Gottesman Types for Quantum Programs

Robert Rand
University of Maryland
rrand@cs.umd.edu

Aarthi Sundaram
Microsoft Quantum
aarthims@gmail.com

Kartik Singhal
University of Chicago
ks@cs.uchicago.edu

Brad Lackey
Microsoft Quantum
University of Maryland
bclackey@umd.edu

The Heisenberg representation of quantum operators provides a powerful technique for reasoning about quantum circuits, albeit those restricted to the common (non-universal) Clifford set H, S and $CNOT$. The Gottesman-Knill theorem showed that we can use this representation to efficiently simulate Clifford circuits. We show that Gottesman's semantics for quantum programs can be treated as a type system, allowing us to efficiently characterize a common subset of quantum programs. We also show that it can be extended beyond the Clifford set to partially characterize a broad range of programs. We apply these types to reason about separable states and the superdense coding algorithm.

1 Introduction

The *Heisenberg representation* of quantum mechanics treats quantum operators as functions on unitary matrices, rather than on the quantum state. For instance, for any quantum state $|\phi\rangle$,

$$ZH|\phi\rangle = HX|\phi\rangle \tag{1}$$

so an H gate can be viewed as a higher-order function that takes Z to X (and similarly takes X to Z). Gottesman [7] uses this representation to present the rules for how the Clifford quantum gates H, S and $CNOT$ operate on Pauli X and Z gates, which is sufficient to fully describe the behavior of the Clifford operators. There, H is given the following description based on its action above:

$$H : \mathbf{Z} \to \mathbf{X} \qquad H : \mathbf{X} \to \mathbf{Z}$$

In Gottesman's paper, the end goal is to fully describe quantum programs and prove the *Gottesman-Knill theorem*, which shows that any Clifford circuit can be simulated efficiently. Here we observe that the judgements above look like typing judgments, and show that they can indeed be treated as such (§3). As such, they can be used to make coarse guarantees about programs, without fully describing the programs' behaviors. We show a simple example of applying this system to the superdense coding algorithm (§5). In §6, we demonstrate, using the GHZ state $|000\rangle + |111\rangle$, how the type system is capable of tracking both the creation and destruction of entanglement. In §7, we extend the type system to deal with programs outside the Clifford group and use it to characterize the Toffoli gate. We discuss the possible future applications of this system in §8.

The system and examples in this paper are formalized in Coq at `https://github.com/inQWIRE/GottesmanTypes`.

To appear in EPTCS.

2 Gottesman Semantics

Gottesman's semantics for H, S, and $CNOT$ are given by the following table.

$$\begin{array}{rlrl} H: & \mathbf{X}\to\mathbf{Z} & CNOT: & \mathbf{X}\otimes\mathbf{I}\to\mathbf{X}\otimes\mathbf{X} \\ H: & \mathbf{Z}\to\mathbf{X} & CNOT: & \mathbf{I}\otimes\mathbf{X}\to\mathbf{I}\otimes\mathbf{X} \\ S: & \mathbf{X}\to\mathbf{Y} & CNOT: & \mathbf{Z}\otimes\mathbf{I}\to\mathbf{Z}\otimes\mathbf{I} \\ S: & \mathbf{Z}\to\mathbf{Z} & CNOT: & \mathbf{I}\otimes\mathbf{Z}\to\mathbf{Z}\otimes\mathbf{Z} \end{array}$$

Note that these rules are intended to simply describe the action of each gate on the corresponding unitary matrices, as in equation 1. However, given the action of a circuit on every permutation of X and Z (or any spanning set) we can deduce the semantics of the program itself [7].[1] For our purposes, though, we will treat these as a type system, justifying that choice in §3.

We can combine our typing rules by simple multiplication, for instance combining the second and third rules for $CNOT$, we get

$$CNOT : (\mathbf{IZ}\otimes\mathbf{XI}\to\mathbf{IZ}\otimes\mathbf{XI}) = \mathbf{Z}\otimes\mathbf{X}\to\mathbf{Z}\otimes\mathbf{X}$$

In the rule for S, $\mathbf{Y}$ is equivalent to $i\mathbf{XZ}$ so we can reason about an S applied twice compositionally. We use $f \,\$\, g$ for forward function composition (equivalent to $g\circ f$):

$$S;S : (\mathbf{X}\to i\mathbf{XZ}\,\$\, i\mathbf{XZ}\to i\mathbf{YZ}) = \mathbf{X}\to -\mathbf{X}$$

Once we have a type system for H, S and $CNOT$ we can define additional gates in terms of these, and derive their types. For instance, the Pauli Z gate is simply $S;S$, for which we derived the following type for $\mathbf{X}$ and trivially can derive the type for $\mathbf{Z}$:

$$\begin{aligned} Z:\ & \mathbf{X}\to -\mathbf{X} \\ Z:\ & \mathbf{Z}\to\mathbf{Z} \end{aligned}$$

Defining X as $H;Z;H$, we can derive the type for the Pauli X gate as:

$$\begin{aligned} X = H;Z;H:\ & (\mathbf{X}\to\mathbf{Z}\,\$\,\mathbf{Z}\to\mathbf{Z}\,\$\,\mathbf{Z}\to\mathbf{X}) = \mathbf{X}\to\mathbf{X} \\ X = H;Z;H:\ & (\mathbf{Z}\to\mathbf{X}\,\$\,\mathbf{X}\to -\mathbf{X}\,\$\,-\mathbf{X}\to -\mathbf{Z}) = \mathbf{Z}\to -\mathbf{Z} \end{aligned}$$

Likewise, $Y = S;X;Z;S$ and so, the type for the Pauli Y gate would be:

$$\begin{aligned} Y = S;Z;X;S:\ & (\mathbf{Z}\to\mathbf{Z}\,\$\,\mathbf{Z}\to\mathbf{Z}\,\$\,\mathbf{Z}\to -\mathbf{Z}\,\$\,-\mathbf{Z}\to -\mathbf{Z}) = \mathbf{Z}\to -\mathbf{Z} \\ Y = S;Z;X;S:\ & (\mathbf{X}\to\mathbf{Y}\,\$\,\mathbf{Y}\to -\mathbf{Y}\,\$\,-\mathbf{Y}\to\mathbf{Y}\,\$\,\mathbf{Y}\to -\mathbf{X}) = \mathbf{X}\to -\mathbf{X} \end{aligned}$$

