A Type System for Quantum Resources

Robert Rand[1], Aarthi Sundaram[1], Kartik Singhal[2], and Brad Lackey[1,3]

[1]Joint Centre for Quantum Information and Computer Science, University of Maryland
[2]Department of Computer Science, University of Chicago
[3]Quantum Systems Group, Microsoft Quantum, Redmond, WA

1 Introduction

Type systems have long been a central feature of quantum programming languages. The quantum lambda calculus [16] introduced linear types for guaranteeing the no-cloning theorem of quantum mechanics and QWIRE [12, 14] added dependent types for specifying circuit families. However, the basic type in all of these systems is the *qubit* from which we can build pairs or more complex data structures of qubits.

By contrast, quantum resource theories recognize different kinds of quantum data. For instance, LOCC (local operations and classical communication) as the resource theory of entanglement [10, 6] emphasizes the difference between separable states and entangled states. Similarly, the various resource theories of coherence [3, 18] are based on defining classes of channels that preserve incoherent states (and hence do not generate coherence).

In these systems, communication, entanglement and coherence may be treated as resources that can be converted and consumed. An example is quantum teleportation that shows how we can use an EPR pair and two bits to transmit a qubit. This can be thought of as a *proof* that two bits and one *ebit* (the EPR pair) can be converted to one qubit of communication. Similarly, in the superdense coding protocol, which we present in §3 below, two bits of information can be transmitted between two parties using one ebit and one qubit of quantum communication. Consequently the ebit is a good candidate for a *type* in a type system.

In this work, we present a lightweight type system for stabilizer quantum mechanics. This system distinguishes between classical and maximally coherent qubits, as well as separable and maximally entangled qubit pairs. As such, it can act as a lightweight type system for computing with the common Clifford set of quantum gates. Together with a supertype for non-stabilizer states (analogous to Top or Object), this can make useful guarantees about even general quantum programs.

The system we present here is quite limited in that it only talks about a subset of Clifford gates and two qubit entanglement. This restriction is precisely what allows it to efficiently and precisely characterize quantum programs that in the general case have exponential running time. In §§4 and 5, we sketch out how to extend the system with more qubits and gates, while proposing a variety of applications for the current system and its extensions.

2 Types for a Stabilizer Subsystem

Single Qubit Types We will begin by considering two single-qubit states of interest. The type **Z** consists of the eigenvectors of the Pauli Z matrix, which are the computational basis states $|0\rangle$ and $|1\rangle$ up to a global phase. (That is, $e^{i\theta}|0\rangle$ is in **Z** for any θ.) The type **X** is that of the equal superpositions $|+\rangle$ and $|-\rangle$ (the eigenvectors of Pauli X matrix) again up to a global phase.

If we consider the Z, X and H operations, both X and Z take basis states to basis states and equal superpositions to equal superpositions. On the other hand, the Hadamard gate H moves between the types,

taking $\mathbf{Z}$ to $\mathbf{X}$ and $\mathbf{X}$ to $\mathbf{Z}$. Hence we can give our operators the following types:

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

Adding Entanglement In order to type two-qubit gates, for example the controlled-NOT ($CNOT$), we first need to add pair types corresponding to unentangled qubits. We write these as $A \otimes B$ and can similarly write $f \otimes g$ for operations f, g to represent applying f to the first element and g to the second. We also introduce the I operation that acts as the identity, with the type $A \to A$ for any A. Then $\otimes$ is functorial in that if $f : A \to A'$ and $g : B \to B'$ then $f \otimes g : A \otimes B \to A' \otimes B'$.

In order to derive the additional types we need, consider applying $CNOT$ to the possible combinations of $\mathbf{Z}$ and $\mathbf{X}$. If we apply $CNOT$ to $\mathbf{Z} \otimes \mathbf{Z}$ it behaves like a classical XOR on the second qubit, leaving it in one of the basis states. Similarly, $CNOT$ takes $\mathbf{Z} \otimes \mathbf{X}$ to $\mathbf{Z} \otimes \mathbf{X}$. Somewhat more surprisingly, for any $\phi \in \{|{+}{+}\rangle, |{+}{-}\rangle, |{-}{+}\rangle, |{-}{-}\rangle\}$, $CNOT(\phi)$ is still in the same set, meaning $CNOT : \mathbf{X} \otimes \mathbf{X} \to \mathbf{X} \otimes \mathbf{X}$. Finally, applying a $CNOT$ from an equal superposition qubit to a classical qubit yields one of the four well-known Bell pairs, up to a global phase:

$$|\Phi^+\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \quad |\Phi^-\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \quad |\Psi^+\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \quad |\Psi^-\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle).$$

We call the type of these states $\mathbf{E}$ (for EPR pair).

