# 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 (§4). In §5, 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 §6.

The system and examples in this paper are formalized in Coq in the Heisenberg branch of the SQIR [9] repository:

`https://github.com/inQWIRE/SQIR/blob/Heisenberg/core/HeisenbergSem.v`

Submitted to:
QPL 2020

## 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]. 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'll implicitly compose arrows by using multiple arrows in sequence:

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

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:

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

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

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

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

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

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.

## 3 Interpretation on Basis States

We can interpret the type $\mathbf{Z} \to \mathbf{X}$ 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 [14] 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 leaving the Clifford circuits, see §5.) 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 $\lambda$ 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 $\lambda$. □

**The role of I** $\mathbf{I}$ plays an interesting role in this type system. For $\mathbf{I}$ 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 $\mathbf{I}$ as a kind of top type, where $A \leq \mathbf{I}$ for any type $A$. However, this would be incompatible with $\otimes$. For example, if we consider the two qubit Bell pair $|\Phi^+\rangle$, this has type $\mathbf{X} \otimes \mathbf{X}$ but not type $\mathbf{X} \otimes \mathbf{I}$. Namely $\mathbf{X} \otimes \mathbf{I}$ contains precisely the separable two qubit states where the first qubit is an eigenstate of $\mathbf{X}$, so the neither type is a subtype of the other. This generalizes to $n$ qubit systems; for ease of reading we consider $\mathbf{X} \otimes \mathbf{I}^{\otimes n-1}$ but the result easily extends to the case where the single non-$\mathbf{I}$ term appears in any factor.

**Proposition 2.** *For any Pauli matrix $U \in \{\pm X, \pm Y, \pm Z\}$, 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 $+1$ eigenstate for $U$ and $\left|\phi^{\perp}\right\rangle$ be the $-1$ eigenstate for $U \in \{\pm X, \pm Y, \pm Z\}$. 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 $(U \otimes I^{\otimes n-1}) |\delta\rangle = \lambda |\delta\rangle$. 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

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

where we have used that $|\phi\rangle$ and $\left|\phi^{\perp}\right\rangle$ are the $+1$ and $-1$ eigenvalues of $U$ respectively. As the components of the expansion are orthogonal to each other, $\lambda$ must satisfy:

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

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

Combining Propositions 1 and 2, we obtain the following corollary:

**Corollary 3.0.1.** *Every term of type $U \otimes \mathbf{I}^{\otimes n-1}$ is separable, for $U \in \{\pm X, \pm Y, \pm Z\}$. That is, the factor associated with $U$ is unentangled with the rest of the system.*

In this light, $CNOT : \mathbf{Z} \otimes \mathbf{X} \to \mathbf{Z} \otimes \mathbf{X}$ actually has a weaker type than we desire. Given that it emerges from $CNOT : \mathbf{I} \otimes \mathbf{X} \to \mathbf{I} \otimes \mathbf{X}$ and $CNOT : \mathbf{Z} \otimes \mathbf{I} \to \mathbf{Z} \otimes \mathbf{I}$, we can introduce a $\times$ operator for both of these separable cases and deduce that $CNOT : \mathbf{Z} \times \mathbf{X} \to \mathbf{Z} \times \mathbf{X}$, i.e. given a separable input in the appropriate bases, the output is likewise separable.

Revisiting *SWAP*, we have $SWAP : \mathbf{X} \times \mathbf{I} \to \mathbf{I} \times \mathbf{X}$ and $SWAP : \mathbf{I} \times \mathbf{Z} \to \mathbf{Z} \times \mathbf{I}$ (immediately from the corresponding $\otimes$ types and the lemma above) and hence can derive $SWAP : \mathbf{X} \times \mathbf{Z} \to \mathbf{Z} \times \mathbf{X}$.

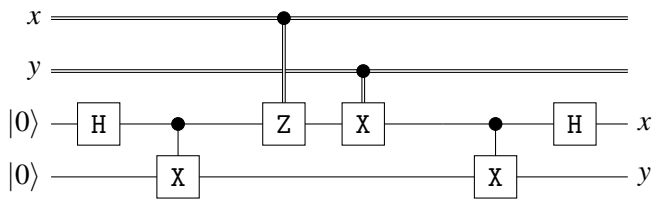


Figure 1: Superdense Coding sending classical bits $x$ and $y$ from Alice to Bob.