We can also define more complicated gates like CZ and $SWAP$ as $H_2;CNOT;H_2$ (where H_2 is H applid to $CNOT$'s target qubit) and $CNOT;NOTC;CNOT$, (where $NOTC$ is a flipped $CNOT$) for which we can easily derive the following types:

$$\begin{array}{rlrl} CZ: & \mathbf{X}\otimes\mathbf{I}\to\mathbf{X}\otimes\mathbf{Z} & SWAP: & \mathbf{X}\otimes\mathbf{I}\to\mathbf{I}\otimes\mathbf{X} \\ CZ: & \mathbf{I}\otimes\mathbf{X}\to\mathbf{Z}\otimes\mathbf{X} & SWAP: & \mathbf{I}\otimes\mathbf{X}\to\mathbf{X}\otimes\mathbf{I} \\ CZ: & \mathbf{Z}\otimes\mathbf{I}\to\mathbf{Z}\otimes\mathbf{I} & SWAP: & \mathbf{Z}\otimes\mathbf{I}\to\mathbf{I}\otimes\mathbf{Z} \\ CZ: & \mathbf{I}\otimes\mathbf{Z}\to\mathbf{I}\otimes\mathbf{Z} & SWAP: & \mathbf{I}\otimes\mathbf{Z}\to\mathbf{Z}\otimes\mathbf{I} \end{array}$$

[1] The intuition behind this comes from the fact that this set of X and Z operators form the generators of the Pauli basis for quantum states. Hence, deducing the action of a unitary matrix on them provides an information theoretically complete picture of the action of the matrix on any input and suffices to deduce the semantics of the program.

By combining the rules stated above, we can also derive the action of *SWAP* on $\mathbf{X} \otimes \mathbf{Y}$ as:

$$SWAP : (\mathbf{X} \otimes \mathbf{Y} = \mathbf{XII} \otimes \mathbf{I}(i\mathbf{X})\mathbf{Z}) \to (\mathbf{I}(i\mathbf{X})\mathbf{Z} \otimes \mathbf{XII} = \mathbf{Y} \otimes \mathbf{X})$$

This gives us less information than the types for *SWAP* above, but might be the type we intend for a given use of swapping: In particular, we might want to use a *SWAP* to exchange qubits in the *X* and *Y* bases, as we will show.

We show the full rules for typing circuits in fig. 1, which we will reference throughout this paper.

3 Interpretation on Basis States

We can interpret the type $H : \mathbf{Z} \to \mathbf{X}$ as saying that H takes a qubit in the $\mathbf{Z}$ basis (that is, $|0\rangle$ or $|1\rangle$) to a qubit in the $\mathbf{X}$ basis ($|+\rangle$ and $|-\rangle$). This form of reasoning is present in Perdrix's [15] work on abstract interpretation for quantum systems, which classifies qubits in an $\mathbf{X}$ or $\mathbf{Z}$ basis, for the purpose of tracking entanglement. Unfortunately, that system cannot leave the $\mathbf{X}$ and $\mathbf{Z}$ bases, and hence cannot derive that $Z : \mathbf{X} \to -\mathbf{X}$ due to the intermediate $\mathbf{Y}$. More fundamentally, while we can show that *SWAP* has type $\mathbf{X} \otimes \mathbf{Z} \to \mathbf{Z} \otimes \mathbf{X}$, the first *CNOT* application entangles the two qubits (represented in our system by $\mathbf{XZ} \otimes \mathbf{XZ}$) which Perdrix classifies as simply $\top$ and marks as potentially entangled. (We also use $\top$ but only for gates that lie outside the set of Clifford operators, see §7.) As a result, Perdrix's system typically classifies most circuits as $\top$ after just a few gate applications, while ours never leaves the Pauli bases as long as we apply Clifford gates.

Proposition 1. *Given a unitary $U : A \to B$ in the Heisenberg interpretation, U takes every eigenstate of A to an eigenstate of B.*

Proof. From eq. [1] in Gottesman [7], given a state $|\psi\rangle$ and an operator U,

$$UN|\psi\rangle = UNU^\dagger U|\psi\rangle.$$

Additionally, in the Heisenberg interpretation this can be denoted as: $U : N \to UNU^\dagger$. Suppose that $|\psi\rangle$ is an eigenstate of N with eigenvalue λ and let $|\phi\rangle$ denote the state after U acts on $|\psi\rangle$. Then,

$$\lambda|\phi\rangle = U(\lambda|\psi\rangle) = UN|\psi\rangle = UNU^\dagger U|\psi\rangle = (UNU^\dagger)|\phi\rangle.$$

Hence, $|\phi\rangle$ is an eigenstate of the modified operator $UNU^\dagger$ with eigenvalue λ. □

Intersection Types It may seem odd that we have given multiple types to H, S, *CNOT* and the various derived operators. This isn't particularly rare in type systems with subtyping, which is an appropriate lens for viewing the types we have given above. However, it is useful to have the most descriptive type for any term in our language. We can give these using intersection types. For instance, we have the following types for H and *CNOT*:

$$\begin{aligned} H &: (\mathbf{X} \to \mathbf{Z}) \cap (\mathbf{Z} \to \mathbf{X}) \\ CNOT &: (\mathbf{X} \otimes \mathbf{I} \to \mathbf{X} \otimes \mathbf{X}) \cap (\mathbf{I} \otimes \mathbf{X} \to \mathbf{I} \otimes \mathbf{X}) \cap (\mathbf{Z} \otimes \mathbf{I} \to \mathbf{Z} \otimes \mathbf{I}) \cap (\mathbf{I} \otimes \mathbf{Z} \to \mathbf{Z} \otimes \mathbf{Z}) \end{aligned}$$

From these we can derive any of the types given earlier, using the standard rules for intersections (fig. 1). Another advantage of using intersection types is that it is closely related to the stabilizer formalism that is used extensively in error-correction. This connection is further discussed in §8.

