijections, we obtain a canonical multiset bijection $s : m \cong_m m_1$ and $s' : m \cong_m m_2$. Then $\langle m, m_1 +_r m_2 \rangle_{r'} = \langle m, m_1 \rangle_s + \langle m, m_2 \rangle_{s'} \in$ RHS. □

PROOF OF INNER-PROD-SUM-EXACT. If $P = \{m\}$, an element in the right-hand side can be written as $\langle m, m_1 \rangle_s + \langle m, m_2 \rangle_{s'} = \langle m, m_1 +_r m_2 \rangle_{r'} \in$ LHS. □

PROOF OF INNER-PROD-BIGMIX. Any element in the left-hand side is written as $\sum_{x \in I} \langle m_x, m'_x \rangle_{r_x}$, which is equal to $\langle \biguplus_x m_x, \biguplus_x m'_x \rangle_{\biguplus_{x \in I} r_x} \in$ RHS. □

PROOF OF INNER-PROD-BIGMIX-EXACT. Let $m_x \in P_x$ and $m'_x \in Q$. As in the proof of BIGMIX-SUM, any multiset bijection r between $m \triangleq \biguplus_x m_x$ and $m' \triangleq \biguplus_x m'_x$ that satisfy $m +_r m' \downarrow$ has the form $\biguplus_x r_x$ for some $r_x : m_x \cong_m m'_x$. Therefore, $\langle \biguplus_x m_x, \biguplus_x m'_x \rangle_r = \sum_x \langle m_x, m'_x \rangle_{r_x} \in$ LHS. □

Proof of INNER-PROD-FRAME. Any element in the left-hand side can be written as $\langle m_1 +_r m_2, m_3 +_s m_4 \rangle$ where $m_1 \in P$, $m_2 \in Q$, $m_3 \in P'$, and $m_4 \in Q'$ with multiset bijections $r \colon m_1 \cong_m m_2$ and $s \colon m_3 \cong_m m_4$. Thanks to the assumption, $m_1 \cdot m_3$ is defined. Because $m_1 +_r m_3$ and $m_2 +_s m_4$ are defined, $m_2 \cdot m_4 \downarrow$ follows from Lemma 19. There is an induced multiset bijection $r \times s \colon m_1 \cdot m_3 \cong_m m_2 \cdot m_4$ that satisfies

$$\langle m_1 +_r m_2, m_3 +_s m_4 \rangle = \{\!\{ \langle a, b \rangle \cdot \langle c, d \rangle \mid (a, b) \in_m r, (c, d) \in_m s \}\!\}$$
$$= \{\!\{ \langle a \cdot c, b \cdot d \rangle \mid (a \cdot c, b \cdot d) \in_m r \times s \}\!\} \in \text{RHS}. \qquad \square$$

Proof of INNER-PROD-UNFRAME. Let $\langle m \cdot m_1, m' \cdot m_2 \rangle_r$ be an element of left-hand side where $m \in P$, $m' \in Q$, $m_1 \in R$, $m_2 \in R'$, and $r \colon m \cdot m_1 \cong_m m' \cdot m_2$. Since $P \supset Q$, there exists some multiset bijection $s_0 \colon m \cong_m m'$ such that $m +_{s_0} m' \downarrow$. Since P is unambiguous, similarly to the proof of SUM-PRECISE, we can prove that such s_0 is unique. From Lemma 19, for any $a \in_m m$, $b \in_m m'$, $(a, b) \in_m s_0$ or $a \# b$. Because $\langle m \cdot m_1, m' \cdot m_2 \rangle_r$ is defined, for any $(a \cdot c, b \cdot d) \in_m r$, $\langle a \cdot c, b \cdot d \rangle \downarrow$, which is equivalent to say $a \cdot c + b \cdot d \downarrow$. So $\neg(a \# b)$, thus $(a, b) \in_m s_0$. Therefore, $(a \cdot c, b \cdot d) \in_m r \implies (a, b) \in_m s_0$, and $\langle a \cdot c, b \cdot d \rangle = \langle a, b \rangle \cdot \langle c, d \rangle$. For each $a \in_m m$, let r_a be the multiset bijection between m_1 and m_2 such that $(a \cdot c, b \cdot d) \in_m r \implies (c, d) \in_m r_a$ Then

$$\sum\nolimits_{(a \cdot c, b \cdot d) \in_m r} \langle a \cdot c, b \cdot d \rangle$$
$$= \sum\nolimits_{(a, b) \in_m s_0} \sum\nolimits_{(c, d) \in_m r_a} \langle a, b \rangle \cdot \langle c, d \rangle$$
$$= \sum\nolimits_{(a, b) \in_m s_0} \langle a, b \rangle \cdot \sum\nolimits_{(c, d) \in_m r_a} \langle c, d \rangle$$
$$= \sum\nolimits_{(a, b) \in_m r_0} \alpha \langle a, b \rangle$$
$$= \alpha \langle m, m' \rangle_{r_0} \in \alpha \langle P, Q \rangle. \qquad \square$$

Proof of ORTH-PROB. Let $m \in P$, $m' \in Q$, and $r \colon m \cong_m m'$. Then

$$\langle \alpha m +_r \beta m', \alpha m +_r \beta m' \rangle_{\text{id}} = \bar{\alpha}\alpha \langle m, m \rangle_{\text{id}} + \bar{\alpha}\beta \langle m, m' \rangle_r + \bar{\beta}\alpha \langle m', m \rangle_{r^{-1}} + \bar{\beta}\beta \langle m', m' \rangle_{\text{id}}$$
$$= |\alpha|^2 + 0 + 0 + |\beta|^2 = 1. \qquad \square$$

Proof of ORTH-SUM. Follows from INNER-PROD-SUM. $\qquad \square$

Proof of ORTH-BIGMIX. Follows from INNER-PROD-BIGMIX-EXACT. $\qquad \square$

Proof of ORTH-FRAME. Let $m \in P$, $m' \in Q$, $m_1 \in R$, $m_2 \in R'$, and $r \colon m \cdot m_1 \cong_m m' \cdot m_2$, such that $\langle m \cdot m_1, m' \cdot m_2 \rangle_r \downarrow$. Similarly to the proof of INNER-PROD-UNFRAME, we can prove that, there exist $s_0 \colon m \cong_m m'$ and $r_a \colon m_1 \cong_m m_2$ such that $(a \cdot c, b \cdot d) \in_m r \implies (a, b) \in_m s_0 \land (c, d) \in_m r_a$, $m +_{s_0} m' \downarrow$, and $m_1 +_{r_a} m_2 \downarrow$. Moreover, since R is unambiguous, $\{ s' \colon m_1 \cong_m m_2 \mid m_1 +_{s'} m_2 \downarrow \}$ is a singleton or empty. If not empty, we denote the unique element as s_0', and then

$$\sum\nolimits_{(a \cdot c, b \cdot d) \in_m r} \langle a \cdot c, b \cdot d \rangle = \sum\nolimits_{(a, b) \in_m s_0} \langle a, b \rangle \cdot \sum\nolimits_{(c, d) \in_m r_a} \langle c, d \rangle$$
$$= \sum\nolimits_{(a, b) \in_m s_0} \langle a, b \rangle \cdot \sum\nolimits_{(c, d) \in_m s_0'} \langle c, d \rangle$$
$$\in \{0\}.$$

If such s_0' does not exist, then $m = m' = \{\!\{ \}\!\}$. The orthogonality is trivial in this case. $\qquad \square$

B More on Case Studies

B.1 Dirty Qubit: Implementation of CCCX by Toffoli Gates

Here we give the details of § 5.1.

The CCCX gate is a 4-qubit gate that computes $\mathsf{ret} \leftarrow \mathsf{ret} \veebar (\mathsf{x} \wedge \mathsf{y} \wedge \mathsf{z})$ where $\veebar$ denotes exclusive OR. We can implement this CCCX gate using only CCX gates (computing $\mathsf{ret} \leftarrow \mathsf{ret} \veebar (\mathsf{x} \wedge \mathsf{y})$, also called Toffoli gates) with a dirty auxiliary qubit tmp, a qubit whose state is unknown, as follows:

$$\mathsf{dCCCX}[\mathsf{x}, \mathsf{y}, \mathsf{z}, \mathsf{tmp}, \mathsf{ret}] \triangleq \mathsf{CCX}[\mathsf{z}, \mathsf{tmp}, \mathsf{ret}]; \mathsf{CCX}[\mathsf{x}, \mathsf{y}, \mathsf{tmp}]; \mathsf{CCX}[\mathsf{z}, \mathsf{tmp}, \mathsf{ret}]; \mathsf{CCX}[\mathsf{x}, \mathsf{y}, \mathsf{tmp}].$$

The CCCX gate and the circuit dCCCX are visualized in Fig. 20.