## 4 Example: Superdense Coding

To illustrate the power of this simple system, consider the example of superdense coding as in fig. 1. Superdense coding allows Alice to convey two bits of information $x$ and $y$, which we treat as qubits in the **Z** state, to Bob by sending a single qubit and consuming one EPR pair.

We can type this as follows, annotating each step of the program with its types on the right. We treat $x$ and $y$ as qubits in the **Z** basis and likewise for the $|0\rangle$ inputs:

```
Definition superdense :=
  INIT ;     Z ⊗ Z ⊗ Z ⊗ Z   (* initial state *)
  H 2  ;     Z ⊗ Z ⊗ X ⊗ Z
  CNOT 2 3 ; Z ⊗ Z ⊗ XZ ⊗ XZ (* Bell pair *)
  CZ 0 2 ;   I ⊗ Z ⊗ XZ ⊗ XZ
  CNOT 1 2 ; I ⊗ I ⊗ XZ ⊗ XZ (* map bits onto bell pair *)
  CNOT 2 3 ; I ⊗ I ⊗ X ⊗ Z
  H 2        I ⊗ I ⊗ Z ⊗ Z   (* decode qubits *)
```

The most difficult steps of this derivation are applications of the *CZ* and subsequent *CNOT*, but even these are relatively straightforward: We simply combine three basic types for CZ, $\mathbf{Z}\otimes\mathbf{I}\rightarrow\mathbf{Z}\otimes\mathbf{I}$, $\mathbf{I}\otimes\mathbf{X}\rightarrow\mathbf{Z}\otimes\mathbf{X}$, and $\mathbf{I}\otimes\mathbf{Z}\rightarrow\mathbf{I}\otimes\mathbf{Z}$, yielding $\mathbf{ZIZ}\rightarrow\mathbf{IXZ}$ or $\mathbf{I}\rightarrow\mathbf{XZ}$, and the *CNOT* works identically. Given that every $k$-qubit gate requires us to combine up to $2k$ rules, if $k$ is bounded, the type-inference time is linear in the number of gates. (If we provide gates of arbitrary arity, it becomes $g*q$, where $g$ is the number of gates and $q$ is the number of qubits or circuit width.)

As we would hope, the outcome of this computation shows that the two qubits are in the **Z** basis. This is a useful sanity check for our circuit: If the final qubits were in a different basis, we would know that we had implemented superdense coding incorrectly!

## 5 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:

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

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 $T$s, 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:

$$\begin{array}{ll} TOFFOLI: & \mathbf{Z}\otimes\mathbf{I}\otimes\mathbf{I}\to\mathbf{Z}\otimes\mathbf{I}\otimes\mathbf{I} \\ TOFFOLI: & \mathbf{I}\otimes\mathbf{Z}\otimes\mathbf{I}\to\mathbf{I}\otimes\mathbf{Z}\otimes\mathbf{I} \\ TOFFOLI: & \mathbf{I}\otimes\mathbf{I}\otimes\mathbf{Z}\to\top\otimes\top\otimes\top \end{array} \qquad \begin{array}{ll} TOFFOLI: & \mathbf{X}\otimes\mathbf{I}\otimes\mathbf{I}\to\top\otimes\top\otimes\top \\ TOFFOLI: & \mathbf{I}\otimes\mathbf{X}\otimes\mathbf{I}\to\top\otimes\top\otimes\top \\ TOFFOLI: & \mathbf{I}\otimes\mathbf{I}\otimes\mathbf{X}\to\mathbf{I}\otimes\mathbf{I}\otimes\mathbf{X} \end{array}$$

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

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

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.

**Adding Measurement** Adding measurement in Gottesman's system is difficult as it is not a linear operation, meaning that we cannot simply add a corresponding type for measurement to the system. However, we do know that applying a $Z$ rotation before measurement is a no-op, so we can give the following direct types to measurement:

$$Meas: \mathbf{Z} \to \mathbf{I}$$
$$Meas: \mathbf{X} \to \top$$

which is akin to the types we gave to $T$ earlier.