1. Grammar:

$$G := \mathbf{I} \mid \mathbf{X} \mid \mathbf{Z} \mid -G \mid iG \mid G * G \mid G \otimes G \mid G \to G \mid G \cap G \mid G \times G \mid \top$$

2. Multiplication and Tensor Laws:

$$\begin{array}{cccc} \mathbf{X}*\mathbf{X} = \mathbf{I} & \mathbf{Z}*\mathbf{Z} = \mathbf{I} & \mathbf{Z}*\mathbf{X} = -\mathbf{X}*\mathbf{Z} & A*\mathbf{I} = A = \mathbf{I}*A \\ --A = A & i(iA) = -A & i(-A) = -(iA) & A*(B*C) = A*B*C \end{array}$$

$$-A*B = -(A*B) = A*-B \qquad iA*B = i(A*B) = A*iB$$

$$A \otimes (B \otimes C) = (A \otimes B) \otimes C \qquad A \otimes B = (A \otimes \mathbf{I}) * (\mathbf{I} \otimes B)$$

$$iA \otimes B = i(A \otimes B) = A \otimes iB \qquad -A \otimes B = -(A \otimes B) = A \otimes -B$$

$$(A \otimes B) * (C \otimes D) = (A * C) \otimes (B * D) \text{ where } |A| = |C|$$

3. Tensors Rules:

$$\frac{g\ 1 : A \to A' \qquad |E| = m-1}{g\ m : E \otimes A \otimes E' \to E \otimes A' \otimes E'}\ \otimes_1 \qquad \frac{g\ 2\ 1 : A \otimes B \to A' \otimes B'}{g\ 1\ 2 : B \otimes A \to B' \otimes A'}\ \otimes\text{-REV}$$

$$\frac{g\ 1\ 2 : A \otimes B \to A' \otimes B' \qquad |E| = m-1 \qquad |E'| = n-m-1}{g\ m\ n : E \otimes A \otimes E' \otimes B \otimes E'' \to E \otimes A' \otimes E' \otimes B' \otimes E''}\ \otimes_2$$

4. Arrow and Sequence Rules:

$$\frac{p : A \to A' \qquad p : B \to B'}{p : (A * B) \to (A' * B')}\ \text{MUL} \qquad \frac{p : A \to A'}{p : iA \to iA'}\ \text{IM}$$

$$\frac{p_1 : A \to B \qquad p_2 : B \to C}{p_1; p_2 : A \to C}\ \text{CUT} \qquad \frac{p : A \to A'}{p : -A \to -A'}\ \text{NEG}$$

$$p_1; (p_2; p_3) : A \equiv (p_1; p_2); p_3 : A$$

5. Intersection Rules:

$$I^{\otimes n} \cap A = A \qquad A \cap A = A \qquad A \cap B = B \cap A \qquad A \cap B \cap C = A \cap (B \cap C)$$

$$\frac{g : A \qquad g : B}{g : A \cap B}\ \cap\text{-I} \qquad \frac{g : A \cap B}{g : A}\ \cap\text{-E}$$

$$\frac{g : (A \to B) \cap (A \to C)}{g : A \to (B \cap C)}\ \cap\text{-ARR} \qquad \frac{g : (A \to A') \cap (B \to B')}{g : (A \cap B) \to (A' \cap B')}\ \cap\text{-ARR-DIST}$$

6. Separability Rules

$$(A \otimes \mathbf{I}^n) = A \times \mathbf{I}^n \text{ where } A \in \{1, -1, i, -i\} * \{X, Y, Z\}$$

$$(A \times B) \cap (A \otimes C) = A \times (B \cap C) \qquad (A \times B) \cap (I^{\otimes n} \otimes C) = A \times (B \cap C)$$

Figure 1: The grammar and typing rules for Gottesman types. The grammar allows us to describe ill-formed types, such as $\mathbf{X} \cap (\mathbf{I} \otimes \mathbf{Z})$, but these don't type any circuits. The intersection and arrow typing rules are derived from standard subtyping rules [16, Chapter 15]. The arity of a type not containing $\{\times, \to\}$ is the longest sequence of atoms connected by tensors. For instance, $|\mathbf{I} * \mathbf{Z} \otimes i\mathbf{X}| = |(\mathbf{X} \otimes \mathbf{X}) \cap (\mathbf{Z} \otimes \mathbf{Z})| = 2$.

The role of I **I** plays an interesting role in this type system. For **I** alone, we have the following two facts, the first drawn from the Heisenberg representation of quantum mechanics and the second from our interpretation of types as eigenstates:

$$\forall\, U, \quad U : \mathbf{I} \to \mathbf{I} \tag{2}$$

$$\forall\ |\psi\rangle,\ |\psi\rangle : \mathbf{I} \tag{3}$$

This would lead us to treat **I** as a kind of top type, where $A <: \mathbf{I}$ for any type A. However, this would be incompatible with $\otimes$. For example, the two qubit Bell pair $|\Phi^+\rangle$ has type $\mathbf{X}\otimes\mathbf{X}$ but not type $\mathbf{X}\otimes\mathbf{I}$. By contrast, $\mathbf{X}\otimes\mathbf{I}$ contains precisely the separable two qubit states where the first qubit is an eigenstate of **X**. This second type, which is neither a subtype nor supertype of the first allows us to consider the important property of *separability* or non-entanglement of qubits.

4 Separability

Proposition 2. *For any Pauli matrix $U \in \{-1,1,-i,i\} * \{X,Y,Z\}$, the eigenstates of $U \otimes I^{\otimes n-1}$ are all the vectors $|u\rangle \otimes |\psi\rangle$ where $|u\rangle$ is an eigenstate of U and $|\psi\rangle$ is any state.*

Proof. Let $|\phi\rangle$ be the λ eigenstate and $\left|\phi^\perp\right\rangle$ be the $-\lambda$ eigenstate for $U \in \{1,-1,i,-i\} * \{X,Y,Z\}$ where $\lambda \in \{1,i\}$. Note that $\{|\phi\rangle, \left|\phi^\perp\right\rangle\}$ forms a single-qubit basis.

First, consider states of the form $|\gamma\rangle = |u\rangle \otimes |\psi\rangle$ where $|u\rangle \in \{|\phi\rangle, \left|\phi^\perp\right\rangle\}$ and $|\psi\rangle \in \mathbb{C}^{2^{n-1}}$. Clearly,

$$(U \otimes I^{\otimes n-1})\,|\gamma\rangle = (U \otimes I^{\otimes n-1})\,|u\rangle \otimes |\psi\rangle = (U\,|u\rangle) \otimes |\psi\rangle = \lambda_u\,|u\rangle \otimes |\psi\rangle\,.$$

Hence, every state of the form of $|\gamma\rangle$ is an eigenstate of $U \otimes I^{\otimes n-1}$. Additionally, note that by similar reasoning, for every separable state $|\gamma\rangle = |v\rangle \otimes |\psi\rangle$ where $|v\rangle \notin \{|\phi\rangle, \left|\phi^\perp\right\rangle\}$, $|\gamma\rangle$ is not an eigenstate of $U \otimes I^{\otimes n-1}$.

