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

KARTIK SINGHAL, University of Chicago, USA
ROBERT RAND, University of Chicago, USA
MATTHEW AMY, Simon Fraser University, Canada

There has been a recent surge of interest in separation logics for quantum computation. Unlike Reynolds' initial attempts to formulate a separation logic for reasoning about *pointer manipulation* using small, intuitive programs, the attempts in the quantum setting have been to provide precise formulations of theory without providing much intuition. We argue that these attempts may be misguided: the fundamental problem hindering scalability of Hoare-like logics is modular reasoning about *entangled*, rather than separable, states. Hiding behind all their formal complexity, current logics are either too restrictive or, at best, solve simplistic problems. We motivate the quantum program verification community by proposing an ideal specification language.

CCS Concepts: • **Computer systems organization** → **Quantum computing**; • **Theory of computation** → **Separation logic**; **Hoare logic**; *Program specifications*; *Pre- and postconditions*; *Assertions*.

Additional Key Words and Phrases: Separation logic, Hoare logic, quantum computation, entanglement, purification, specification language, pre- and postconditions, program specifications

1 INTRODUCTION

Why does Separation Logic work in the classical setting? According to Pym et al. [2019], SL works because it provides the following features:

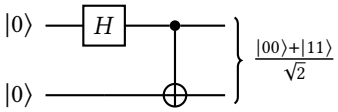
(1) a useful conceptual model of computer memory;
(2) a logical model and a *specification language* for true statements about memory;
(3) a productive overlap of these models—conceptual and logical;
(4) the *separating conjunction*, $*$, enabling the *frame rule* for local reasoning;
(5) scalable pre-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

The frame rule says that if we have a valid triple $\{P\}\ c\ \{Q\}$ then we can extend its pre- and postcondition with some predicate R, provided that R does not have any free variables in common with the variables modified by c, obtaining $\{P * R\}\ c\ \{Q * R\}$. This way, we can reason locally about program fragments while ignoring the global context surrounding that fragment.

Existing work on separation logics for quantum computation by Zhou et al. [2021] and Le et al. [2022] only caters to two of the above features—a logical model and the frame rule. Specifically, neither provides a model for quantum memory that quantum software engineers may rely on, nor do they offer a usable specification language for them to write true statements about the quantum state. The lack of a conceptual quantum memory model and, more importantly, a useful assertion language hampers the scalability of pre- and postconditions, rendering these logics difficult to use in practice.

The most intuitive interpretation of separating conjunction in the quantum is setting is separability: When two states are non-entangled, a $*$ may be placed between those states, which is what the existing works adopt. Unlike the classical setting, where the possibility of aliasing of pointers hindered applications of Hoare logic to practical software for decades prior to the invention of separation logic, there is no issue of aliasing among separable quantum states, obviating the need for a more general notion of separating conjunction. However, what these quantum separation logics lack is the ability to state anything useful about entangled states locally. In current quantum

Authors' addresses: Kartik Singhal, University of Chicago, Chicago, USA, ks@cs.uchicago.edu; Robert Rand, University of Chicago, Chicago, USA, rand@uchicago.edu; Matthew Amy, Simon Fraser University, Burnaby, Canada, matt_amy@sfu.ca.




```
operation bell00(a:Qubit, b:Qubit):Unit {
  H(a);
  CNOT(a,b);
}
```

Fig. 1. Circuit and program to prepare the first Bell state

separation logics, one resorts to global reasoning when entangled qubits are encountered. If that is so, we may ask why existing quantum Hoare logics [Ying 2012; Unruh 2019] are insufficient.

A quantum separation logic should be able to answer the following questions:

- What does the separating conjunction gain us that existing Hoare logics struggle with?
- What is the underlying memory model? Is the separation logic tailored to that model?
- Why *quantum* separation logic? Is there something unique to quantum computing at play, or is it simply a matter of combining classical separation logic with quantum predicates?
- What is the quantum equivalent of a separating implication, $-\!\!*$ (magic wand)?

Specifying a Bell State. Fig. 1 shows a circuit and the corresponding program to create the first Bell state.