# 6 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 paper focused on defining 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. While these two types in particular are not directly comparable, they do have states in common. A direct corollary of Proposition 2 is: if two types have states in common then their associated Pauli operators commute.

Thus we could view the states in both $\mathbf{X} \otimes \mathbf{X}$ and $\mathbf{X} \otimes \mathbf{I}$ as having a new type $\langle \mathbf{X} \otimes \mathbf{X},\ \mathbf{X} \otimes \mathbf{I} \rangle$, as explained here. These states are precisely the joint eigenstates of both $X \otimes X$ and $X \otimes I$, and hence eigenstates of all Pauli operators in the Abelian group $\langle X \otimes X,\ X \otimes I \rangle$. We write $\langle \mathbf{X} \otimes \mathbf{X},\ \mathbf{X} \otimes \mathbf{I} \rangle$ for the collection of such states. An Abelian group of this form is typically called a stabilizer group; formally, a stabilizer group is any Abelian group of Pauli operators on $n$ qubits not containing $-I^{\otimes n}$. The joint $+1$-eigenspace of all the operators of a stabilizer groups is referred to as its stabilizer code [6]. The eigenspaces associated with other eigenvalue combinations are called syndrome spaces. Therefore the "stabilizer types" formed as above consist precisely of a stabilizer code and its associated syndrome spaces, hence capturing the notion of a Pauli frame [11].

The Gottesman semantics of §2 lifts to stabilizer types canonically by evaluating the operation on each generator of the stabilizer type. This provides a type theory for stabilizer codes that includes encoding and decoding [3], and syndrome extraction [17] circuits. Potential other applications could include analyzing circuits that implement non-Clifford gates on stabilizer codes [13, 19] and circuits that fault-tolerantly switch between codes [4].

**Tracking Entanglement and Separability** In §3, we gave an example of typing judgements for separable states. Unfortunately, this has limited use as most types contain both entangled and separable states. To make a more precise statements about when *CNOT* may introduce or destroy entanglement we introduced the operator $\times$ on separable states. Namely we set $\mathbf{Z} \times \mathbf{X}$ for the type of separable states formed from $\mathbf{I} \otimes \mathbf{X}$ and $\mathbf{Z} \otimes \mathbf{I}$. Using stabilizer types we can rigorously formalize this operation by setting $\mathbf{Z} \times \mathbf{X} = \langle \mathbf{I} \otimes \mathbf{X},\ \mathbf{Z} \otimes \mathbf{I} \rangle$. In generality, given an $n$ qubit type $T_1$ and an $m$ qubit type $T_2$ we set $T_1 \times T_2 = \langle \mathbf{I}^{\otimes n} \otimes T_2,\ T_1 \otimes \mathbf{I}^{\otimes m} \rangle$. If $T_1$ or $T_2$ are themselves stabilizer types, we extend this by using the

generators of these types. Then the deduction $CNOT : \mathbf{Z} \times \mathbf{X} \to \mathbf{Z} \times \mathbf{X}$ from §3 follows immediately from the definition of $\mathbf{Z} \times \mathbf{X}$ and the calculations $CNOT : \mathbf{I} \times \mathbf{X} \to \mathbf{I} \times \mathbf{X}$ and $CNOT : \mathbf{Z} \times \mathbf{I} \to \mathbf{Z} \times \mathbf{I}$.

Consider for example Gottesman [7], who shows that we can derive the circuit for constructing a Bell pair by considering the eigenstates of $\mathbf{I} \otimes \mathbf{I}$, $\mathbf{X} \otimes \mathbf{X}$, $\mathbf{Z} \otimes \mathbf{Z}$, and $-\mathbf{Y} \otimes \mathbf{Y}$. In particular one finds that the Bell pair is of type $\mathbf{E} = \langle \mathbf{X} \otimes \mathbf{X},\ \mathbf{Z} \otimes \mathbf{Z} \rangle$, which exists only as a stabilizer type. Specifically, one finds there are precisely fifteen two qubit stabilizer types, nine of which are of the form $P_1 \times P_2$ for $P_1, P_2 \in \{\mathbf{X}, \mathbf{Y}, \mathbf{Z}\}$. The type $\mathbf{E}$ is an example of a type not of this form and, like the other five stabilizer types not of this form, consists entirely of entangled states. Using these types, one can specify precisely on which stabilizer states *CNOT* is entangling and disentangling.

