Q*: Implementing Quantum Separation Logic in F*

Kesha Hietala, Sarah Marshall, Robert Rand, Nikhil Swamy
Amazon Microsoft UChicago Microsoft

PLanQC |2022⟩ Ljubljana

Background: Separation Logic

- An extension of Hoare Logic with additional operators including $\star$ ("separating conjunction"), which describes disjoint parts of the heap
  - $\{x \mapsto 0 \star y \mapsto 0\}$ says that variables x and y both have value 0, and that they are *distinct* (i.e., not aliases of each other)
- The *frame rule* supports scalable reasoning

$$\frac{\{P\}\ C\ \{Q\}}{\{P \star R\}\ C\ \{Q \star R\}},\ \mathsf{mod}(C) \cap \mathsf{fv}(R) = \emptyset$$

  - Allows us to prove a "local" property $\{P\}\ C\ \{Q\}$ and extend it to a "global" property $\{P \star R\}\ C\ \{Q \star R\}$

Quantum Separation Logic?

- $\star$ describes *separability* of quantum states
  - $P_1 \star P_2$ says that we can partition the global state Ψ into Ψ_1 and Ψ_2 such that P_1 holds of Ψ_1, P_2 holds of Ψ_2, and $\Psi = \Psi_1 \otimes \Psi_2$
  - Provides a convenient notation for describing whether states are entangled
  - Allows us to reason modularly about parts of the state that are not entangled
- Proposed by Zhou at el. (2021) and Le et al. (2022)
  - But no implementation

F*: A Proof-Oriented Programming Language

- F* is a functional programming language and proof assistant from Microsoft Research
  - Uses the Z3 solver in the backend for automation

https://www.fstar-lang.org/

- Steel (Fromherz et al. (2021)) is an F* implementation of a *concurrent* separation logic
  - By building on top of Steel, we get a framework for the ★ operator and frame rule "for free"

Modeling Quantum State

- In order to interpret $\star$ in Steel's separation logic, we need a model of quantum state & a partial commutative monoid over it
- We define a type qvec qs, which is a wrapper around a complex vector of length $2^{|qs|}$ with a commutative definition of tensor
  - Our underlying matrix library is a port of the QuantumLib library (used in SQIR and QWIRE) from Coq to F*
  - Our commutative definition of tensor is a work-in-progress, but our idea is to apply the standard Kronecker product followed by a permutation matrix to maintain a fixed ordering of qubits

Q* = F* with Quantum Actions

- We introduce a predicate $\overline{q} \mapsto |\psi\rangle$, which says that the set of qubits $\overline{q}$ are collectively in state $|\psi\rangle$ (and, implicitly, unentangled with outside qubits)
- We define four quantum *actions*

$$\{\, \mathbf{emp} \,\}\; q \;\leftarrow\; \mathbf{alloc} \,\{\, q \mapsto |0\rangle \,\}$$

$$\{\, q \mapsto |\psi\rangle \,\}\; \mathbf{discard}\; q \,\{\, \mathbf{emp} \,\}$$

$$\{\, q \cup \overline{q} \mapsto |\psi\rangle \,\}\; b \;\leftarrow\; \mathbf{measure}\; q \,\{\, q \mapsto |b\rangle \star \overline{q} \mapsto \mathrm{proj}(q, b, |\psi\rangle) \,\}$$

$$\{\, \overline{q} \mapsto |\psi\rangle \,\}\; \mathbf{apply}\; \mathbf{U}\; \overline{q} \,\{\, \overline{q} \mapsto U\,|\psi\rangle \,\}$$

- And an *entailment rule*

$$\overline{q_1} \cup \overline{q_2} \mapsto |\psi_1\rangle_{\overline{q_1}} \otimes |\psi_2\rangle_{\overline{q_2}} \iff (\overline{q_1} \mapsto |\psi_1\rangle) \star (\overline{q_2} \mapsto |\psi_2\rangle)$$