Now we show, that any state not in this separable form cannot be an eigenstate of $U \otimes I^{\otimes n-1}$. By way of contradiction assume that $|\delta\rangle$ is an eigenstate of $U \otimes I^{\otimes n-1}$ with $(U \otimes I^{\otimes n-1})\,|\delta\rangle = \mu\,|\delta\rangle$. Expand

$$|\delta\rangle = \alpha\,|\phi\rangle\,|\psi_1\rangle + \beta\left|\phi^\perp\right\rangle|\psi_2\rangle$$

where $|\psi_1\rangle, |\psi_2\rangle \in \mathbb{C}^{2^{n-1}}$. Then we compute

$$\begin{aligned}
(U \otimes I^{\otimes n-1})\,|\delta\rangle &= \alpha(U\,|\phi\rangle)\,|\psi_1\rangle + \beta(U\left|\phi^\perp\right\rangle)\,|\psi_2\rangle \\
&= \lambda\alpha\,|\phi\rangle\,|\psi_1\rangle - \lambda\beta\left|\phi^\perp\right\rangle|\psi_2\rangle \\
&= \mu\alpha\,|\phi\rangle\,|\psi_1\rangle + \mu\beta\left|\phi^\perp\right\rangle|\psi_2\rangle
\end{aligned}$$

where we have used that $|\phi\rangle$ and $\left|\phi^\perp\right\rangle$ are the $+\lambda$ and $-\lambda$ eigenvalues of U respectively. As the components of the expansion are orthogonal to each other, μ must satisfy:

$$\mu\alpha = \lambda\alpha \text{ and } \mu\beta = -\lambda\beta.$$

As $\lambda \neq 0$, since $U \otimes I^{\otimes n-1}$ is unitary, we either have (i) $\alpha = 0$, $\mu = -\lambda$, and $|\delta\rangle = \left|\phi^\perp\right\rangle|\psi_2\rangle$ or (ii) $\beta = 0$, $\mu = +\lambda$, and $|\delta\rangle = |\phi\rangle\,|\psi_1\rangle$. In either case $|\delta\rangle$ has a separable form as claimed. □

Following Gottesman's notation, let $U_k := I^{\otimes(n-k)} \otimes U \otimes I^{\otimes(n-k)}$ for $1 \leq k \leq n$ and a single qubit Pauli U. (Gottesman writes $\overline{\mathbf{X}}_k$ and $\overline{\mathbf{Z}}_k$; we'll elide the bar but make frequent use of the notation.) Combining Propositions 1 and 2, we obtain the following corollary:

Corollary 1. *Every term of type U_k is separable, for $U \in \{\pm X, \pm Y, \pm Z\}$ and $1 \leq k \leq n$. That is, the factor associated with U is unentangled with the rest of the system.*

In this light, we can reconsider the types we've given to *CNOT*. In §2, we showed that *CNOT* has the type $\mathbf{Z} \otimes \mathbf{X} \to \mathbf{Z} \otimes \mathbf{X}$. This is true but not particularly useful to a programmer who doesn't view the eigenvectors of $\mathbf{Z} \otimes \mathbf{X}$ as a helpful category. If however, we look at the intersection type given for *CNOT* in §3, we see that *CNOT* has the type $(\mathbf{Z} \otimes \mathbf{I} \to \mathbf{Z} \otimes \mathbf{I}) \cap (\mathbf{I} \otimes \mathbf{X} \to \mathbf{I} \otimes \mathbf{X})$. This says that *CNOT* takes $|i\rangle \otimes |\psi\rangle$ where $i \in \{0,1\}$ to a similar separable state, and likewise for when the second qubit is in $\mathbf{X}$. Hence, we can conclude that it takes a separable pair of $\mathbf{Z}$ and $\mathbf{X}$ qubits to a similar separable pair.

Our ∩-ARR-DIST rule (fig. 1), which follows directly from the subtyping rules for arrow and intersection[2], allows us to make this explicit. We can weaken *CNOT*'s type to $(\mathbf{Z} \otimes \mathbf{I} \cap \mathbf{I} \otimes \mathbf{X}) \to (\mathbf{Z} \otimes \mathbf{I} \cap \mathbf{I} \otimes \mathbf{X})$. This type, which encodes the separability of the $\mathbf{Z}$ and $\mathbf{X}$ qubits, is useful enough that we will introduce a new type constructor $\times$, where $A \times \mathbf{I}^{\otimes n} = A \otimes \mathbf{I}^{\otimes n}$ for $A \in \{\pm X, \pm Y, \pm Z\}$. This distributes in the expected way over intersections: $(A \times B) \cap (A \otimes C) = A \times (B \cap C)$ and $(A \times B) \cap (I \otimes C) = A \times (B \cap C)$, allowing us to derive $\mathit{CNOT} : \mathbf{Z} \times \mathbf{X}$.

5 Example: Superdense Coding

To illustrate the power of this simple system, consider the example of superdense coding as in fig. 2. Superdense coding allows Alice to convey two bits of information x and y, which we treat as qubits in the $\mathbf{Z}$ state, to Bob by sending a single qubit and consuming one EPR pair.

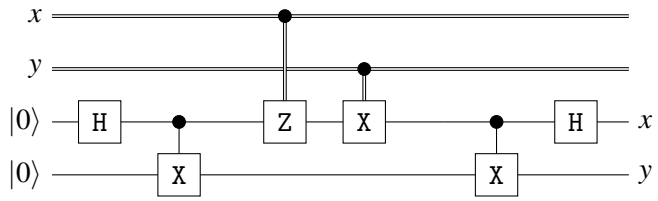

Figure 2: Superdense Coding sending classical bits x and y from Alice to Bob.

The desired type for this circuit is $\mathbf{Z} \times \mathbf{Z} \times \mathbf{Z} \times \mathbf{Z}$, showing that four classical bits are transmitted. Hence, we need to derive the output types for $\mathbf{Z}_1, \mathbf{Z}_2, \mathbf{Z}_3$ and $\mathbf{Z}_4$. We can trivially derive that `superdense` has types $\mathbf{Z}_1 \to \mathbf{Z}_1$ and $\mathbf{Z}_2 \to \mathbf{Z}_2$ (since *CZ* and *CNOT* have types $\mathbf{Z} \otimes \mathbf{I} \to \mathbf{Z} \otimes \mathbf{I}$), so we'll look at the derivation for $\mathbf{Z}_3$:

```
Definition superdense :=
  INIT ;     I ⊗ I ⊗ Z ⊗ I   (* initial type *)
  H 3  ;     I ⊗ I ⊗ X ⊗ I   (* create Bell pair *)
  CNOT 3 4 ; I ⊗ I ⊗ X ⊗ X
  CZ 1 3 ;   Z ⊗ I ⊗ X ⊗ X   (* map bits onto Bell pair *)
  CNOT 2 3 ; Z ⊗ I ⊗ X ⊗ X
  CNOT 3 4 ; Z ⊗ I ⊗ X ⊗ I   (* decode qubits *)
  H 3        Z ⊗ I ⊗ Z ⊗ I
```

[2]Thanks to Andreas Rossberg for pointing this out on Stack Overflow.