It's not hard to see that applying an X or Z to either element of an EPR pair yields an EPR pair. Applying a $CNOT$ to an EPR pair reverses the entanglement, yielding a term of type $\mathbf{X} \otimes \mathbf{Z}$. It's also important to note that the $\mathbf{E}$ type is symmetric: If $(q_1, q_2) : \mathbf{E}$ then $(q_2, q_1) : \mathbf{E}$, allowing us to perform a $CNOT$ in the other direction.

However, if we apply an H to either qubit of an EPR pair we obtain one of the following four states:

$$\frac{1}{2}(|00\rangle{+}|01\rangle{+}|10\rangle{-}|11\rangle) \quad \frac{1}{2}(|00\rangle{+}|01\rangle{-}|10\rangle{+}|11\rangle) \quad \frac{1}{2}(|00\rangle{-}|01\rangle{+}|10\rangle{+}|11\rangle) \quad \frac{1}{2}(|00\rangle{-}|01\rangle{-}|10\rangle{-}|11\rangle)$$

These correspond to the two-qubit *graph states* [9, 2], which we call $\mathbf{G}$.

Adding $\mathbf{E}$ and $\mathbf{G}$ to the system, we get the typing rules in fig. 1. As the controlled-Z gate is semantically equivalent to $(I \otimes H)CNOT(I \otimes H)$ we can easily obtain the rules for CZ as well. Finally, the rules for measurement in the computational basis are clear for $\mathbf{Z}$, $\mathbf{X}$ and $\mathbf{E}$, and straightforward to verify for $\mathbf{G}$.

$Z : \mathbf{Z} \to \mathbf{Z}$	$Z : \mathbf{X} \to \mathbf{X}$	$Z \otimes I : \mathbf{E} \to \mathbf{E}$	$Z \otimes I : \mathbf{G} \to \mathbf{G}$
$X : \mathbf{Z} \to \mathbf{Z}$	$X : \mathbf{X} \to \mathbf{X}$	$X \otimes I : \mathbf{E} \to \mathbf{E}$	$X \otimes I : \mathbf{G} \to \mathbf{G}$
$H : \mathbf{Z} \to \mathbf{X}$	$H : \mathbf{X} \to \mathbf{Z}$	$H \otimes I : \mathbf{E} \to \mathbf{G}$	$H \otimes I : \mathbf{G} \to \mathbf{E}$

$CNOT : \mathbf{Z} \otimes \mathbf{Z} \to \mathbf{Z} \otimes \mathbf{Z}$	$CNOT : \mathbf{Z} \otimes \mathbf{X} \to \mathbf{Z} \otimes \mathbf{X}$	$CNOT : \mathbf{X} \otimes \mathbf{Z} \to \mathbf{E}$
$CNOT : \mathbf{X} \otimes \mathbf{X} \to \mathbf{X} \otimes \mathbf{X}$	$CNOT : \mathbf{E} \to \mathbf{Z} \otimes \mathbf{X}$	$CNOT : \mathbf{G} \to \mathbf{G}$
$CZ : \mathbf{Z} \otimes \mathbf{Z} \to \mathbf{Z} \otimes \mathbf{Z}$	$CZ : \mathbf{Z} \otimes \mathbf{X} \to \mathbf{Z} \otimes \mathbf{X}$	$CZ : \mathbf{X} \otimes \mathbf{Z} \to \mathbf{X} \otimes \mathbf{Z}$
$CZ : \mathbf{X} \otimes \mathbf{X} \to \mathbf{G}$	$CZ : \mathbf{E} \to \mathbf{E}$	$CZ : \mathbf{G} \to \mathbf{X} \otimes \mathbf{X}$

$meas : \mathbf{Z} \to \mathbf{Z}$	$meas : \mathbf{X} \to \mathbf{Z}$	$meas \otimes I : \mathbf{E} \to \mathbf{Z} \otimes \mathbf{Z}$	$meas \otimes I : \mathbf{G} \to \mathbf{Z} \otimes \mathbf{X}$

Figure 1: The typing rules for two qubits and Z, X, H and $CNOT$. All the rules for one-qubit operations applied to two-qubit systems are symmetric.

3 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.

