Beyond Separation: Toward a Specification Language for Modular Reasoning about Quantum Programs

Kartik Singhal [1] Robert Rand [1] Matthew Amy [2]

[1]University of Chicago, USA [2]Simon Fraser University, Canada

Introduction

Instead of rushing to adapt classical reasoning techniques such as separation logic to the quantum setting, we should carefully consider: what is so unique about quantum computation?

Separation Logic (SL) was famously invented to reason about pointer aliasing, there is no quantum analog of aliasing (except in high-level languages like Q# [1]). Yet, there are efforts like those by Zhou et al. [2] and Le et al. [3] that introduce quantum separation logics (QSLs).

The fundamental problem hindering the scalability of Hoare logics in the quantum domain is modular reasoning about *entangled*, rather than separable, states.

We argue that separation logic may not be the right tool for reasoning about quantum programs and, with examples, make the case for focusing more on a programmer-friendly specification language inspired by Unruh [4].

Why does Separation Logic work in the classical setting?

Pym, Spring, and O'Hearn [5] suggest that SL is successful because it offers:

- a useful conceptual model of memory
- a logical model and a specification language for true statements about memory
- a productive overlap of these models—conceptual and logical
- the separating conjunction ($*$) enabling the frame rule for local reasoning
- scalable pre- and postconditions.

Existing works only provide two of these five features: a logical model and separating conjunction along with a frame rule. The logical model arguably does not serve as a reasonable specification language [6], and the frame rules are not strong enough. Further, unlike in the classical setting, there is no agreement on a conceptual model of quantum memory.

There is a lot more to do before SL can be considered a viable reasoning technique for quantum computation.

Questions we should ask about QSL

- What do we gain with SL that quantum Hoare logics do not provide?
- SL is an extension of classical Hoare logic, does the QSL extend an existing QHL?
- What is the memory model under consideration? How do we tailor a QSL to that model?
- What is the quantum equivalent of a separating implication, $-\!\!*$ (magic wand)?

Specifying a Partial Bell State

The following small footprint specification holds for a Bell state preparation program:

$$\{\, \mathtt{a}{:}|0\rangle,\ \mathtt{b}{:}|0\rangle \,\}\ \mathtt{bell00(a,b)}\ \{\, (\mathtt{a},\mathtt{b}){:}|\Phi^+\rangle \,\}$$

where $|\Phi^+\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$. Note that the postcondition has to specify the joint state of a and b as they are entangled. What if we wanted to specify the state associated with just b? QSLs do not provide an answer!

Purification for modular reasoning

The principle of local reasoning requires that the *specification and proof concentrate on only those cells in memory that a program accesses*. If qubits are considered as cells, this notion transfers directly to the quantum setting.

But modular reasoning about entangled states required a quantum-information theoretic trick – *purification* or "Going to the Church of the Larger Hilbert Space".

With this trick, we can specify pure states associated with quantum variables even if they are in a mixed entangled state. Unruh's ghost variables encode this idea in a program logic.

Using a fresh ghost variable, e, we can provide a small footprint specification for b as $(e, b) : |\Phi^+\rangle$ in any context.

Specifying Modular Teleportation

Consider the following spec:

$$\{\, (\mathtt{m},E){:}|\psi\rangle,\ \mathtt{b}{:}|0\rangle \,\}$$
$$\mathtt{teleport(m,b)}$$
$$\{\, \mathtt{class(m)},\ (\mathtt{b},E){:}|\psi\rangle \,\}$$

The pre- and postconditions completely specify this program:

- it can teleport not just a pure but also an entangled state; the ghost variable, E, encodes the rest of the system that message m may be part of.
- it requires Bob's qubit b to be in state $|0\rangle$.
- the message qubit is measured (is in a classical state) at the end of the program.
- message qubit is swapped by b in the original entangled system.

The teleport program is itself modular:

```
operation teleport(m,b) {
    new a;
    bell00(a,b);
    let resultbits = encode(m,a);
    decode(b,resultbits);
}
```

and requires careful specification of its parts.

Specifying the Modules

Let us see the code first:

```
operation bell00(a,b) {
    H(a);
    CNOT(a,b);
}
operation encode(m,a) {
    CNOT(m,a);
    H(m);
    return (M(m),M(a));
}
operation decode(b,(m1,m2)) {
    if m1 then Z(b);
    if m2 then X(b);
}
```

`bell00` was specified earlier, the others are:

$$\{\, (\mathtt{m},E){:}\alpha|0\rangle|\psi_1\rangle+\beta|1\rangle|\psi_2\rangle,\ (\mathtt{a},e_b){:}|\Phi^+\rangle \,\}$$
$$\mathtt{encode(m,a)}$$
$$\{\, \mathtt{class(m,a)},\ (e_m,E,e_a,e_b){:}\alpha|+\rangle|\psi_1\rangle|\Phi^+\rangle+\beta|-\rangle|\psi_2\rangle|\Psi^+\rangle \}$$

and

$$\{\, (e_m,E,e_a,\mathtt{b}){:}\alpha|+\rangle|\psi_1\rangle|\Phi^+\rangle+\beta|-\rangle|\psi_2\rangle|\Psi^+\rangle \,\}$$
$$\mathtt{decode(b,(m1,\ m2))}$$
$$\{\, (\mathtt{b},E){:}\alpha|0\rangle|\psi_1\rangle+\beta|1\rangle|\psi_2\rangle \,\}$$

Notice some peculiarities about the usage of ghost variables here:

- even though b is not measured, it is out of scope in `encode`; hence, we use the ghost variable e_b to specify the state of a.
- since both m and a are measured in `encode` we replace them with corresponding ghost variables in the postcondition specifying the joint entangled system.
- in both cases, when a qubit is out of scope or it is measured, the subscript encodes the relationship between the purifying system and the missing/measured qubit.
- ghost variables used in the way shown let us keep track of the complete information that may get lost during measurement.
- The substitution step for ghost variables in `teleport` after the calls to `encode` and `decode` requires a yet unclear frame rule.

The challenge is to formalize these reasoning principles in a deductive system.

References

[1] Kartik Singhal et al. "Q# as a Quantum Algorithmic Language". In: *Proc. QPL '22*. To appear. 2022. arXiv: 2206.03532.

[2] Li Zhou et al. "A Quantum Interpretation of Bunched Logic & Quantum Separation Logic". In: *Proc. LICS '21*. 2021, pp. 1–14. DOI: 10.1109/LICS52264.2021.9470673. arXiv: 2102.00329.

[3] Xuan-Bach Le et al. "A Quantum Interpretation of Separating Conjunction for Local Reasoning of Quantum Programs Based on Separation Logic". In: *Proc. ACM Program. Lang.* 6.POPL, 36 (2022). DOI: 10.1145/3498697.

[4] Dominique Unruh. "Quantum Hoare Logic with Ghost Variables". In: *Proc. LICS '19*. 2019, 47. DOI: 10.1109/LICS.2019.8785779.

[5] David Pym, Jonathan M. Spring, and Peter O'Hearn. "Why Separation Logic Works". In: *Philosophy & Technology* 32.3 (2019), pp. 483–516. DOI: 10.1007/s13347-018-0312-8.

[6] M. Ying. *Birkhoff-von Neumann Quantum Logic as an Assertion Language for Quantum Programs*. 2022. arXiv: 2205.01959.