We would like to prove the equivalence between CCCX and dCCCX in our logic in any context, i.e., with any other qubits entangled with the qubits involved. We show the following Hoare triple:

$$\big\{ (\mathsf{x}, \mathsf{y}, \mathsf{z}, \mathsf{tmp}, \mathsf{ret}, \bar{w}) \mapsto |\psi\rangle \big\} \; \mathsf{dCCCX} \; \big\{ (\mathsf{x}, \mathsf{y}, \mathsf{z}, \mathsf{tmp}, \mathsf{ret}, \bar{w}) \mapsto (\mathsf{CCCX}_{\mathsf{x},\mathsf{y},\mathsf{z},\mathsf{ret}} \otimes \mathsf{id}_{\mathsf{tmp},\bar{w}}) |\psi\rangle \big\}.$$

To this end, we first prove the following Hoare triple for any classical state $i, j, k, \ell, m \in \{0, 1\}$:

$$\big\{ (\mathsf{x}, \mathsf{y}, \mathsf{z}, \mathsf{tmp}, \mathsf{ret}) \mapsto |ijk\ell m\rangle \big\} \; \mathsf{dCCCX} \; \big\{ (\mathsf{x}, \mathsf{y}, \mathsf{z}, \mathsf{tmp}, \mathsf{ret}) \mapsto |ijk\ell(m \veebar (i \wedge j \wedge k))\rangle \big\}.$$

This can be done as follows. Here, we simply write $\wedge$ as multiplication. Note that this part is no different from symbolic execution of the usual classical bitwise operations.

$$\big\{ \mathsf{x} \mapsto |i\rangle * \mathsf{y} \mapsto |j\rangle * \mathsf{z} \mapsto |k\rangle * \mathsf{tmp} \mapsto |\ell\rangle * \mathsf{ret} \mapsto |m\rangle \big\}$$

$$\mathsf{CCX}[\mathsf{z}, \mathsf{tmp}, \mathsf{ret}]$$

$$\big\{ \mathsf{x} \mapsto |i\rangle * \mathsf{y} \mapsto |j\rangle * \mathsf{z} \mapsto |k\rangle * \mathsf{tmp} \mapsto |\ell\rangle * \mathsf{ret} \mapsto |m \veebar k\ell\rangle \big\}$$

$$\mathsf{CCX}[\mathsf{x}, \mathsf{y}, \mathsf{tmp}]$$

$$\big\{ \mathsf{x} \mapsto |i\rangle * \mathsf{y} \mapsto |j\rangle * \mathsf{z} \mapsto |k\rangle * \mathsf{tmp} \mapsto |\ell \veebar ij\rangle * \mathsf{ret} \mapsto |m \veebar k\ell\rangle \big\}$$

$$\mathsf{CCX}[\mathsf{z}, \mathsf{tmp}, \mathsf{ret}]$$

$$\big\{ \mathsf{x} \mapsto |i\rangle * \mathsf{y} \mapsto |j\rangle * \mathsf{z} \mapsto |k\rangle * \mathsf{tmp} \mapsto |\ell \veebar ij\rangle * \mathsf{ret} \mapsto |m \veebar k\ell \veebar k(\ell \veebar ij)\rangle \big\}$$

$$\big\{ \mathsf{x} \mapsto |i\rangle * \mathsf{y} \mapsto |j\rangle * \mathsf{z} \mapsto |k\rangle * \mathsf{tmp} \mapsto |\ell \veebar ij\rangle * \mathsf{ret} \mapsto |m \veebar ijk\rangle \big\}$$

$$\mathsf{CCX}[\mathsf{x}, \mathsf{y}, \mathsf{tmp}]$$

$$\big\{ \mathsf{x} \mapsto |i\rangle * \mathsf{y} \mapsto |j\rangle * \mathsf{z} \mapsto |k\rangle * \mathsf{tmp} \mapsto |\ell \veebar ij \veebar ij\rangle * \mathsf{ret} \mapsto |m \veebar ijk\rangle \big\}$$

$$\big\{ \mathsf{x} \mapsto |i\rangle * \mathsf{y} \mapsto |j\rangle * \mathsf{z} \mapsto |k\rangle * \mathsf{tmp} \mapsto |\ell\rangle * \mathsf{ret} \mapsto |m \veebar ijk\rangle \big\}$$

Now that we have proven the Hoare triple for the concrete state $|ijk\ell m\rangle$, we can generalize it to any quantum state $|\psi\rangle$ as follows:

$$\frac{\forall i, j, k, \ell, m \in \{0, 1\}.}{\big\{ (\mathsf{x}, \mathsf{y}, \mathsf{z}, \mathsf{tmp}, \mathsf{ret}) \mapsto |ijk\ell m\rangle \big\} \, \mathsf{dCCCX} \, \big\{ (\mathsf{x}, \mathsf{y}, \mathsf{z}, \mathsf{tmp}, \mathsf{ret}) \mapsto (\mathsf{CCCX}_{\mathsf{x},\mathsf{y},\mathsf{z},\mathsf{ret}} \otimes \mathsf{id}_{\mathsf{tmp}}) |ijk\ell m\rangle \big\}} \quad (17)$$

$$\frac{\forall i, j, k, \ell, m \in \{0, 1\}.}{\begin{array}{c}\big\{ (\mathsf{x}, \mathsf{y}, \mathsf{z}, \mathsf{tmp}, \mathsf{ret}) \mapsto \alpha_{ijk\ell m} |ijk\ell m\rangle * \bar{w} \mapsto \big|\psi_{ijk\ell m}\big\rangle \big\} \, \mathsf{dCCCX} \\ \big\{ (\mathsf{x}, \mathsf{y}, \mathsf{z}, \mathsf{tmp}, \mathsf{ret}) \mapsto (\mathsf{CCCX}_{\mathsf{x},\mathsf{y},\mathsf{z},\mathsf{ret}} \otimes \mathsf{id}_{\mathsf{tmp}}) \alpha_{ijk\ell m} |ijk\ell m\rangle * \bar{w} \mapsto \big|\psi_{ijk\ell m}\big\rangle \big\}\end{array}} \quad (16)$$

$$\frac{}{\begin{array}{c}\big\{ (\mathsf{x}, \mathsf{y}, \mathsf{z}, \mathsf{tmp}, \mathsf{ret}, \bar{w}) \mapsto \sum_{i,j,k,\ell,m} \alpha_{ijk\ell m} |ijk\ell m\rangle \big|\psi_{ijk\ell m}\big\rangle \big\} \, \mathsf{dCCCX} \\ \big\{ (\mathsf{x}, \mathsf{y}, \mathsf{z}, \mathsf{tmp}, \mathsf{ret}, \bar{w}) \mapsto (\mathsf{CCCX}_{\mathsf{x},\mathsf{y},\mathsf{z},\mathsf{ret}} \otimes \mathsf{id}_{\mathsf{tmp},\bar{w}}) \sum_{i,j,k,\ell,m} \alpha_{ijk\ell m} |ijk\ell m\rangle \big|\psi_{ijk\ell m}\big\rangle \big\}\end{array}}$$

We first use HOARE-FRAME to add the auxiliary qubits $\bar{w} \mapsto \big|\psi_{ijk\ell m}\big\rangle$ and the global phase $\alpha_{ijk\ell m}$ in the pre/postconditions at step (17). Finally, we use HOARE-SUM to sum over all $ijk\ell m \in \{0, 1\}$ to obtain the last line step (16). We used the linearity of the CCCX gate to commute the CCCX gate with the sum $\sum_{i,j,k,\ell,m}$ and the global phase $\alpha_{ijk\ell m}$. Since the choice of $\alpha_{ijk\ell m}$ and $\big|\psi_{ijk\ell m}\big\rangle$ is

arbitrary, and every quantum state can be written in the form $\sum_{i,j,k,\ell,m} \alpha_{ijk\ell m} |ijk\ell m\rangle |\psi_{ijk\ell m}\rangle$, we conclude that the Hoare triple holds for any quantum state $|\psi\rangle$.

B.2 Program with Measurements: EPR Paradox

The EPR paradox is a famous quantum phenomenon that shows that the measurement results of two entangled qubits are correlated, no matter how far apart the two qubits are. The *Bell state* or *EPR pair* is a state of 2-qubits (x and y) written as $\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. We suppose that those entangled qubits are shared between Alice and Bob. If Alice measures her qubit x and Bob measures his qubit y, the EPR paradox states that their measurement results agree with probability 1, even though the outcomes are not determined prior to measurement. The program can be written as follows, where the Bell state is prepared in the first step:

$$\mathsf{EPR}^{a,b,r}[\mathsf{x},\mathsf{y}] \quad \triangleq \quad \mathsf{Bell}[\mathsf{x},\mathsf{y}];\ a \leftarrow \mathsf{M}_\mathsf{Z}[\mathsf{x}];\ b \leftarrow \mathsf{M}_\mathsf{Z}[\mathsf{y}];\ r \leftarrow a \veebar b.$$

The classical boolean variable r is added to the program to indicate the difference between the measurement results of Alice and Bob. Therefore, the correctness of the program can be specified as the following assertion:

$$\exists\, P : \text{frameable, prob 1},\ Q : \text{frameable, prob 0}.$$
$$\big\{ (\mathsf{x},\mathsf{y}) \mapsto |00\rangle \big\}^{a,b,r} \ \mathsf{EPR}^{a,b,r}[\mathsf{x},\mathsf{y}] \ \big\{ (r \mapsto 0 * P) \oplus (r \mapsto 1 * Q) \big\}.$$