The following *small footprint* specification (which follows the principle of local reasoning) should hold for a program that implements this circuit:

$$\{a : |0\rangle, b : |0\rangle\}\ \mathrm{bell}_{00}(a, b)\ \{(a, b) : |\Phi^+\rangle\}$$

Here $|\Phi^+\rangle = \frac{|00\rangle+|11\rangle}{\sqrt{2}}$ and the comma in the precondition means conjunction (for brevity). Note that the postcondition has to specify the joint state of a and b as they are entangled. What if we wanted to specify just the state associated with the subsystem b? In quantum teleportation, for instance, both parties have one half of the entangled pair, so we should be able to reason about either half locally.

2 TOWARD A USABLE SPECIFICATION LANGUAGE

What is the memory model?

> The logic is nice, but it's the model that's really important.
>
> — Reynolds to O'Hearn [2019]

A key reason for the success of classical separation logic is that it exposes a coherent view of the conceptual and logical memory model to the programmer. It is important to consider the difference between a stack-allocated value and a heap-allocated memory cell because the pointer aliasing problem is relevant only to the latter. Hence, $*$ splits only the heap, not the stack. Further, depending on the memory management primitives provided by a programming language, different sets of (small) axioms for those primitives are needed.

Since there is no agreement on a standard view of quantum memory, when designing a separation-like logic, it is important to emphasize the memory model being assumed. We should ask whether the qubits are allocated in a stack-like manner (as recently considered by Singhal et al. [2022]) or dynamically, similar to a classical heap. Further, we should ask whether this distinction in the memory model affects the reasoning available in the logic.

Zhou et al. [2021] focus on the quantum **while**-language [Ying 2012] that supports neither allocation nor deallocation of qubits and assumes a fixed set of qubits (i. e., no memory management). From the classical viewpoint, the utility of separation logic in this setting feels ill-motivated. Le et al. [2022]'s language considers a heap-like model of qubit allocation and deallocation and hence

is closer to the traditional problems considered in the classical literature. There is no aliasing, however.

Extending the principle of local reasoning to entangled states. O'Hearn et al. [2001] summarize the principle of local reasoning as follows:

> A specification and proof can concentrate on only those cells in memory that a program accesses.

The notion of "local reasoning" in Zhou et al. [2021] is subtly different from the traditional notion in separation logic literature because they do not explicitly refer to qubits in their specifications; "modular" reasoning would be more apt for their work. This modular reasoning is restricted to globally disjoint computations rather than the ones which internally entangle then disentangle qubits. Further, they cover entanglement reasoning using their UnCR ("uncorrelated") rule, but the authors leave out any discussion, preferring explanation by example using a somewhat complicated case study of Variational Quantum Algorithm (VQA).

Le et al.'s logic [2022] is closer to classical reasoning about memory management and hence supports an analogous notion of local reasoning. Its rule for measurement is crippled, however, in that it only supports reasoning about programs that defer measurement to the end. Further, it is limited to reasoning about separable states without any notion of entanglement between states.

While these works provide the first steps toward local reasoning about quantum programs, there is work to be done to develop scalable modes of modular reasoning. Here, we outline a practical specification language designed for scalable reasoning about quantum programs with evolving entanglement.

2.1 Programming Language

Let us consider a simple ML-like programming language for concreteness. Here is the grammar for primitive quantum commands:

Cmd	::=	`new q`	Fresh qubit allocation
		U_n`(q1,..,qn)`	Unitary gate application
		`M(q)`	Measurement

Qubits are managed in a stack-like fashion and automatically deallocated at the end of the lexical scope of the allocation.

Since we are imagining a computational model where there is no heap, we might consider our computational state to be a mapping from qubit variables to quantum state vectors. Unfortunately, this model breaks down when we have entangled states, so we can generalize the keys of this mapping to be a set of variables instead (with a suitable ordering that corresponds to the state vector described).

$$\text{States}_V = V \longrightarrow \text{State Vectors}$$

where V is a finite set of variables. Here are a few examples:

$$q : |0\rangle$$

$$(a, b) : |\Phi^+\rangle$$

2.2 Assertions

Assertions describe states. We use Unruh's assertion language, which builds upon the seminal logic of Birkhoff and von Neumann [1936]. Unruh [2019] defines quantum equivalents of $\top, \bot, \wedge, \vee$ as subspaces of a Hilbert space over a set of quantum variables, provides additional predicates specific to quantum computation and, most interestingly, some predicates that use *ghost variables* to express

properties like separability of the quantum state. Ghost variables are variables that appear in the logical predicates but not the program itself, and can be used to express, for instance, that a qubit has been measured.