We can similarly derive $\texttt{superdense} : \texttt{I} \otimes \texttt{I} \otimes \texttt{I} \otimes \texttt{Z} \to \texttt{I} \otimes \texttt{Z} \otimes \texttt{I} \otimes \texttt{Z}$. We can now combine all four derivations using our distributivity rule for $\to$ and $\cap$ as follows:

$$\dfrac{\dfrac{\dfrac{\dfrac{\texttt{superdense} : (\mathbf{Z}_1 \to \mathbf{Z}_1) \cap (\mathbf{Z}_2 \to \mathbf{Z}_2) \cap (\mathbf{Z}_3 \to \mathbf{Z} \otimes \mathbf{I} \otimes \mathbf{Z} \otimes \mathbf{I}) \cap (\mathbf{Z}_4 \to \mathbf{I} \otimes \mathbf{Z} \otimes \mathbf{I} \otimes \mathbf{Z})}{\texttt{superdense} : \mathbf{Z}_1 \cap \mathbf{Z}_2 \cap \mathbf{Z}_3 \cap \mathbf{Z}_4 \to \mathbf{Z}_1 \cap \mathbf{Z}_2 \cap \mathbf{Z} \otimes \mathbf{I} \otimes \mathbf{Z} \otimes \mathbf{I} \cap \mathbf{I} \otimes \mathbf{Z} \otimes \mathbf{I} \otimes \mathbf{Z}}}{\texttt{superdense} : \mathbf{Z} \times \mathbf{Z} \times \mathbf{Z} \times \mathbf{Z} \to \mathbf{Z} \times (\mathbf{Z} \otimes \mathbf{I} \otimes \mathbf{I} \cap \mathbf{I} \otimes \mathbf{Z} \otimes \mathbf{I} \cap \mathbf{Z} \otimes \mathbf{I} \otimes \mathbf{Z})}}{\texttt{superdense} : \mathbf{Z} \times \mathbf{Z} \times \mathbf{Z} \times \mathbf{Z} \to \mathbf{Z} \times (\mathbf{Z} \times (\mathbf{Z} \otimes \mathbf{I} \cap \mathbf{I} \otimes \mathbf{Z})}}{\texttt{superdense} : \mathbf{Z} \times \mathbf{Z} \times \mathbf{Z} \times \mathbf{Z} \to \mathbf{Z} \times \mathbf{Z} \times \mathbf{Z} \times \mathbf{Z}}$$

6 Example: GHZ state, Creation and Unentangling

To demonstrate how we can track the possibly entangling and disentangling property of the CNOT gate, we can look at the example of creating the GHZ state $|000\rangle + |111\rangle$ starting from $|000\rangle$ and then disentangling it. A similar example was considered by Honda [9] to demonstrate how his system can track when *CNOT* displays either its entangling or un-entangling behaviour. One crucial difference is that Honda uses the denotational semantics of density matrices which, in practice, would scale badly with the size of the program being type checked. Our approach is closer to that of Perdrix [14, 15] in terms of design and scalability but capable of showing separability where the prior systems could not.

We will consider the following GHZ program acting on the initial state $\mathbf{Z} \times \mathbf{Z} \times \mathbf{Z} = \mathbf{Z}_1 \cap \mathbf{Z}_2 \cap \mathbf{Z}_3$. We first follow the derivation for $\mathbf{Z}_1$:

```
Definition GHZ :=
  INIT ;      Z ⊗ I ⊗ I   (* initial state *)
  H 1;        X ⊗ I ⊗ I
  CNOT 1 2;   X ⊗ X ⊗ I   (* Bell Pair *)
  CNOT 2 3:   X ⊗ X ⊗ X   (* GHZ State created *)
```

Repeating the derivation for Z_2 and Z_3, We obtain the following type:

$$\texttt{GHZ} : (\mathbf{Z}_1 \to \mathbf{X} \otimes \mathbf{X} \otimes \mathbf{X}) \cap (\mathbf{Z}_2 \to \mathbf{Z} \otimes \mathbf{Z} \otimes \mathbf{I}) \cap (\mathbf{Z}_3 \to \mathbf{I} \otimes \mathbf{Z} \otimes \mathbf{Z})$$

This type is non-trivial to read, but we can clearly see that entanglement is produced between the three qubits.

If we now apply $\texttt{CNOT 2 1}$, we get the following type:

$$\texttt{GHZ; CNOT 2 1} : (\mathbf{Z}_1 \to \mathbf{I} \otimes \mathbf{X} \otimes \mathbf{X}) \cap (\mathbf{Z}_2 \to \mathbf{Z} \otimes \mathbf{I} \otimes \mathbf{I}) \cap (\mathbf{Z}_3 \to \mathbf{I} \otimes \mathbf{Z} \otimes \mathbf{Z})$$

We can reduce the output type to $\mathbf{Z} \times (\mathbf{X} \otimes \mathbf{X} \cap \mathbf{Z} \otimes \mathbf{Z})$, showing that we've neatly separated the first qubit from the Bell pair on qubits 2 and 3 in what used to be a GHZ state.

If we then apply $\texttt{CNOT 3 2}$ we get

$$\texttt{GHZ; CNOT 2 1; CNOT 3 2} : (\mathbf{Z}_1 \to \mathbf{I} \otimes \mathbf{I} \otimes \mathbf{X}) \cap (\mathbf{Z}_2 \to \mathbf{Z} \otimes \mathbf{I} \otimes \mathbf{I}) \cap (\mathbf{Z}_3 \to \mathbf{I} \otimes \mathbf{Z} \otimes \mathbf{I})$$

resulting in the fully separable state $\mathbf{Z} \times \mathbf{Z} \times \mathbf{X}$.