This assertion can be proved as follows, setting $R_{k,\ell} \triangleq a \mapsto k * b \mapsto \ell * (\mathsf{x},\mathsf{y}) \mapsto |k\ell\rangle$:

$$\big\{ (\mathsf{x},\mathsf{y}) \mapsto |00\rangle \big\}^{a,b,r}$$
$$\mathsf{Bell}[\mathsf{x},\mathsf{y}] \quad \big\{ \tfrac{1}{\sqrt{2}} \sum_{i=0,1} (\mathsf{x},\mathsf{y}) \mapsto |ii\rangle \big\}^{a,b,r}$$
$$a \leftarrow \mathsf{M}_\mathsf{Z}[\mathsf{x}] \quad \big\{ \tfrac{1}{\sqrt{2}} \bigoplus_{i=0,1} a \mapsto i * (\mathsf{x},\mathsf{y}) \mapsto |ii\rangle \big\}^{b,r}$$
$$b \leftarrow \mathsf{M}_\mathsf{Z}[\mathsf{y}] \quad \big\{ \tfrac{1}{\sqrt{2}} \bigoplus_{i=0,1} \bigoplus_{j=0,1} \delta_{ij} * a \mapsto i * b \mapsto i * (\mathsf{x},\mathsf{y}) \mapsto |ii\rangle \big\}^{r}$$
$$\Big\{ \Big(\tfrac{1}{\sqrt{2}} \bigoplus_{k=0,1} R_{k,k} \Big) \oplus \Big(0 \cdot \bigoplus_{k=0,1} R_{k,\neg k} \Big) \Big\}^{r}$$
$$r \leftarrow a \veebar b$$
$$\Big\{ \Big(\tfrac{1}{\sqrt{2}} \bigoplus_{k=0,1} r \mapsto 0 * R_{k,k} \Big) \oplus \Big(0 \cdot \bigoplus_{k=0,1} r \mapsto 1 * R_{k,\neg k} \Big) \Big\}$$
$$\Big\{ \Big(r \mapsto 0 * \tfrac{1}{\sqrt{2}} \bigoplus_{k=0,1} R_{k,k} \Big) \oplus \Big(r \mapsto 1 * 0 \cdot \bigoplus_{k=0,1} R_{k,\neg k} \Big) \Big\}.$$

We can finally set $P \triangleq \tfrac{1}{\sqrt{2}} \bigoplus_{k=0,1} R_{k,k}$ and $Q \triangleq 0 \cdot \bigoplus_{k=0,1} R_{k,\neg k}$. Note that P : prob 1 holds because $R_{k,\ell}$: prob 1 and $\left(\tfrac{1}{\sqrt{2}} \right)^2 \cdot (1+1) = 1$.

B.3 Quantum Teleportation

Here we give the details of § 5.2.

Quantum teleportation is a protocol that allows one to send the state of a qubit $|\psi\rangle$ from Alice to Bob using only classical communication and pre-shared entangled qubits. The protocol is written

as the following circuit:
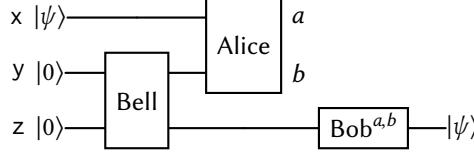
In our logic, the correctness of the teleportation program can be specified as follows:

$$\exists\, P : \text{frameable, prob 1.}\ \forall\, |\psi\rangle\,.$$

$$\big\{\, x \mapsto |\psi\rangle * y \mapsto |0\rangle * z \mapsto |0\rangle \,\big\}^{a,b}\ \text{Teleport}^{a,b}[x, y, z]\ \big\{\, z \mapsto |\psi\rangle * P \,\big\}.$$

The specification states that the final state of the qubit z will be the same as the initial state of the qubit x.

The first step of the protocol is to distribute the entangled qubits y and z to Alice and Bob. The specification of this preparation step is as follows:

$$\big\{\, y \mapsto |0\rangle * z \mapsto |0\rangle \,\big\}\ \text{Bell}[y, z]\ \big\{\, (y, z) \mapsto \tfrac{1}{\sqrt 2}(|00\rangle + |11\rangle) \,\big\}.$$

Then, Alice performs the following program to generate the classical bits a and b that will be sent to Bob:

$$\text{Alice}^{a,b}[x, y]\ \ =\ \ \text{[circuit]}\ \ =\ \ \text{CX}[x, y];\ \text{H}[x];\ a \leftarrow \text{M}_Z[x];\ b \leftarrow \text{M}_Z[y].$$
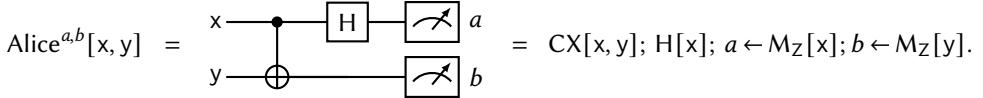
Finally, Bob receives the classical bits a and b from Alice, and performs the following program to recover the state $|\psi\rangle$:

$$\text{Bob}^{a,b}[z]\ \ =\ \ \text{[circuit]}\ \ =\ \ \text{if } b \text{ then } \text{X}[z];\ \text{if } a \text{ then } \text{Z}[z].$$

We prove the correctness of the protocol modularly, i.e., we prove the specifications of Alice and Bob separately, and then combine them to prove the specification of the whole teleportation program. We first prove the specification of Alice assuming the inputs are in the classical state $|xi\rangle$ for $x, i \in \{0, 1\}$:

$$\big\{\, (x, y) \mapsto |xi\rangle \,\big\}^{a,b}\ \text{Alice}^{a,b}[x, y]\ \Big\{\, \tfrac{1}{\sqrt 2} \bigoplus\nolimits^{a,b} (-1)^{x \wedge a}\, \delta_{x \vee i, b} \cdot (x, y) \mapsto |ab\rangle \,\Big\}.$$

This proof can be done symbolically as follows:

$$\big\{\, (x, y) \mapsto |xi\rangle \,\big\}^{a,b}\ \text{CX}[x, y]\ \big\{\, (x, y) \mapsto |x(x \veebar i)\rangle \,\big\}^{a,b}$$

$$\text{H}[x]\ \Big\{\, \tfrac{1}{\sqrt 2} \sum\nolimits_{k=0,1} (-1)^{x \wedge k} * (x, y) \mapsto |k(x \veebar i)\rangle \,\Big\}^{a,b}$$

$$a \leftarrow \text{M}_Z[x]\ \Big\{\, \tfrac{1}{\sqrt 2} \bigoplus\nolimits^{a} (-1)^{x \wedge a} * (x, y) \mapsto |a(x \veebar i)\rangle \,\Big\}^{b}$$

$$b \leftarrow \text{M}_Z[y]\ \Big\{\, \tfrac{1}{\sqrt 2} \bigoplus\nolimits^{a,b} (-1)^{x \wedge a} \cdot \delta_{(x \veebar i), b} * (x, y) \mapsto |ab\rangle \,\Big\}.$$

Note that we are also identifying the classical variables a and b with the boolean values they store. We can also prove the following specification of Bob in a straightforward way:

$$\big\{\, z \mapsto |i\rangle * a \mapsto a * b \mapsto x \veebar i \,\big\}\ \text{Bob}^{a,b}[z]\ \big\{\, (-1)^{x \wedge a}\, z \mapsto |x\rangle \,\big\}.$$

By combining the above specifications, we prove the following specification of the whole teleportation program for the classical input $|x\rangle$ for $x \in \{0, 1\}$:

$$\exists P : \text{frameable, prob 1.} \quad \big\{ \, \mathsf{x} \mapsto |x\rangle * \mathsf{y} \mapsto |0\rangle * \mathsf{z} \mapsto |0\rangle \, \big\}^{a,b} \; \text{Teleport}^{a,b}[\mathsf{x}, \mathsf{y}, \mathsf{z}] \; \big\{ \, \mathsf{z} \mapsto |x\rangle * P \, \big\}.$$

We can prove this by the following derivation:

$$\big\{ \mathsf{x} \mapsto |x\rangle \, * \, (\mathsf{y}, \mathsf{z}) \mapsto |00\rangle \big\}^{a,b}$$

$$\text{Bell}[\mathsf{y}, \mathsf{z}]$$

$$\big\{ \mathsf{x} \mapsto |x\rangle \, * \, (\mathsf{y}, \mathsf{z}) \mapsto \tfrac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \big\}^{a,b} \; \big\{ \tfrac{1}{\sqrt{2}} \sum_{i=0,1} \mathsf{x} \mapsto |x\rangle \, * \, (\mathsf{y}, \mathsf{z}) \mapsto |ii\rangle \big\}^{a,b}$$

$$\text{Alice}^{a,b}[\mathsf{x}, \mathsf{y}]$$

$$\Big\{ \tfrac{1}{2} \sum_{i=0,1} \bigoplus^{a,b} (-1)^{x \wedge a} \, \delta_{(x \veebar i),b} \cdot (\mathsf{x}, \mathsf{y}) \mapsto |ab\rangle \, * \, \mathsf{z} \mapsto |i\rangle \Big\}$$

$$\text{Bob}^{a,b}[\mathsf{z}]$$

$$\Big\{ \tfrac{1}{2} \sum_{i=0,1} \bigoplus^{a,b} \delta_{(x \veebar i),b} \cdot (\mathsf{x}, \mathsf{y}) \mapsto |ab\rangle \, * \, \mathsf{z} \mapsto |x\rangle \Big\}$$

$$\Big\{ \tfrac{1}{2} \bigoplus^{a,b} \sum_{i=0,1} \delta_{(x \veebar i),b} \cdot (\mathsf{x}, \mathsf{y}) \mapsto |ab\rangle \, * \, \mathsf{z} \mapsto |x\rangle \Big\}$$

$$\Big\{ \tfrac{1}{2} \bigoplus^{a,b} (\mathsf{x}, \mathsf{y}) \mapsto |ab\rangle \, * \, \mathsf{z} \mapsto |x\rangle \Big\} \; \Big\{ \mathsf{z} \mapsto |x\rangle \, * \, \bigoplus^{a,b} (\mathsf{x}, \mathsf{y}) \mapsto \tfrac{1}{2} |ab\rangle \Big\}.$$