Ghost variables are also used to describe the state of a qubit which is part of an entangled state. Variables such as e and E denote entangled ghost variables. As we will see, being able to describe the state of a qubit (or qubits) in entanglement locally, even when other qubits that are part of the same subsystem are not in scope, imparts much expressive power to the specification language. The quantum information-theoretic notion of purification enables this trick. Here is an example:

$$(a, e) : |\Phi^+\rangle$$

where e is a fresh entangled ghost variable corresponding to the other half of the first Bell state. We can think of these ghost variables as existential variables that we can introduce to represent any missing part of an entangled system, i. e., the example above can be thought of as $\exists e.(a, e) : |\Phi^+\rangle$. These variables get instantiated when we switch our reasoning from local to global using the equivalent of a frame rule. This substantially increases the expressiveness of our assertions. The teleportation example below demonstrates what we mean.

2.3 Specifications

Once we have a programming language and know how to write assertions, we can specify the behavior of complete programs using traditional Hoare triples. We can specify teleportation program as $\{\texttt{msg:}|\psi\rangle\texttt{,tgt:}|0\rangle\}$ `teleport(msg,tgt)` $\{\texttt{class(msg),tgt:}|\psi\rangle\}$, where `class(q1,..,qn)` states that the qubits `q1,..,qn` are in a classical state.

3 EXAMPLES

First, let us see the reasoning steps involved in a simple example to create a Bell pair and measure the qubits one by one.

```
new a;
% {a:|0⟩}
new b;
% {a:|0⟩, b:|0⟩}
H(a);
% {a:|+⟩, b:|0⟩}
CNOT(a, b);
% {(a,b):(|00⟩+|11⟩)/√2}
let c = M(a);
% {(e_c,b):(|00⟩+|11⟩)/√2, class(a), (c = 0 ∨ c = 1)}
let d = M(b);
% {(e_c,e_d):(|00⟩+|11⟩)/√2, class(a,b), (c = 0 ∨ c = 1), (d = 0 ∨ d = 1)}
% => simplification/consequence step
% {class(a,b), [1/2,(c,d = 0,0) ∨ 1/2,(c,d = 1,1)]}
```

After `let c = M(a)`, qubit b is in a mixed state, so we use a ghost variable e_c to refer to the purifying system. Note that the subscript encodes the relationship between the purifying system and the measurement outcome. If we need to branch over a measurement result, the logic then has complete information about all possible branches, as shown in the last step. Contrast this with Le et al.'s logic that does not allow controlling over a measurement result.

As our second example, consider another specification of teleportation in Fig. 2, which can transport not just a pure but also an entangled state. Here, the m qubit (for message) can be part of a much larger system; everything except the message is represented by the entangled ghost

```
let entangle(a,b) be {
    H(a);
    CNOT(a,b);
} in
let encode(m,a) be {
    CNOT(m,a);
    H(m);
    return (M(m),M(a));
} in
let decode(b,(m1,m2)) be {
    if m1 then Z(b);
    if m2 then X(b);
} in
let teleport(m,b) be {
    new a;
    entangle(a,b);
    let resultbits = encode(m,a);
    decode(b,resultbits);
}
```

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

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

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

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

Fig. 2. Modular teleportation and specification of its parts

variables E (which could be empty in the case of a pure state). a denotes Alice's qubit, b denotes Bob's. Note that $|\Psi^+\rangle = \frac{|01\rangle+|10\rangle}{\sqrt{2}}$ and, in the specification of encode, we decompose the entangled state $|\psi\rangle$ of the subsystem containing qubit m over the computational basis to allow a type of local reasoning over unentangled states in superposition.