7 Beyond the Clifford Group

Universal quantum computation requires that we use gates outside the Clifford set, the two most studied candidates being the T $(\pi/8)$ and Toffoli gates. In order to type T and Toffoli, we will need to add a

new $\top$ type, which describes any basis state in the state interpretation of our types, and any unitary in the Heisenberg interpretation. We can then give the following type to the T gate:

$$T : \mathbf{Z} \to \mathbf{Z}$$
$$T : \mathbf{X} \to \top$$

Here $\top$ really is a top type. Unlike $\mathbf{I}$, which behaves like an identity, $\top$ behaves like an annihilator:

$$\forall A, \mathbf{I}A = A = A\mathbf{I}$$
$$\forall A, \top A = \top = A\top$$

Instead of explicitly giving a type to the Toffoli gate, we can derive it from Toffoli's standard decomposition into T, H and $CNOT$ gates:

```
Definition TOFFOLI a b c :=
  H c;
  CNOT b c; T† c;
  CNOT a c; T c;
  CNOT b c; T† c;
  CNOT a c; T b; T c;
  H c;
  CNOT a b; T a; T† b;
  CNOT a b.
```

We first note that $T^\dagger$ is simply seven consecutive Ts, so like T, it preserves $\mathbf{Z}$ and takes $\mathbf{X}$ to $\top$. Now consider TOFFOLI's action on $\mathbf{I}\otimes\mathbf{I}\otimes\mathbf{X}$. We can annotate the program with the types after every line:

```
Definition TOFFOLI a b c :=
  INIT ;                 I ⊗ I ⊗ X
  H c ;                  I ⊗ I ⊗ Z
  CNOT b c; T† c;        I ⊗ Z ⊗ Z
  CNOT a c; T c;         Z ⊗ Z ⊗ Z
  CNOT b c; T† c;        Z ⊗ I ⊗ Z
  CNOT a c; T b; T c;    I ⊗ I ⊗ Z
  H c;                   I ⊗ I ⊗ X
  CNOT a b; T a; T† b;   I ⊗ I ⊗ X
  CNOT a b.              I ⊗ I ⊗ X
```

The T's here are all identities, so only the behaviors of H (which takes $\mathbf{X}$ to $\mathbf{Z}$ and back) and $CNOT$ (which takes $\mathbf{I}\otimes\mathbf{Z}$ to $\mathbf{Z}\otimes\mathbf{Z}$ and back) are relevant.

The derivations for $\mathbf{Z}\otimes\mathbf{I}\otimes\mathbf{I}$ and $\mathbf{I}\otimes\mathbf{Z}\otimes\mathbf{I}$ are similar, and the other three states map to the top element:

$$\texttt{TOFFOLI}: \mathbf{Z}\otimes\mathbf{I}\otimes\mathbf{I} \to \mathbf{Z}\otimes\mathbf{I}\otimes\mathbf{I} \qquad \texttt{TOFFOLI}: \mathbf{X}\otimes\mathbf{I}\otimes\mathbf{I} \to \top\otimes\top\otimes\top$$
$$\texttt{TOFFOLI}: \mathbf{I}\otimes\mathbf{Z}\otimes\mathbf{I} \to \mathbf{I}\otimes\mathbf{Z}\otimes\mathbf{I} \qquad \texttt{TOFFOLI}: \mathbf{I}\otimes\mathbf{X}\otimes\mathbf{I} \to \top\otimes\top\otimes\top$$
$$\texttt{TOFFOLI}: \mathbf{I}\otimes\mathbf{I}\otimes\mathbf{Z} \to \top\otimes\top\otimes\top \qquad \texttt{TOFFOLI}: \mathbf{I}\otimes\mathbf{I}\otimes\mathbf{X} \to \mathbf{I}\otimes\mathbf{I}\otimes\mathbf{X}$$

Using our technique for deriving judgements about separability, we can further derive the following type for the Toffoli gate

$$\texttt{TOFFOLI}: \mathbf{Z}\times\mathbf{Z}\times\mathbf{X} \to \mathbf{Z}\times\mathbf{Z}\times\mathbf{X}$$

which says that if you feed a Toffoli three separable qubits in the $\mathbf{Z}$, $\mathbf{Z}$ and $\mathbf{X}$ basis, you receive three qubits in the same basis back.

8 Applications and Future Work

We think that this type system, along with possible extensions, is broadly applicable. Here we outline some of the possible uses of the type system along with (where necessary) extensions that will enable these uses.

Stabilizer Types and Quantum Error Correction This aim of this paper is to define types for unitary operations, however in §3 we interpreted types such as $\mathbf{X}\otimes\mathbf{X}$ and $\mathbf{X}\otimes\mathbf{I}$ as being inhabited by certain states. Gates and circuits, by acting on such states, obtain an arrow type. But as we vary the input states on which unitaries act, they in fact obtain many different arrow types. We used the concept of intersection types to characterize this phenomenon.

However the nature of Pauli operators allows for a different treatment using the stabilizer formalism [7]. Namely two tensor products of Pauli operators (of the same arity) must either commute or anticommute. Consequently, any collection of such types with a joint eigenspace (and hence having a nonempty intersection type) necessarily pairwise commutes. Then, in fact, any product of these operators will also share this eigenspace and would serve equally well in describing the intersection type.

For example, $X\otimes X$ and $X\otimes I$ commute and the intersection type $(\mathbf{X}\otimes\mathbf{X})\cap(\mathbf{X}\otimes\mathbf{I})$ is nonempty, being $\{|{+}{+}\rangle, |{+}{-}\rangle, |{-}{+}\rangle, |{-}{-}\rangle\}$. These states are also eigenstates of $I\otimes X = (X\otimes X)(X\otimes I)$ and hence also of type $\mathbf{I}\otimes\mathbf{X} = (\mathbf{X}\otimes\mathbf{X})*(\mathbf{X}\otimes\mathbf{I})$. Consequently our intersection type could be equally well presented as $(\mathbf{X}\otimes\mathbf{X})\cap(\mathbf{I}\otimes\mathbf{X})$ or $(\mathbf{I}\otimes\mathbf{X})\cap(\mathbf{X}\otimes\mathbf{I})$. And so we should not think of the intersection type $(\mathbf{X}\otimes\mathbf{X})\cap(\mathbf{X}\otimes\mathbf{I})$ being determined by $\mathbf{X}\otimes\mathbf{X}$ and $\mathbf{X}\otimes\mathbf{I}$ alone, but rather by all the elements in the (commutative) group they generate.

Such a commutative group is typically called a stabilizer group. Formally, a stabilizer group is any commutative group of tensor products of Pauli operators some common arity n that does not contain $-I^{\otimes n}$. This latter condition ensures the joint $+1$-eigenspace of all the operators in a stabilizer group exists, which then is called the stabilizer code associated to the group [6]. The eigenspaces associated with other eigenvalue combinations are called syndrome spaces. Therefore our intersection type (presuming it does not include $-\mathbf{I}^n$) is realized as a stabilizer code and its associated syndrome spaces, hence capturing the notion of a Pauli frame [10].

The Gottesman semantics of §2 then can be interpreted as type theory for stabilizer groups and stabilizer codes, which is the direction taken by [9]. This provides a type theory for stabilizer codes that includes encoding and decoding [3], and syndrome extraction [19] circuits. Potential other applications could include analyzing circuits that implement non-Clifford gates on stabilizer codes [13, 21] and circuits that fault-tolerantly switch between codes [4].