Now we can set $P \triangleq \bigoplus^{a,b} (\mathsf{x}, \mathsf{y}) \mapsto \tfrac{1}{2} |ab\rangle$ because it is frameable and prob 1. In the general case, the input state of x can be any quantum state $|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle$. The Hoare triple we have just proved can be immediately generalized using HOARE-SCALE and HOARE-SUM:

$$\frac{\dfrac{\forall x. \; \big\{ \mathsf{x} \mapsto |x\rangle * (\mathsf{y}, \mathsf{z}) \mapsto |00\rangle \big\}^{a,b} \; \text{Teleport}^{a,b}[\mathsf{x}, \mathsf{y}, \mathsf{z}] \; \big\{ \mathsf{z} \mapsto |x\rangle * P \big\}}{\forall x. \; \big\{ \mathsf{x} \mapsto \alpha_x |x\rangle * (\mathsf{y}, \mathsf{z}) \mapsto |00\rangle \big\}^{a,b} \; \text{Teleport}^{a,b}[\mathsf{x}, \mathsf{y}, \mathsf{z}] \; \big\{ \mathsf{z} \mapsto \alpha_x |x\rangle * P \big\}} \; \text{HOARE-SCALE}}{\big\{ \mathsf{x} \mapsto |\psi\rangle * (\mathsf{y}, \mathsf{z}) \mapsto |00\rangle \big\}^{a,b} \; \text{Teleport}^{a,b}[\mathsf{x}, \mathsf{y}, \mathsf{z}] \; \big\{ \mathsf{z} \mapsto |\psi\rangle * P \big\}} \; \text{HOARE-SUM}$$

B.4 Lattice Surgery: Implementation of CNOT with Measurements

Here we give the details of §5.3.

Lattice surgery is a technique for fault-tolerant quantum computing that uses surface codes [Dennis et al. 2002]. In lattice surgery, the CNOT gate is implemented without using two-qubit gates, but using only two-qubit measurements, as shown in Fig. 2. The right-hand side circuit mCNOT implements the CNOT gate using two 2-qubit measurements, M_{XX} and M_{ZZ}, and some single-qubit gates with an auxiliary qubit y initialized to $|0\rangle$. The program $\text{mCNOT}^{\iota,\kappa,\lambda}[\mathsf{x}, \mathsf{y}, \mathsf{z}]$ for the circuit has been given in §5.3. We want to formally prove that this circuit indeed implements the CNOT gate. That is, we want to prove the following specification:

$$\exists P : \text{frameable, prob 1.} \; \forall |\psi\rangle \, .$$

$$\big\{ (\mathsf{x}, \mathsf{z}) \mapsto |\psi\rangle * \mathsf{y} \mapsto |0\rangle \big\}^{\iota,\kappa,\lambda} \; \text{mCNOT}^{\iota,\kappa,\lambda}[\mathsf{x}, \mathsf{y}, \mathsf{z}] \; \big\{ (\mathsf{x}, \mathsf{z}) \mapsto CX |\psi\rangle * P \big\}.$$

The specification of 2-qubit measurements M_{XX} and M_{ZZ} is given as follows.

$$\forall s \in \{+, -\}. \; \big\{ (\mathsf{x}, \mathsf{y}) \mapsto |ss\rangle \big\} \; M_{XX}^{a}[\mathsf{x}, \mathsf{y}] \; \big\{ (\mathsf{x}, \mathsf{y}) \mapsto |ss\rangle \, {}_0\oplus_1^{a} \, (\mathsf{x}, \mathsf{y}) \mapsto 0 \big\}$$

$$\forall s, t \in \{+, -\} \text{ s.t. } s \neq t. \; \big\{ (\mathsf{x}, \mathsf{y}) \mapsto |st\rangle \big\} \; M_{XX}^{a}[\mathsf{x}, \mathsf{y}] \; \big\{ (\mathsf{x}, \mathsf{y}) \mapsto 0 \, {}_0\oplus_1^{a} \, (\mathsf{x}, \mathsf{y}) \mapsto |st\rangle \big\}$$

$$\forall i \in \{0, 1\}. \; \big\{ (\mathsf{x}, \mathsf{y}) \mapsto |ii\rangle \big\} \; M_{ZZ}^{a}[\mathsf{x}, \mathsf{y}] \; \big\{ (\mathsf{x}, \mathsf{y}) \mapsto |ii\rangle \, {}_0\oplus_1^{a} \, (\mathsf{x}, \mathsf{y}) \mapsto 0 \big\}$$

$$\forall i, j \in \{0, 1\} \text{ s.t. } i \neq j. \; \big\{ (\mathsf{x}, \mathsf{y}) \mapsto |ij\rangle \big\} \; M_{ZZ}^{a}[\mathsf{x}, \mathsf{y}] \; \big\{ (\mathsf{x}, \mathsf{y}) \mapsto 0 \, {}_0\oplus_1^{a} \, (\mathsf{x}, \mathsf{y}) \mapsto |ij\rangle \big\}$$

By linearity, we can assume that the initial state of x and z is some disentangled state. Here, we take the initial state to be $(x, z) \mapsto |a\rangle\, (H\,|b\rangle)$ for $a, b \in \{0, 1\}$. The qubit z is not in the classical state $|0\rangle$ or $|1\rangle$, but we choose $|+\rangle \triangleq H\,|0\rangle$ and $|-\rangle \triangleq H\,|1\rangle$ for simplicity. Since $\langle|+\rangle, |-\rangle\rangle$ spans the whole 2-dimensional Hilbert space $\mathbb{C}^2$, this is enough to prove the correctness of the program. The derivation is as follows:

$$\left\{ x \mapsto |a\rangle \,*\, y \mapsto |0\rangle \,*\, z \mapsto H\,|b\rangle \right\}^{\iota, \kappa, \lambda}$$

$$\left\{ x \mapsto |a\rangle \,*\, \frac{1}{\sqrt{2}} \sum_c y \mapsto H\,|c\rangle \,*\, z \mapsto H\,|b\rangle \right\}^{\iota, \kappa, \lambda}$$

$$M_{XX}^\iota[y, z]$$

$$\left\{ x \mapsto |a\rangle \,*\, \frac{1}{\sqrt{2}} \bigoplus^\iota \sum_c \delta_{\iota, c \veebar b} \cdot y \mapsto H\,|c\rangle \,*\, z \mapsto H\,|b\rangle \right\}^{\kappa, \lambda}$$

$$\left\{ x \mapsto |a\rangle \,*\, \frac{1}{\sqrt{2}} \bigoplus^\iota y \mapsto H\,|b \veebar \iota\rangle \,*\, z \mapsto H\,|b\rangle \right\}^{\kappa, \lambda} \qquad (c := b \veebar \iota)$$

if ι then $Z[x]$

$$\left\{ \frac{1}{\sqrt{2}} \bigoplus^\iota (-1)^{a\iota} \cdot x \mapsto |a\rangle \,*\, y \mapsto H\,|b \veebar \iota\rangle \,*\, z \mapsto H\,|b\rangle \right\}^{\kappa, \lambda}$$

$$\left\{ \frac{1}{2} \bigoplus^\iota \sum_d (-1)^{a\iota + (b \veebar \iota)d} \cdot (x, y) \mapsto |ad\rangle \,*\, z \mapsto H\,|b\rangle \right\}^{\kappa, \lambda}$$

$$M_{ZZ}^\kappa[x, y]$$

$$\left\{ \frac{1}{2} \bigoplus^{\iota, \kappa} \sum_d \delta_{\kappa, a \veebar d} \cdot (-1)^{a\iota + (b \veebar \iota)d} \cdot (x, y) \mapsto |ad\rangle \,*\, z \mapsto H\,|b\rangle \right\}^{\lambda}$$

$$\left\{ \frac{1}{2} \bigoplus^{\iota, \kappa} (-1)^{a\iota + (b \veebar \iota)(a \veebar \kappa)} \cdot (x, y) \mapsto |a(a \veebar \kappa)\rangle \,*\, z \mapsto H\,|b\rangle \right\}^{\lambda} \qquad (d := a \veebar \kappa)$$

if κ then $X[z]$

$$\left\{ \frac{1}{2} \bigoplus^{\iota, \kappa} (-1)^{a\iota + (b \veebar \iota)(a \veebar \kappa) + b\kappa} \cdot (x, y) \mapsto |a(a \veebar \kappa)\rangle \,*\, z \mapsto H\,|b\rangle \right\}^{\lambda}$$

$$\left\{ \frac{1}{2} \bigoplus^{\iota, \kappa} (-1)^{ab + \iota\kappa} \cdot (x, y) \mapsto |a(a \veebar \kappa)\rangle \,*\, z \mapsto H\,|b\rangle \right\}^{\lambda}$$

$$H[y]$$

$$\left\{ \frac{1}{2} \bigoplus^{\iota, \kappa} (-1)^{ab + \iota\kappa} \cdot x \mapsto |a\rangle \,*\, y \mapsto H\,|a \veebar \kappa\rangle \,*\, z \mapsto H\,|b\rangle \right\}^{\lambda}$$

$$M_Z^\lambda[y]$$

$$\left\{ \frac{1}{2\sqrt{2}} \bigoplus^{\iota, \kappa, \lambda} (-1)^{ab + \iota\kappa + (a \veebar \kappa)\lambda} \cdot (x, y) \mapsto |a\lambda\rangle \,*\, z \mapsto H\,|b\rangle \right\}$$

if λ then $Z[x]$

$$\left\{ \frac{1}{2\sqrt{2}} \bigoplus^{\iota, \kappa, \lambda} (-1)^{ab + \iota\kappa + (a \veebar \kappa)\lambda + a\lambda} \cdot (x, y) \mapsto |a\lambda\rangle \,*\, z \mapsto H\,|b\rangle \right\}$$

$$\left\{ \frac{1}{2\sqrt{2}} \bigoplus^{\iota, \kappa, \lambda} (-1)^{ab + \iota\kappa + \kappa\lambda} \cdot (x, y) \mapsto |a\lambda\rangle \,*\, z \mapsto H\,|b\rangle \right\}$$

$$\left\{ (-1)^{ab} \frac{1}{2\sqrt{2}} \cdot (x, z) \mapsto |a\rangle \otimes H\,|b\rangle \,*\, \bigoplus^{\iota, \kappa, \lambda} (-1)^{\iota\kappa + \kappa\lambda} \cdot y \mapsto |\lambda\rangle \right\}$$

$$\left\{ (x, z) \mapsto CX(|a\rangle \otimes H|b\rangle) \; * \; \tfrac{1}{2\sqrt{2}} \bigoplus^{\iota, \kappa, \lambda} (-1)^{\iota\kappa + \kappa\lambda} \cdot y \mapsto |\lambda\rangle \right\}.$$