In [10], an equivalent formulation of stabilizer types is presented. In that system, the type $\mathbf{I} \otimes \mathbf{X}$ would be represented as $(\{2\}, \mathbf{X})$ while $\mathbf{Z} \otimes \mathbf{I}$ would be $(\{1\}, \mathbf{Z})$. Thus the type of separable states $\mathbf{Z} \times \mathbf{X}$ is equivalently denoted as $\{(\{1\}, \mathbf{Z}),\ (\{2\}, \mathbf{X})\}$ while $\mathbf{Z} \otimes \mathbf{X}$, which contains entangled states, is denoted as $(\{1,2\}, \mathbf{Z} \otimes \mathbf{X})$.

**Resource Tracking** Resource monotones track the amount of resources contained in a type. For instance, at a very coarse level, one might quantify the amount of entanglement in a state by counting the number of ebits needed to create the state. In this case, a product state has a resource value 0 while an EPR pair has value 1. While the current system cannot handle such resource monotones, it seems plausible that using a suitable extension of the stabilizer types discussed previously, we can similarly calculate the resource cost of various operations. Say, by counting the number of $\mathbf{E}$ types used in a protocol. Then, superdense coding has an $\mathbf{E}$-count of 1 while entanglement swapping will have an $\mathbf{E}$-count of 2.

Stabilizer types are too fine to capture the notion of local equivalence [12]. Instead, one can coarsen to types generated from graph states [5]. For example, all six entangled stabilizer 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 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 in this direction: The Quipper [8] and Q# [18] languages allow us to *assert* that ancilla are separable and can be safely discarded, while $\mathscr{Q}$WIRE allows us to manually verify this [15]. 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 $\mathbf{Z}$ 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 [16].) With this additional typing information, we can give superdense coding the type

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

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

# 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*. *Journal of mathematical physics* 47(6), p. 062106.

[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.

[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.

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

[6] Daniel Gottesman (1996): *Class of quantum error-correcting codes saturating the quantum Hamming bound*. *Physical Review A* 54(3), p. 18621868, doi:10.1103/physreva.54.1862.

[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.

[9] Kesha Hietala, Robert Rand, Shih-Han Hung, Xiaodi Wu & Michael Hicks (2019): *A Verified Optimizer for Quantum Circuits*. *arXiv preprint arXiv:1912.02250*.

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

[11] Emanuel Knill (2005): *Quantum computing with realistically noisy devices*. *Nature* 434(7029), pp. 39–44, doi:10.1038/nature03350.

[12] Maarten Van den Nest, Jeroen Dehaene & Bart De Moor (2005): *Local unitary versus local Clifford equivalence of stabilizer states*. *Physical Review A* 71(6), p. 062323.

[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.

[14] Simon Perdrix (2008): *Quantum Entanglement Analysis Based on Abstract Interpretation*. In: *Proceedings of 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`.

[15] Robert Rand, Jennifer Paykin, Dong-Ho Lee & Steve Zdancewic (2018): *ReQWIRE: Reasoning about Reversible Quantum Circuits*. In: *Proceedings of the 15th International Conference on Quantum Physics and Logic (QPL '18)*, pp. 299–312, doi:10.4204/EPTCS.287.17.

[16] A. Sabelfeld & A. C. Myers (2006): *Language-based Information-flow Security*. *IEEE Journal on Selected Areas in Communications* 21(1), pp. 5–19, doi:10.1109/JSAC.2002.806121.

[17] 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.

[18] 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: *Proceedings of the Real World Domain Specific Languages Workshop 2018 (RWDSL '18)*, pp. 7:1–7:10, doi:10.1145/3183895.3183901.

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