Beyond application to quantum error correction, treating intersection types through stabilizer groups offers new deduction rules. In the analysis of the GHZ state of §6, the type of `GHZ; CNOT 1 3` has as codomain the intersection type

$$(\mathbf{X}\otimes\mathbf{X}\otimes\mathbf{I})\cap(\mathbf{Z}\otimes\mathbf{Z}\otimes\mathbf{I})\cap(\mathbf{Z}\otimes\mathbf{Z}\otimes\mathbf{Z}).$$

Given the deduction rules of fig. 1 alone, proving separability of this type is impossible. Yet through the stabilizer formalism, one recognizes it also has the type $(\mathbf{I}\otimes\mathbf{I}\otimes\mathbf{Z}) = (\mathbf{Z}\otimes\mathbf{Z}\otimes\mathbf{I})*(\mathbf{Z}\otimes\mathbf{Z}\otimes\mathbf{Z})$, and so we may represent our intersection type equivalently as

$$(\mathbf{X}\otimes\mathbf{X}\otimes\mathbf{I})\cap(\mathbf{Z}\otimes\mathbf{Z}\otimes\mathbf{I})\cap(\mathbf{I}\otimes\mathbf{I}\otimes\mathbf{Z}) = ((\mathbf{X}\otimes\mathbf{X})\cap(\mathbf{Z}\otimes\mathbf{Z}))\times\mathbf{Z}.$$

Adding measurement It's challenging to turn Gottesman's semantics for measurement into a type system in light of the fact that it looks at the operation on all the basis states, rather than simply the evolution of a single Pauli operator. Namely it adds significant computational complexity, where type-checking should be linear. Nonetheless, the action of measurement on stabilizer groups is understood [7]. With the link between intersection types and stabilizers established above, we can see the potential for realizing such a type system. The challenge lies in the fact that to execute the action of measurement on a stabilizer group, one needs to find a set of generators of the group in a particular form. In our grammar, we need to express an intersection type using a special presentation, which requires the introduction of rewrite rules like those we just used in the analysis of the GHZ state.

Formally measurement acts as follows; for ease of exposition, suppose we are measuring the first qubit in the computational basis (a Z-basis measurement). From our initial intersection, do the following:

1. Use the rewrite rules to ensure there are only 0 or 1 terms in the intersection that have $\mathbf{X}$ in the first position. If there is 1 such term, then remove it from the intersection.
2. If there are 0 terms that have $\mathbf{X}$ in the first position, use the rewrite rules to ensure there are only 0 or 1 terms that have $\mathbf{Z}$ in the first position. If there is 1 such term, remove it from the intersection.
3. Add $\mathbf{Z} \otimes \mathbf{I}^{n-1}$ to the intersection.

The resulting intersection type is the type after measurement.

For example in our analysis of the GHZ state in §6, the circuit `GHZ` had a codomain of type

$$(\mathbf{X} \otimes \mathbf{X} \otimes \mathbf{X}) \cap (\mathbf{Z} \otimes \mathbf{Z} \otimes \mathbf{I}) \cap (\mathbf{I} \otimes \mathbf{Z} \otimes \mathbf{Z}).$$

To compute the type of `GHZ; MEAS 1` we enact the above program. Fortunately our intersection already has the requisite form, with the first term being the only one with an $\mathbf{X}$ in the initial position. We remove the term and add $\mathbf{Z} \otimes \mathbf{I} \otimes \mathbf{I}$ to get

$$(\mathbf{Z} \otimes \mathbf{I} \otimes \mathbf{I}) \cap (\mathbf{Z} \otimes \mathbf{Z} \otimes \mathbf{I}) \cap (\mathbf{I} \otimes \mathbf{Z} \otimes \mathbf{Z}).$$

Using these same rewrite rules, we can replace the second term with $\mathbf{I} \otimes \mathbf{Z} \otimes \mathbf{I}$ and with that term replace the last with $\mathbf{I} \otimes \mathbf{I} \otimes \mathbf{Z}$, producing the output type as

$$(\mathbf{Z} \otimes \mathbf{I} \otimes \mathbf{I}) \cap (\mathbf{Z} \otimes \mathbf{I} \otimes \mathbf{I}) \cap (\mathbf{I} \otimes \mathbf{I} \otimes \mathbf{Z}) = \mathbf{Z} \times \mathbf{Z} \times \mathbf{Z}.$$

At first glance, it is not clear that the program for measurement above is well-defined. Why should such a presentation for a general intersection type exist? Even if one does, how do we find one among all possible rewrites? Fortunately neither of the concerns is an issue. It is well known that stabilizer groups have a denotational semantics as binary matrices of dimension linear in the number of qubits, see for example [12, §10.5]. The existence of such a presentation, and a procedure to find one, reduces to straightforward linear algebra. Hence the program above for measurement can be accomplished in at worst quadratic time in the number of qubits.

Resource Tracking Resource monotones track the amount of resources contained in a type. For instance, at a very coarse level, one might quantify the entanglement in a state by counting the number of "ebits" needed to create the state. In our formalism, a Bell state is of type $\mathbf{E} = (\mathbf{X} \otimes \mathbf{X}) \cap (\mathbf{Z} \otimes \mathbf{Z})$, which is counted as having one ebit. A seperable type such as $\mathbf{X} \times \mathbf{Z}$ has a resource value 0 and hence no ebits. While the current system cannot handle such resource monotones, it seems plausible that using a suitable extension could similarly calculate the resource cost of various operations. Say, by counting the

number of **E** types used in a protocol. Then, superdense coding has an **E**-count of 1 while entanglement swapping will have an **E**-count of 2.

Our types are too fine to capture the notion of local equivalence [11]. Instead, one can coarsen to types generated from graph states [5]. For example, all six entangled two qubit types are equivalent under the local operations of $H \otimes I$, $S \otimes I$, $I \otimes H$, and $I \otimes S$, and therefore, from an entanglement monotone perspective, they all contain the same amount of entanglement.

One can show that for three qubits, there are only two classes of entangled types, up to relabeling of the qubit indices [2]. Such computations are quite challenging at scale, and so methods to quantify the amount of entanglement using such states is a long-term goal of this project.

Ancilla correctness. Many quantum circuits introduce ancillary qubits that are used to perform some classical computation and are then discarded in a basis state. Several efforts have been made to verify this behaviour: The Quipper [8] and Q# [20] languages allow us to *assert* that ancilla are separable and can be safely discarded, while $\mathscr{Q}$WIRE allows us to manually verify this [17]. More recently, Silq [1] allows us to define "qfree" functions that never put qubits into a superposition. We hope to avoid this restriction and use our type system to automatically guarantee ancilla correctness by showing that the ancillae are in **Z** and separable.