Although the above proof is a bit long, each step is straightforward. Note that at the second last step, we used the unframe rule BIGMIX-UNFRAME. This proves that the program behaves as the CNOT gate for the initial state $(x, z) \mapsto |a\rangle \otimes H|b\rangle$. Finally, we can set $P \triangleq \tfrac{1}{2\sqrt{2}} \bigoplus^{\iota, \kappa, \lambda} (-1)^{\iota\kappa + \kappa\lambda} \cdot y \mapsto |\lambda\rangle$ (which satisfies frameable and prob 1) and conclude the proof with the following derivation, generalizing the assertion to any initial state of x and z using HOARE-SUM and HOARE-SCALE:

$$\frac{\dfrac{\forall a, s. \left\{ (x, z) \mapsto |as\rangle * y \mapsto |0\rangle \right\}^{\iota, \kappa, \lambda} \; \text{mCNOT}^{\iota, \kappa, \lambda}[x, y, z] \; \left\{ (x, z) \mapsto CX|as\rangle * P \right\}}{\forall a, s. \left\{ (x, z) \mapsto \alpha_{a,s}|as\rangle * y \mapsto |0\rangle \right\}^{\iota, \kappa, \lambda} \; \text{mCNOT}^{\iota, \kappa, \lambda}[x, y, z] \; \left\{ (x, z) \mapsto \alpha_{a,s}CX|as\rangle * P \right\}} \text{HOARE-SCALE}}{\left\{ (x, z) \mapsto |\psi\rangle * y \mapsto |0\rangle \right\}^{\iota, \kappa, \lambda} \; \text{mCNOT}^{\iota, \kappa, \lambda}[x, y, z] \; \left\{ (x, z) \mapsto CX|\psi\rangle * P \right\}} \text{HOARE-SUM}$$

Here, a and s range over $\{0, 1\}$ and $\{+, -\}$, respectively. This works because we can decompose any vector $|\psi\rangle$ into $\sum_{a,s} \alpha_{a,s}|as\rangle$ for some coefficients $\alpha_{a,s} \in \mathbb{C}$.

B.5 Error Correction: Bit-Flip Code

Here we give the details of § 5.4.

Fault-tolerant quantum computation aims to protect quantum information from errors by introducing redundancy, and the error correction codes are a key component of this approach. For example, instead of representing a qubit state $|i\rangle$ (for $i \in \{0, 1\}$) using a single qubit, we can encode it as $|iii\rangle$ using three qubits. This duplication enables error detection and correction if one of the qubits flips due to noise.

More generally, such redundancy means that an abstract single-qubit state—referred to as a *logical qubit*—is encoded across multiple *physical qubits*. Such an encoding is called an *error correction code* and allows the system to tolerate certain classes of errors without compromising the logical information.

One of the simplest quantum error correction codes we consider in this section is the three-qubit bit-flip code. It encodes a logical qubit $\alpha|0\rangle + \beta|1\rangle$ as $\alpha|000\rangle + \beta|111\rangle$, and protects against a single bit-flip error, which is an error that flips the state of a qubit from $|0\rangle$ to $|1\rangle$ or vice versa. The error correction procedure can be expressed as the circuit in Fig. 21. Formally, BitEC can be defined as the following program:

$$\begin{aligned}
\text{BitEC}^{a,b}[x, y, z] \; \triangleq \; & CX[y, x]; \; M_Z^a[x]; \; CX[y, x]; \; CX[z, y]; \; M_Z^b[y]; \; CX[z, y]; \\
& \text{if } a \wedge \neg b \text{ then } X[x]; \; \text{if } a \wedge b \text{ then } X[y]; \; \text{if } \neg a \wedge b \text{ then } X[z].
\end{aligned}$$

The first line applies the CNOT gates and measurements are *error detection* steps. During this phase, we measure the effect of the errors, called the *syndrome*, by measuring the qubits y and z. The second line applies corrective operations based on the measurement results to *recover* the original logical qubit state. The correctness of this error-correction procedure can be specified as the following assertion, which holds for any $e_1, e_2, e_3 \in \{0, 1\}$ satisfying $e_1 + e_2 + e_3 \leq 1$:

$$\begin{aligned}
\forall \alpha, \beta. \quad & \left\{ (x, y, z) \mapsto (X^{e_1} X^{e_2} X^{e_3})(\alpha|000\rangle + \beta|111\rangle) \right\}^{a,b} \\
& \text{BitEC}^{a,b}[x, y, z] \quad \left\{ (x, y, z) \mapsto (\alpha|000\rangle + \beta|111\rangle) * P_{e_1, e_2, e_3}^{\text{BitEC}^{a,b}} \right\}.
\end{aligned}$$

Here, $P_{e_1, e_2, e_3}^{\text{BitEC}^{a,b}}$ is some SL assertion satisfying frameable and prob 1.

We can prove it by the following derivation for $i = 0, 1$:

$$\left\{ x \mapsto |i \veebar e_1\rangle * y \mapsto |i \veebar e_2\rangle * z \mapsto |i \veebar e_3\rangle \right\}^{a,b}$$

$$CX[x, y] \; \left\{ x \mapsto |i \veebar e_1\rangle * y \mapsto |e_1 \veebar e_2\rangle * z \mapsto |i \veebar e_3\rangle \right\}^{a,b}$$

$\mathsf{M}_Z^a[\mathsf{y}]$ $\left\{\bigoplus^a \delta_{a,e_1 \veebar e_2} \cdot \mathsf{x} \mapsto |i \veebar e_1\rangle * \mathsf{y} \mapsto |e_1 \veebar e_2\rangle * \mathsf{z} \mapsto |i \veebar e_3\rangle \right\}^{a,b}$

$\mathsf{CX}[\mathsf{x},\mathsf{y}]$ $\left\{\bigoplus^a \delta_{a,e_1 \veebar e_2} \cdot \mathsf{x} \mapsto |i \veebar e_1\rangle * \mathsf{y} \mapsto |i \veebar e_2\rangle * \mathsf{z} \mapsto |i \veebar e_3\rangle \right\}^a$

$\mathsf{CX}[\mathsf{y},\mathsf{z}]; \mathsf{M}_Z^b[\mathsf{z}]; \mathsf{CX}[\mathsf{y},\mathsf{z}]$

$\left\{\bigoplus^{a,b} \delta_{a,e_1 \veebar e_2}\delta_{b,e_2 \veebar e_3} \cdot \mathsf{x} \mapsto |i \veebar e_1\rangle * \mathsf{y} \mapsto |i \veebar e_2\rangle * \mathsf{z} \mapsto |i \veebar e_3\rangle \right\}$

if $a \wedge \neg b$ then $\mathsf{X}[\mathsf{x}]$; if $a \wedge b$ then $\mathsf{X}[\mathsf{y}]$; if $\neg a \wedge b$ then $\mathsf{X}[\mathsf{z}]$;

$\left\{\bigoplus^{a,b} \delta_{a,e_1 \veebar e_2}\delta_{b,e_2 \veebar e_3} \cdot \mathsf{x} \mapsto |i \veebar e_1 \veebar a \cdot \neg b\rangle * \mathsf{y} \mapsto |i \veebar e_2 \veebar ab\rangle * \mathsf{z} \mapsto |i \veebar e_3 \veebar \neg a \cdot b\rangle \right\}$

$\left\{P_{e_1,e_2,e_3}^{\mathrm{BitEC}^{a,b}} * \mathsf{x} \mapsto |i \veebar e_1 \veebar (e_1 \veebar e_2)(1 \veebar e_2 \veebar e_3)\rangle * \mathsf{y} \mapsto |i \veebar e_2 \veebar (e_1 \veebar e_2)(e_2 \veebar e_3)\rangle * \right.$
$\left. \mathsf{z} \mapsto |i \veebar e_3 \veebar (1 \veebar e_1 \veebar e_2)(e_2 \veebar e_3)\rangle \right\}.$

Here, we let $P_{e_1,e_2,e_3}^{\mathrm{BitEC}^{a,b}} \triangleq \bigoplus^{a,b} \delta_{a,e_1 \veebar e_2}\delta_{b,e_2 \veebar e_3}$, which does not depend on i. Thanks to $e_1 + e_2 + e_3 \leq 1$, with an easy bit manipulation we can show the following (we use $e_i^2 = e_i$ and $e_i e_j = 0$ under $i \neq j$):

$$e_1 \veebar (e_1 \veebar e_2)(1 \veebar e_2 \veebar e_3) = e_1 \veebar e_1 \veebar e_2 \veebar e_2 = 0$$
$$e_2 \veebar (e_1 \veebar e_2)(e_2 \veebar e_3) = e_2 \veebar e_2 = 0$$
$$e_3 \veebar (1 \veebar e_1 \veebar e_2)(e_2 \veebar e_3) = e_3 \veebar e_2 \veebar e_3 \veebar e_2 = 0.$$

Therefore, the postcondition can be simplified to $P_{e_1,e_2,e_3}^{\mathrm{BitEC}^{a,b}} * (\mathsf{x},\mathsf{y},\mathsf{z}) \mapsto |iii\rangle$.