Example: Quantum Teleportation

(written in Microsoft's Q# language)

```
operation Entangle (qAlice : Qubit, qBob : Qubit) : Unit is Adj {
    H(qAlice);
    CNOT(qAlice, qBob);
}

operation SendMsg (qAlice : Qubit, qMsg : Qubit) : (Bool, Bool) {
    Adjoint Entangle(qMsg, qAlice);
    let m1 = M(qMsg);
    let m2 = M(qAlice);
    return (m1 == One, m2 == One);
}

operation DecodeMsg (qBob : Qubit, (b1 : Bool, b2 : Bool)) : Unit {
    if b1 { Z(qBob); }
    if b2 { X(qBob); }
}

operation Teleport (qMsg : Qubit, qBob : Qubit) : Unit {
    use qAlice = Qubit();
    Entangle(qAlice, qBob);
    let classicalBits = SendMsg(qAlice, qMsg);
    DecodeMsg(qBob, classicalBits);
}
```

quantum gates

arguments

return type

operation is automatically "adjointable"

adjoint operation

measurement

classical control flow

qubit allocation

implicit deallocation

Example: Quantum Teleportation

$\{q_A \mapsto |0\rangle \star q_B \mapsto |0\rangle\}$ `Entangle(qA, qB)` $\{\{q_A, q_B\} \mapsto \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)\}$

$\{q_M \cup \bar{q} \mapsto |\phi\rangle \star \{q_A, q_B\} \mapsto \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)\}$

`let (b1,b2) = SendMsg(qA,qM)`

$\{q_M \mapsto |b_1\rangle \star q_A \mapsto |b_2\rangle \star q_B \cup \bar{q} \mapsto Z_{q_B}^{b_1} X_{q_B}^{b_2} |\phi\rangle\}$

$\{q_B \cup \bar{q} \mapsto |\phi\rangle\}$ `DecodeMsg(qB, (b1,b2))` $\{q_B \cup \bar{q} \mapsto X_{q_B}^{b_2} Z_{q_B}^{b_1} |\phi\rangle\}$

$\{q_M \cup \bar{q} \mapsto |\phi\rangle \star q_B \mapsto |0\rangle\}$ `Teleport(qM, qB)` $\{q_B \cup \bar{q} \mapsto |\phi\rangle\}$

Prototype Implementation

- Available at github.com/microsoft/qsharp-verifier/tree/sep-logic
  - Not under active development, but open to contributions!

- Type for teleport:

```
val teleport (#qs:qbits)
             (qM:qbit{ disjoint {qM} qs })
             (#st:qvec (union {qM} qs))
             (qB:qbit{ qB <> qM /\ disjoint {qB} qs })
  : STT unit
        (pts_to (union {qM} qs) st `star` pts_to {qB} (ket _ false))
        (fun _ -> pts_to (union {qB} qs) st)
```

"other" qubits in the environment (implicit)

message qubit, distinct from qs

initial state of qM and qs (implicit)

Bob's qubit, distinct from qs and qM

Steel return type is unit

precondition: $\{\, q_M \cup \overline{q} \mapsto |\psi\rangle \star q_B \mapsto |0\rangle \,\}$

postcondition: $\{\, q_B \cup \overline{q} \mapsto |\psi\rangle \,\}$

Prototype Implementation

```
let teleport
 = let qA = alloc () in
   disjointness (single qA) (single qB) #_;
   disjointness (single qA) (union (single qM) qs) #_;
   entangle qA qB;
   let bits = send_msg qA qM #_ in
   decode_msg qB qs bits;
   discard qA _;
   discard qM _;
   teleport_lemma (fst bits) (snd bits) qB qs
                  (relabel_indices (union (single qB) qs) state);
   rewrite (pts_to (union (single qB) qs) _)
           (pts_to (union (single qB) qs)
             (relabel_indices (union (single qB) qs) state))
```

code

proof

Additional Applications

- Discard safety
  - A qubit must be unentangled when deallocated
- Qubit resetting and reuse
  - Confirm that a qubit is in the $|0\rangle$ state on discard
- No cloning
  - Check whether qubits alias one another
- More in Kesha's thesis: https://khieta.github.io/files/drafts/khieta-dissertation.pdf

Future Directions

Code available at: github.com/microsoft/qsharp-verifier
Separation logic w/ teleport example in the **sep-logic** branch

- Verify more interesting classical/quantum programs
  - Idea: Use Steel to reason about hybrid quantum/classical *concurrent* programs
- Fix rough edges in the implementation (many admits in our lin. algebra code)
- Integrate Q* into the Q# toolchain