Provenance tracking Another useful addition to the type system is *ownership*. Superdense coding is a central example of a class of quantum communication protocols. By annotating the typing judgements with ownership information and restricting multiqubit operations to qubits under the same ownership, we can guarantee that superdense coding only transmits a single qubit, via a provided channel $\mathscr{C}$. (Note that measurement and ownership types are both forms of static information-flow control [18].) With this additional typing information, we can give superdense coding the type

$$\mathbf{Z}_A \times \mathbf{Z}_A \times \mathbf{Z}_A \times \mathbf{Z}_B \to \mathbf{Z}_A \times \mathbf{Z}_A \times \mathbf{Z}_B \times \mathbf{Z}_B$$

indicating that a single qubit has passed from Alice's control to Bob's.

Acknowledgements

The first author would like to acknowledge the support of the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Quantum Testbed Pathfinder Program under Award Number DE-SC0019040. The third author was funded by EPiQC, an NSF Expedition in Computing, under grant CCF-1730449.

References

[1] Benjamin Bichsel, Maximilian Baader, Timon Gehr & 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 (PLDI '20)*. To appear.

[2] Sergey Bravyi, David Fattal & Daniel Gottesman (2006): *GHZ extraction yield for multipartite stabilizer states*. *J. Math. Phys.* 47(6), p. 062106, doi:10.1063/1.2203431. Available at `https://arxiv.org/abs/quant-ph/0504208`.

[3] Richard Cleve & Daniel Gottesman (1997): *Efficient Computations of Encodings for Quantum Error Correction*. *Phys. Rev. A* 56, pp. 76–82, doi:10.1103/PhysRevA.56.76. Available at `https://arxiv.org/abs/quant-ph/9607030`.

[4] Kristina R Colladay & Erich J Mueller (2018): *Rewiring Stabilizer Codes*. New Journal of Physics 20(8), p. 083030, doi:10.1088/1367-2630/aad8dd. Available at https://arxiv.org/abs/1707.09403.

[5] David Fattal, Toby S Cubitt, Yoshihisa Yamamoto, Sergey Bravyi & Isaac L Chuang (2004): *Entanglement in the stabilizer formalism*. arXiv preprint. Available at https://arxiv.org/abs/quant-ph/0406168.

[6] Daniel Gottesman (1996): *Class of Quantum Error-Correcting Codes Saturating the Quantum Hamming Bound*. Physical Review A 54(3), p. 1862–1868, doi:10.1103/physreva.54.1862. Available at https://arxiv.org/abs/quant-ph/9604038.

[7] Daniel Gottesman (1998): *The Heisenberg Representation of Quantum Computers*. In: Group22: Proceedings of the XXII International Colloquium on Group Theoretical Methods in Physics, pp. 32–43. Available at https://arxiv.org/abs/quant-ph/9807006.

[8] Alexander S. Green, Peter LeFanu Lumsdaine, Neil J. Ross, Peter Selinger & Benoît Valiron (2013): *Quipper: A Scalable Quantum Programming Language*. In: Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '13), pp. 333–342, doi:10.1145/2491956.2462177. Available at https://arxiv.org/abs/1304.3390.

[9] Kentaro Honda (2015): *Analysis of Quantum Entanglement in Quantum Programs using Stabilizer Formalism*. In: Proc. QPL '15, pp. 262–272, doi:10.4204/EPTCS.195.19.

[10] Emanuel Knill (2005): *Quantum computing with realistically noisy devices*. Nature 434(7029), pp. 39–44, doi:10.1038/nature03350. Available at https://arxiv.org/abs/quant-ph/0410199.

[11] Maarten Van den Nest, Jeroen Dehaene & Bart De Moor (2005): *Local unitary versus local Clifford equivalence of stabilizer states*. Phys. Rev. A 71, p. 062323, doi:10.1103/PhysRevA.71.062323. Available at https://arxiv.org/abs/quant-ph/0411115.

[12] Michael A. Nielsen & Isaac L. Chuang (2010): *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, doi:10.1017/CBO9780511976667.

[13] Adam Paetznick & Ben W Reichardt (2013): *Universal fault-tolerant quantum computation with only transversal gates and error correction*. Physical review letters 111(9), p. 090505, doi:10.1103/PhysRevLett.111.090505. Available at https://arxiv.org/abs/1304.3709.

[14] Simon Perdrix (2007): *Quantum Patterns and Types for Entanglement and Separability*. Electronic Notes in Theoretical Computer Science 170, pp. 125–138, doi:10.1016/j.entcs.2006.12.015. Proc. QPL '05.

[15] Simon Perdrix (2008): *Quantum Entanglement Analysis Based on Abstract Interpretation*. In: Proc. 15th International Static Analysis Symposium, pp. 270–282, doi:10.1007/978-3-540-69166-2_18. Available at https://arxiv.org/abs/0801.4230.

[16] Benjamin C. Pierce (2002): *Types and Programming Languages*. MIT Press.

[17] Robert Rand, Jennifer Paykin, Dong-Ho Lee & Steve Zdancewic (2018): *ReQWIRE: Reasoning about Reversible Quantum Circuits*. In: Proc. QPL '18, pp. 299–312, doi:10.4204/EPTCS.287.17.

[18] A. Sabelfeld & A. C. Myers (2006): *Language-based Information-flow Security*. IEEE J. Sel. Areas Commun. 21(1), pp. 5–19, doi:10.1109/JSAC.2002.806121. Available at https://www.cs.cornell.edu/andru/papers/jsac/sm-jsac03.pdf.

[19] Andrew M Steane (1997): *Active stabilization, quantum computation, and quantum state synthesis*. Physical Review Letters 78(11), p. 2252, doi:10.1103/PhysRevLett.78.2252. Available at https://arxiv.org/abs/quant-ph/9611027.

[20] Krysta Svore, Alan Geller, Matthias Troyer, John Azariah, Christopher Granade, Bettina Heim, Vadym Kliuchnikov, Mariia Mykhailova, Andres Paz & Martin Roetteler (2018): *Q#: Enabling Scalable Quantum Computing and Development with a High-level DSL*. In: Proc. Real World Domain Specific Languages Workshop (RWDSL) 2018, pp. 7:1–7:10, doi:10.1145/3183895.3183901. Available at https://arxiv.org/abs/1803.00652.

[21] Theodore J Yoder (2017): *Universal fault-tolerant quantum computation with Bacon-Shor codes*. arXiv preprint. Available at https://arxiv.org/abs/1705.01686.