Finally, we can use HOARE-SUM and HOARE-SCALE to generalize this proof to any initial state $\alpha |000\rangle + \beta |111\rangle$ (we here abbreviate $\mathrm{BitEC}^{a,b}[\mathsf{x},\mathsf{y},\mathsf{z}]$ as BitEC and $P_{e_1,e_2,e_3}^{\mathrm{BitEC}^{a,b}}$ as P^{BitEC}):

$$\frac{\dfrac{\forall i \in \{0,1\}. \left\{(\mathsf{x},\mathsf{y},\mathsf{z}) \mapsto (\mathsf{X}^{e_1}\mathsf{X}^{e_2}\mathsf{X}^{e_3})|iii\rangle \right\}^{a,b} \mathrm{BitEC} \left\{P^{\mathrm{BitEC}} * (\mathsf{x},\mathsf{y},\mathsf{z}) \mapsto |iii\rangle \right\}}{\left\{(\mathsf{x},\mathsf{y},\mathsf{z}) \mapsto (\mathsf{X}^{e_1}\mathsf{X}^{e_2}\mathsf{X}^{e_3})(\alpha|000\rangle) \right\}^{a,b} \mathrm{BitEC} \left\{P^{\mathrm{BitEC}} * (\mathsf{x},\mathsf{y},\mathsf{z}) \mapsto \alpha|000\rangle \right\}}\text{ HOARE-SCALE} \quad \left\{(\mathsf{x},\mathsf{y},\mathsf{z}) \mapsto (\mathsf{X}^{e_1}\mathsf{X}^{e_2}\mathsf{X}^{e_3})(\beta|111\rangle) \right\}^{a,b} \mathrm{BitEC} \left\{P^{\mathrm{BitEC}} * (\mathsf{x},\mathsf{y},\mathsf{z}) \mapsto \beta|111\rangle \right\}}{\left\{(\mathsf{x},\mathsf{y},\mathsf{z}) \mapsto (\mathsf{X}^{e_1}\mathsf{X}^{e_2}\mathsf{X}^{e_3})(\alpha|000\rangle + \beta|111\rangle) \right\}^{a,b} \mathrm{BitEC} \left\{P^{\mathrm{BitEC}} * (\mathsf{x},\mathsf{y},\mathsf{z}) \mapsto \alpha|000\rangle + \beta|111\rangle \right\}}\text{ HOARE-SUM}$$

Our proof of the correctness of the bit-flip code has some similarities with verification of quantum error correction codes using symbolic execution as in [Fang and Ying 2024], though our approach is fundamentally different. Their approach uses density matrices and stabilizer codes, while our approach uses vector spaces and separation logic.

There are other kinds of errors that can occur in quantum computation, such as *phase-flip errors*. A phase-flip error negates the relative phase between $|0\rangle$ and $|1\rangle$, i.e., applies the Z gate, turning $|1\rangle$ to $-|1\rangle$ but $|0\rangle$ to $|0\rangle$. In other words, it transforms a qubit state $|+\rangle$ to $|-\rangle$ and vice versa. In order to correct such phase-flip errors, we need a different error correction code. However, this can be done easily by using the bit-flip code in a clever way. Since a bit-flip code encodes a logical qubit $|0\rangle$ as $|000\rangle$ and $|1\rangle$ as $|111\rangle$ to protect against a bit-flip error, we can instead take three copies with respect to the X basis $\langle|+\rangle,|-\rangle\rangle$ rather than the Z basis $\langle|0\rangle,|1\rangle\rangle$, i.e., encode a qubit $|+\rangle$ as $|+++\rangle$ and $|-\rangle$ as $|---\rangle$. As the Hadamard gate H transforms the X basis to the Z basis and vice versa, we can derive the phase-flip code error correction procedure PhaseEC simply by putting H gates before and after the phase-flip code error correction procedure BitEC.

The phase-flip code error correction procedure PhaseEC can be derived from BitEC as follows:
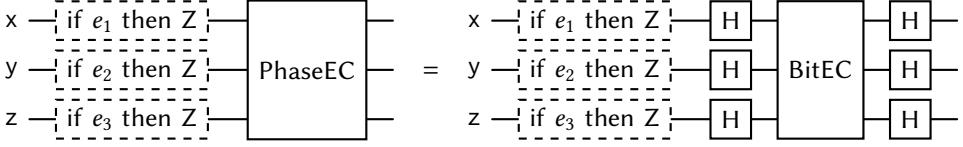
Formally, PhaseEC is defined as the following program:

$$\mathsf{PhaseEC}^{a,b}[x, y, z] \triangleq \mathsf{H}[x]; \mathsf{H}[y]; \mathsf{H}[z]; \mathsf{BitEC}^{a,b}[x, y, z]; \mathsf{H}[x]; \mathsf{H}[y]; \mathsf{H}[z].$$

We can immediately derive the following specification of the procedure PhaseEC from the specification of BitEC, for any $e_1, e_2, e_3 \in \{0, 1\}$ such that $e_1 + e_2 + e_3 \leq 1$:

$$\forall \alpha, \beta. \quad \left\{ (x, y, z) \mapsto \mathsf{Z}^{e_1} \mathsf{Z}^{e_2} \mathsf{Z}^{e_3}(\alpha \left|{+}{+}{+}\right\rangle + \beta \left|{-}{-}{-}\right\rangle) \right\}^{a,b}$$
$$\mathsf{PhaseEC}^{a,b}[x, y, z] \quad \left\{ (x, y, z) \mapsto (\alpha \left|{+}{+}{+}\right\rangle + \beta \left|{-}{-}{-}\right\rangle) * P_{e_1,e_2,e_3}^{\mathsf{BitEC}^{a,b}} \right\}.$$

B.6 Error Correction: Shor Code

Here we give the details of § 5.5.

Now, we can combine the bit-flip code and the phase-flip code to construct a universal quantum error correction code capable of correcting any single-qubit error. This construction is known as the *Shor code*, which encodes a single logical qubit into 9 physical qubits. The Shor code achieves this by nesting two types of codes: we first apply a phase-flip code to the logical qubit, and then apply a bit-flip code to each of the resulting qubits. Concretely, the Shor code encodes the logical qubit state $\alpha \left|0\right\rangle + \beta \left|1\right\rangle$ into

$$\alpha \left|0_L 0_L 0_L\right\rangle + \beta \left|1_L 1_L 1_L\right\rangle,$$

where $\left|0_L\right\rangle$ and $\left|1_L\right\rangle$ are phase-flip-encoded states. In this way, the Shor code uses a bit-flip code over logical qubits that have already been protected against phase-flip errors, ensuring robustness against both bit-flip and phase-flip errors.

Such codes that can correct both bit-flip and phase-flip errors are universal in the sense that they can correct any single-qubit unitary error. The sketch of the intuitive proof of this is as follows. Because the Pauli gates id, X, Z and $\mathsf{Y} = -\sqrt{-1}\,\mathsf{ZX}$ form an orthogonal basis for the whole linear space of complex 2×2 matrices, any single-qubit unitary error U can be decomposed into a linear combination of id, X, Z and ZX, with a coefficient vector of norm 1. So we can write $U = \lambda_0 \mathsf{id} + \lambda_1 \mathsf{X} + \lambda_2 \mathsf{Z} + \lambda_3 \mathsf{ZX}$ for some $(\lambda_i)_{i=0}^{3}$ such that $\sum_{i=0}^{3} |\lambda_i|^2 = 1$. Note that a state $\left|\psi\right\rangle$ with the error U applied, $U \left|\psi\right\rangle$, can be expressed as

$$\lambda_0 \left|\psi\right\rangle + \lambda_1 \mathsf{X} \left|\psi\right\rangle + \lambda_2 \mathsf{Z} \left|\psi\right\rangle + \lambda_3 \mathsf{ZX} \left|\psi\right\rangle.$$

By linearity of quantum computation, it suffices to show that the matrices X, Z and ZX, can be corrected by the Shor code. Perhaps surprisingly, this is immediate, because PhaseEC and BitEC$_L$ are designed to correct X and Z errors, respectively.

The Shor code is a perfect example to demonstrate the power of our separation logic. The error correction circuit ShorEC for the Shor code is given in Fig. 22 and the formal definition has been given in § 5.5. As we can see, the whole circuit has 9 qubits, meaning the dimension of the Hilbert space is $2^9 = 512$. However, it can be decomposed into smaller modules, which can be verified independently using separation logic. This not only makes the verification process more manageable, it also lets us use our linear-combination rule to simplify the proof, since the universality of the Shor code is based on the linearity of quantum computation as sketched above.
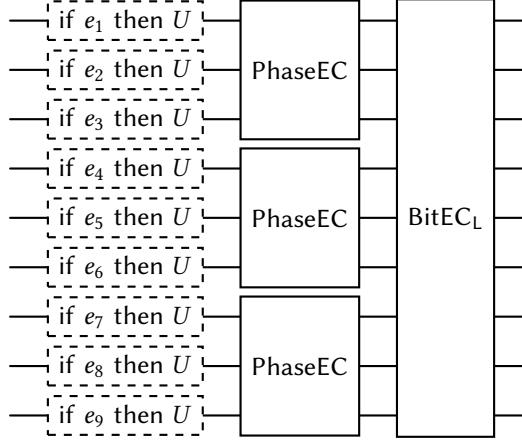
Fig. 22. Error correction circuit for the Shor code.