The spec for teleport fully specifies what we would like the function to do—we can state that the message qubit is measured (is in the classical state) at the end and that the message qubit gets swapped by Bob's qubit in the original entangled system. Similarly, in the case of encode, we state that we expect qubit a to be a part of a Bell pair and that both input qubits get measured at the end. Meanwhile, ghost variables let us keep track of the complete information that may get lost during measurement. The substitution step for ghost variables in the teleport function after the call to encode and decode functions does most of the heavy lifting.

4 RELATED WORK

The first use of ghost or auxiliary variables to account for divergence based on control was considered by Ying et al. [2018]. Unruh [2019] developed the notion of ghost variables as used here to account for entanglement and measurement in quantum programs. Unruh's predicates were also adopted in the context of Hoare Type Theory by Singhal [2020]. In preliminary work, Singhal and Reppy [2021] raised a need for a separation logic for quantum programs which could be used as an assertion logic in their type theory.

Ying [2022] raises similar concerns as us for the need for an assertion/specification language but in the context of general quantum Hoare logics and only accounting for separable Hilbert spaces. We focus on being able to specify and reason locally about entangled systems.

REFERENCES

Garrett Birkhoff and John von Neumann. 1936. "The Logic of Quantum Mechanics." *Annals of Mathematics*, 37, 4, 823–843. DOI: 10.2307/1968621.

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, 36. DOI: 10.1145/3498697.

Peter O'Hearn. Jan. 2019. "Separation Logic." *Commun. ACM*, 62, 2, (Jan. 2019), 86–95. DOI: 10.1145/3211968.

Peter O'Hearn, John Reynolds, and Hongseok Yang. 2001. "Local Reasoning about Programs that Alter Data Structures." In: *Computer Science Logic*. Springer, 1–19. http://www0.cs.ucl.ac.uk/staff/p.ohearn/papers/localreasoning.pdf. DOI: 10.1007/3-540-44802-0_1.

David Pym, Jonathan M. Spring, and Peter O'Hearn. 2019. "Why Separation Logic Works." *Philosophy & Technology*, 32, 3, 483–516. DOI: 10.1007/s13347-018-0312-8.

Kartik Singhal. 2020. "Quantum Hoare Type Theory." Master's thesis. University of Chicago, Chicago, IL. https://ks.cs.uchicago.edu/publication/qhtt-masters/ arXiv: 2012.02154.

Kartik Singhal, Kesha Hietala, Sarah Marshall, and Robert Rand. 2022. "Q# as a Quantum Algorithmic Language." In: *Proceedings of the 19th International Conference on Quantum Physics and Logic (QPL), Oxford, U.K., June 27–July 1, 2022*. To appear. https://ks.cs.uchicago.edu/publication/q-algol/.

Kartik Singhal and John Reppy. 2021. "Quantum Hoare Type Theory: Extended Abstract." In: *Proc. QPL '20* (EPTCS). Vol. 340, 291–302. https://ks.cs.uchicago.edu/publication/qhtt/. DOI: 10.4204/EPTCS.340.15.

Dominique Unruh. 2019. "Quantum Hoare Logic with Ghost Variables." In: *Proc. LICS '19*. IEEE Computer Society. arXiv: 1902.00325. DOI: 10.1109/LICS.2019.8785779.

Mingsheng Ying. 2012. "Floyd–Hoare Logic for Quantum Programs." *ACM Trans. Program. Lang. Syst.*, 33, 6, 19. DOI: 10.1145/2049706.2049708.

Mingsheng Ying. 2022. *Birkhoff-von Neumann Quantum Logic as an Assertion Language for Quantum Programs*. (2022). arXiv: 2205.01959.

Mingsheng Ying, Li Zhou, and Yangjia Li. 2018. *Reasoning about Parallel Quantum Programs*. (2018). arXiv: 1810.11334.

Li Zhou, Gilles Barthe, Justin Hsu, Mingsheng Ying, and Nengkun Yu. 2021. "A Quantum Interpretation of Bunched Logic & Quantum Separation Logic." In: *Proc. LICS '21*, 1–14. arXiv: 2102.00329. DOI: 10.1109/LICS52264.2021.9470673.