Let us first explain how the last component BitEC_L is constructed. This program is essentially the same as the 3-qubit bit-flip code BitEC we have already verified, but with each physical qubit replaced by a logical qubit encoded by the phase-flip code. Since we used X and CNOT gates and a measurement to implement the bit-flip code, we first implement these gates for the phase-flip code. That is, we verify the following assertions for programs X_L, CX_L and $\text{M}_\text{Z}^\text{L}$ to use them as building blocks for the error correction of the bit-flip code.

$$\exists\, Q : \text{frameable, prob } 1. \ \ \forall i \in \{0,1\}. \ \left\{ \bar{\text{x}} \mapsto |i_\text{L}\rangle \right\} \text{X}_\text{L}[\bar{\text{x}}] \left\{ \bar{\text{x}} \mapsto |\neg i_\text{L}\rangle * Q \right\}$$

$$\exists\, Q : \text{frameable, prob } 1. \ \ \forall i,j \in \{0,1\}. \ \left\{ (\bar{\text{x}},\bar{\text{y}}) \mapsto |i_\text{L} j_\text{L}\rangle \right\} \text{CX}_\text{L}[\bar{\text{x}},\bar{\text{y}}] \left\{ (\bar{\text{x}},\bar{\text{y}}) \mapsto |i_\text{L}(i \veebar j)_\text{L}\rangle * Q \right\}$$

$$\exists\, Q : \text{frameable, prob } 1. \ \left\{ \bar{\text{x}} \mapsto |0_\text{L}\rangle \right\}^a \text{M}_\text{Z}^{\text{L}a}[\bar{\text{x}}] \left\{ \left(\bar{\text{x}} \mapsto |0_\text{L}\rangle \ _0\oplus_1^a \ \bar{\text{x}} \mapsto 0 \right) * Q \right\}$$

$$\exists\, Q : \text{frameable, prob } 1. \ \left\{ \bar{\text{x}} \mapsto |1_\text{L}\rangle \right\}^a \text{M}_\text{Z}^{\text{L}a}[\bar{\text{x}}] \left\{ \left(\bar{\text{x}} \mapsto 0 \ _0\oplus_1^a \ \bar{\text{x}} \mapsto |0_\text{L}\rangle \right) * Q \right\}$$

Notably, we can abstract the factor Q by the condition Q : frameable, prob 1. Then, we can replace the basic gates X, CX and M_Z in BitEC with the above programs to implement BitEC_L. More concretely, BitEC_L is defined as follows:

$$\text{BitEC}_\text{L}^{a,b}[\bar{\text{x}},\bar{\text{y}},\bar{\text{z}}] \ \triangleq \ \text{CX}_\text{L}[\bar{\text{y}},\bar{\text{x}}]; \ \text{M}_\text{Z}^{\text{L}a}[\bar{\text{x}}]; \ \text{CX}_\text{L}[\bar{\text{y}},\bar{\text{x}}]; \ \text{CX}_\text{L}[\bar{\text{z}},\bar{\text{y}}]; \ \text{M}_\text{Z}^{\text{L}b}[\bar{\text{y}}]; \ \text{CX}_\text{L}[\bar{\text{z}},\bar{\text{y}}];$$
$$\text{if } a \wedge \neg b \text{ then } \text{X}_\text{L}[\bar{\text{x}}]; \ \text{if } a \wedge b \text{ then } \text{X}_\text{L}[\bar{\text{y}}]; \ \text{if } \neg a \wedge b \text{ then } \text{X}_\text{L}[\bar{\text{z}}].$$

We can also verify the correctness of BitEC_L exactly as we have done for the normal bit-flip code:

$$\exists\, Q : \text{frameable, prob } 1. \ \ \forall e_1, e_2, e_3 \text{ s.t. } e_1 + e_2 + e_3 \leq 1. \ \ \forall \alpha, \beta.$$
$$\left\{ (\bar{\text{x}},\bar{\text{y}},\bar{\text{z}}) \mapsto (\text{X}_\text{L}^{e_1}\text{X}_\text{L}^{e_2}\text{X}_\text{L}^{e_3})(\alpha\,|0_\text{L}0_\text{L}0_\text{L}\rangle + \beta\,|1_\text{L}1_\text{L}1_\text{L}\rangle) \right\}^{a,b}$$
$$\text{BitEC}_\text{L}^{a,b}[\bar{\text{x}},\bar{\text{y}},\bar{\text{z}}] \ \ \left\{ (\bar{\text{x}},\bar{\text{y}},\bar{\text{z}}) \mapsto (\alpha\,|0_\text{L}0_\text{L}0_\text{L}\rangle + \beta\,|1_\text{L}1_\text{L}1_\text{L}\rangle) * P_{e_1,e_2,e_3}^{\text{BitEC}^{a,b}} * Q \right\}.$$

This procedure of lifting the proof of correctness of the 3-qubit bit-flip code to the 9-qubit version BitEC_L is not completely automatic. However, we can easily redo the proof of correctness of a generalized bit-flip code implemented on top of any logical qubit encoded by an arbitrary code. This allows us to reuse the specification of the bit-flip code as a building block in the verification of any program that uses it. In this way, we can still avoid verifying the correctness of the bit-flip code every time we use it.

Finally, we verify the correctness of the Shor code. The specification of the correctness is similar to the ones we have seen (we omit classical variables for simplicity):

$$\exists P : \text{frameable, prob 1.} \ \forall \alpha, \beta. \ \ \left\{ \ (\bar{x}, \bar{y}, \bar{z}) \mapsto (U^{e_1} \otimes \cdots \otimes U^{e_9})(\alpha \, |0_L 0_L 0_L\rangle + \beta \, |1_L 1_L 1_L\rangle) \ \right\}$$
$$\text{ShorEC}[\bar{x}, \bar{y}, \bar{z}] \ \ \left\{ \ (\bar{x}, \bar{y}, \bar{z}) \mapsto (\alpha \, |0_L 0_L 0_L\rangle + \beta \, |1_L 1_L 1_L\rangle) * P \ \right\}.$$

This holds for any single-qubit unitary error U and any $e_1, \ldots, e_9 \in \{0, 1\}$ under $\sum_{i=1}^{9} e_i = 1$.[20]

To simplify the proof, we first assume the case where $U = Z^a X^b$ for $a, b \in \{0, 1\}$. We can derive the following on $\text{PhaseEC}[\bar{x}]$.

$$\frac{\forall c \in \{0,1\}. \ \left\{ \bar{x} \mapsto (\bigotimes_{i=1}^{3} Z^{ae_i} X^{be_i}) \, |c_L c_L c_L\rangle \right\} \quad \text{PhaseEC}[\bar{x}] \quad \left\{ \bar{x} \mapsto (\bigotimes_{i=1}^{3} X^{be_i}) \, |c_L c_L c_L\rangle * P_{ae_1, ae_2, ae_3}^{\text{BitEC}} \right\}}{\dfrac{\forall c \in \{0,1\}. \ \left\{ (\bar{x}, \bar{y}, \bar{z}) \mapsto (\bigotimes_{i=1}^{9} Z^{ae_i} X^{be_i}) \, |c_L c_L c_L\rangle \right\} \quad \text{PhaseEC}[\bar{x}] \quad \left\{ (\bar{x}, \bar{y}, \bar{z}) \mapsto (\bigotimes_{i=1}^{3} X^{be_i} \otimes \bigotimes_{i=4}^{9} Z^{ae_i} X^{be_i}) \, |c_L c_L c_L\rangle * P_{ae_1, ae_2, ae_3}^{\text{BitEC}} \right\}}{\left\{ (\bar{x}, \bar{y}, \bar{z}) \mapsto (\bigotimes_{i=1}^{9} Z^{ae_i} X^{be_i})(\alpha \, |0_L 0_L 0_L\rangle + \beta \, |1_L 1_L 1_L\rangle) \right\} \quad \text{PhaseEC}[\bar{x}] \quad \left\{ (\bar{x}, \bar{y}, \bar{z}) \mapsto (\bigotimes_{i=1}^{3} X^{be_i} \otimes \bigotimes_{i=4}^{9} Z^{ae_i} X^{be_i})(\alpha \, |0_L 0_L 0_L\rangle + \beta \, |1_L 1_L 1_L\rangle) * P_{ae_1, ae_2, ae_3}^{\text{BitEC}} \right\}} \text{ HOARE-SUM}} \text{ HOARE-FRAME}$$

Here, we used $ae_1 + ae_2 + ae_3 \leq 1$. Using this kind of derivation for PhaseEC three times and the specification of the enriched bit-flip code BitEC, we can derive the following for any $\alpha, \beta \in \mathbb{C}$, letting $|\psi_L\rangle \triangleq \alpha \, |0_L 0_L 0_L\rangle + \beta \, |1_L 1_L 1_L\rangle$:

$$\left\{ (\bar{x}, \bar{y}, \bar{z}) \mapsto (\bigotimes_{i=1}^{9} Z^{ae_i} X^{be_i}) \, |\psi_L\rangle \right\}$$

$$\text{PhaseEC}[\bar{x}]$$

$$\left\{ (\bar{x}, \bar{y}, \bar{z}) \mapsto (\bigotimes_{i=1}^{3} X^{be_i} \otimes \bigotimes_{i=4}^{9} Z^{ae_i} X^{be_i}) \, |\psi_L\rangle * P_a^{(1)} \right\}$$

$$\text{PhaseEC}[\bar{y}]$$

$$\left\{ (\bar{x}, \bar{y}, \bar{z}) \mapsto (\bigotimes_{i=1}^{6} X^{be_i} \otimes \bigotimes_{i=7}^{9} Z^{ae_i} X^{be_i}) \, |\psi_L\rangle * P_a^{(2)} \right\}$$

$$\text{PhaseEC}[\bar{z}]$$

$$\left\{ (\bar{x}, \bar{y}, \bar{z}) \mapsto (\bigotimes_{i=1}^{9} X^{be_i}) \, |\psi_L\rangle * P_a^{(3)} \right\}$$

$$\left\{ (\bar{x}, \bar{y}, \bar{z}) \mapsto (X_L^{b(e_1 \vee e_2 \vee e_3)} \otimes X_L^{b(e_4 \vee e_5 \vee e_6)} \otimes X_L^{b(e_7 \vee e_8 \vee e_9)}) \, |\psi_L\rangle * P_a^{(3)} \right\}$$

$$\text{BitEC}_L[\bar{x}, \bar{y}, \bar{z}]$$

$$\left\{ (\bar{x}, \bar{y}, \bar{z}) \mapsto |\psi_L\rangle * P_{a,b}^{(4)} \right\}$$

Note that every $P^{(k)}$ is a global phase guard that satisfies $P^{(k)}$: frameable, thanks to the fact that the separating conjunction of frameable assertions is frameable (Fig. 18). Importantly, each $P^{(k)}$ depends on the error parameters a, b (and each e_i), but it does not depend on the initial state $|\psi_L\rangle$. Finally, we can use the linear combination rule to derive the final specification of the Shor code. Let a single-qubit error U be written as $\sum_{a,b \in \{0,1\}} \lambda_{a,b} Z^a X^b$ for $(\lambda_{a,b})_{a,b}$ such that $\sum_{a,b \in \{0,1\}} |\lambda_{a,b}|^2 = 1$. Then we have the following derivation (we here abbreviate $\text{ShorEC}[\bar{x}, \bar{y}, \bar{z}]$ as ShorEC):

$$\frac{\forall a, b. \ \left\{ (\bar{x}, \bar{y}, \bar{z}) \mapsto (\bigotimes_{i=1}^{9} Z^{ae_i} X^{be_i}) \, |\psi_L\rangle \right\} \ \text{ShorEC} \ \left\{ (\bar{x}, \bar{y}, \bar{z}) \mapsto |\psi_L\rangle * P_{a,b}^{(4)} \right\}}{\dfrac{\forall a, b. \ \left\{ (\bar{x}, \bar{y}, \bar{z}) \mapsto (\bigotimes_{i=1}^{9} Z^{ae_i} X^{be_i}) \lambda_{a,b} \, |\psi_L\rangle \right\} \ \text{ShorEC} \ \left\{ (\bar{x}, \bar{y}, \bar{z}) \mapsto |\psi_L\rangle * \lambda_{a,b} P_{a,b}^{(4)} \right\}}{\left\{ (\bar{x}, \bar{y}, \bar{z}) \mapsto \sum_{a,b} (\bigotimes_{i=1}^{9} Z^{ae_i} X^{be_i}) \lambda_{a,b} \, |\psi_L\rangle \right\} \ \text{ShorEC} \ \left\{ (\bar{x}, \bar{y}, \bar{z}) \mapsto |\psi_L\rangle * \sum_{a,b} \lambda_{a,b} P_{a,b}^{(4)} \right\}} \text{ HOARE-SUM}} \text{ HOARE-SCALE}$$

---

[20] We can safely exclude the case where $e_1 = \cdots = e_9 = 0$, because we can set $U = \text{id}$ to model the error-free situation.

The precondition can be rewritten as $\sum_{a,b}(\bigotimes_{i=1}^{9} Z^{ae_i}X^{be_i})\lambda_{a,b}\,|\psi_L\rangle = (\bigotimes_{i=1}^{9} U^{e_i})\,|\psi_L\rangle$. The frameability of $P \triangleq \sum_{a,b} \lambda_{a,b}P_{a,b}^{(4)}$ follows from the construction.

Finally, we prove P: prob 1.[21] Since the coefficients $\lambda_{a,b}$ satisfy $\sum_{a,b\in\{0,1\}}|\lambda_{a,b}|^2 = 1$, it suffices to prove $P_{a,b}^{(4)}$: prob 1 and that the assertions $(P_{a,b}^{(4)})_{a,b\in\{0,1\}}$ are orthogonal. To begin with, $P_{a,b}^{(4)}$ has the following form:

$$P_{e,e',e''}^{\mathsf{BitEC}^{\kappa,\lambda}} \triangleq \bigoplus^{\kappa,\lambda} \delta_{\kappa,e\veebar e'}\delta_{\lambda,e'\veebar e''}$$

$$P_{a,b}^{(4)} \triangleq P_{ae_1,ae_2,ae_3}^{\mathsf{BitEC}^{\kappa_1,\lambda_1}} * P_{ae_4,ae_5,ae_6}^{\mathsf{BitEC}^{\kappa_2,\lambda_2}} * P_{ae_7,ae_8,ae_9}^{\mathsf{BitEC}^{\kappa_3,\lambda_3}} * P_{b(e_1\vee e_2\vee e_3),b(e_4\vee e_5\vee e_6),b(e_7\vee e_8\vee e_9)}^{\mathsf{BitEC}^{\kappa_4,\lambda_4}} * Q.$$

Here, Q is some proposition satisfying frameable and prob 1. We can easily prove $P_{a,b}^{(4)}$: prob 1. Now we prove the orthogonality of $(P_{a,b}^{(4)})_{a,b\in\{0,1\}}$. We can easily prove the orthogonality $P_{\bar{e}}^{\mathsf{BitEC}^{\kappa,\lambda}} \perp P_{\bar{e}'}^{\mathsf{BitEC}^{\kappa,\lambda}}$ for distinct $(\bar{e}), (\bar{e}') \in \{0,1\}^3$. So, for any two distinct parameters $(a,b), (a',b') \in \{0,1\}^2$, at least one of the first four conjuncts in $P^{(4)}$ is orthogonal.[22] Therefore, by the rule ORTH-FRAME in § A.7, the set $\{P_{a,b}^{(4)}\}_{a,b\in\{0,1\}}$ is proved orthogonal.

B.7 Probabilistic Choice and Almost Sure Termination

Here we give the details of § 5.6.

Recall the repeat-until-success program we considered:

$$\mathsf{cointoss}_p' \triangleq \text{ while } a \text{ do } \left(\mathsf{coin}_p^a; c \leftarrow c+1\right) \qquad \mathsf{cointoss}_p \triangleq \mathsf{coin}_p^a; \mathsf{cointoss}_p'.$$

We can prove the following specification for any probability $p \in (0,1]$, guaranteeing almost sure termination:

$$\exists P: \text{frameable, prob 1}. \ \left\{ c \mapsto 0 \right\}^a \mathsf{cointoss}_p \left\{ a \mapsto 0 * P \right\}.$$

Let $P_n \triangleq \sqrt{(1-p)^n} \cdot (a \mapsto 0 \oplus_p a \mapsto 1) * c \mapsto n$, $Q_n \triangleq \sqrt{(1-p)^n} \cdot a \mapsto 0 * c \mapsto n$, and $R_n \triangleq \sqrt{(1-p)^{n+1}} \cdot a \mapsto 1 * c \mapsto n$ for $n \in \mathbb{N}$. Also, we set $P \triangleq \bigoplus_{n\in\mathbb{N}}^c \sqrt{(1-p)^n p}$. Then we can conclude the proof by the following derivation:

$$\cfrac{\{c \mapsto 0\}^a \mathsf{coin}_p^a \{P_0\} \quad \cfrac{\cfrac{\forall n. \{P_n\}\, a\, \left\{\uparrow 0 * Q_n \oplus \uparrow 1 * R_n\right\} \quad \forall n. \{R_n\}\, \mathsf{coin}_p^a; c \leftarrow c+1\, \{P_{n+1}\}}{\{P_0\}\, P_0\, \{\bigoplus_{n\in\mathbb{N}}^c \sqrt{(1-p)^n p} * a \mapsto 0\}}\text{ HOARE-WHILE}}{\cfrac{\{P_0\}\, \mathsf{cointoss}_p'\, \{a \mapsto 0 * P\}}{}}\text{ BIGMIX-UNFRAME}}{\{c \mapsto 0\}^a \mathsf{cointoss}_p \{a \mapsto 0 * P\}}$$

Notably, P: frameable, prob 1 holds. The counter c is crucial for the unambiguity P: unambig. The probability judgment P: prob 1 holds because $\sum_{n\in\mathbb{N}}(1-p)^n p = 1$.

---

[21] This is quite trivial if we know that ShorEC is loop-free, because its termination probability is clearly 1. However, here we prove P: prob 1 even without depending on that knowledge, leveraging orthogonality.

[22] Recall that we assume $e_1 + \cdots + e_9 = 1$. So exactly one of $e_1, \ldots, e_9$ is non-zero. Also, exactly one of $e_1 \vee e_2 \vee e_3$, $e_4 \vee e_5 \vee e_6$ and $e_7 \vee e_8 \vee e_9$ is non-zero.