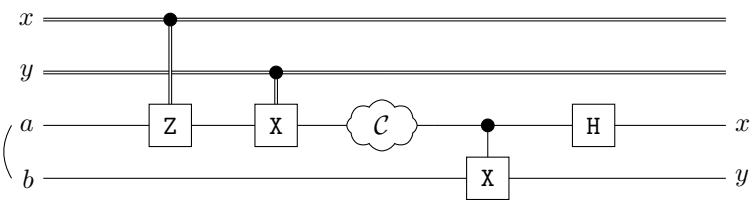

Figure 2: Superdense Coding sending classical bits x and y from Alice to Bob. The channel $\mathcal{C}$ represents Alice transmitting her qubit to Bob.

We can write this out as a simple imperative program with Hoare-style triples representing the types before and after each operation. We treat our *channel* ($\mathcal{C}$), which conveys Alice's qubit to Bob, as perfect and therefore equivalent to a no-op.

$$\begin{array}{lcl} \{x, y : \mathbf{Z};\ (a, b) : \mathbf{E}\} & CZ(x, a);\ CNOT(y, a);\ \mathcal{C}_{A\to B} & \{x, y : \mathbf{Z};\ (a, b) : \mathbf{E}\} \\ \{x, y : \mathbf{Z};\ (a, b) : \mathbf{E}\} & CNOT(a, b) & \{x, y : \mathbf{Z};\ (a, b) : \mathbf{X} * \mathbf{Z}\} \\ \{x, y, b : \mathbf{Z};\ a : \mathbf{X}\} & H(a) & \{x, y, b, a : \mathbf{Z}\} \end{array}$$

This can also be viewed as a simple functional program in which every variable is shadowed: $H(a)$ is equivalent to $a \leftarrow H\ a$. Every typing triple here follows directly from the rules in fig. 1. This proves that our superdense coding program successfully produces two qubits in the $\mathbf{Z}$ basis.

We can further annotate the types with P and M to indicate a substate being *pure* or *mixed*, thereby tracking whenever measurement has introduced non-determinism. In our example this is trivial: We never perform a measurement so if x and y are pure states so are the output qubits.

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 the provided channel $\mathcal{C}$. (Note that measurement and ownership types are both forms of static information-flow control [15].) With this additional typing information, we can produce the typing triple

$$\{x, y : \mathbf{Z}^P_A;\ (a, b) : \mathbf{E}^P_{A,B}\}\ \texttt{superdense(x, y, a, b)}\ \{x, y : \mathbf{Z}^P_A;\ a, b : \mathbf{Z}^P_B\},$$

which states that superdense coding deterministically turns Alice's two classical qubits and shared EPR pair with Bob into two classical qubits for Alice and two for Bob.

4 A check matrix semantics for types

All our typing rules can be efficiently validated using a Gottesman-Knill, or check matrix, semantics (c.f. [1] or [11, §10.5]). We associate an n-qubit Pauli operator to a row vector over the binary field $\mathbb{F}_2$ via

$$(X^{x_1} Z^{z_1}) \otimes \cdots \otimes (X^{x_n} Z^{z_n}) \mapsto \left(\begin{array}{ccc|ccc} x_1 & \cdots & x_n & z_1 & \cdots & z_n \end{array}\right)$$

For instance, $\mathbf{X}$ and $\mathbf{Z}$ have the following check matrices:

$$\mathbf{X} \mapsto \left(\begin{array}{c|c} 1 & 0 \end{array}\right),\ \mathbf{Z} \mapsto \left(\begin{array}{c|c} 0 & 1 \end{array}\right)$$

All our stabilizer types are the joint eigenvalues of a (commutative) group of Pauli operators, typically called the stabilizer group of the state. We define a semantics for our types by taking the check matrix associated with its stabilizer group. For example, the entangled type $\mathbf{E}$ is the joint eigenvector of the group $\langle X \otimes X, Z \otimes Z\rangle$; we associate this type with a check matrix by mapping each generator to a row:

$$\begin{array}{rcl} X \otimes X & \mapsto & \left(\begin{array}{cc|cc} 1 & 1 & 0 & 0 \end{array}\right) \\ Z \otimes Z & \mapsto & \left(\begin{array}{cc|cc} 0 & 0 & 1 & 1 \end{array}\right) \end{array} = \left(\begin{array}{cc|cc} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{array}\right)$$

We could choose different generators for our group, the result of this is to apply row reduction operations to our check matrix. Hence such check matrices are uniquely defined by a reduced echelon form.

The operations X, Z and H take qubit types to qubit types. As X and Z preserve the qubit types while H inverts them, we can denote them in check matrix semantics as linear transformations:

$$X, Z \mapsto \left(\begin{array}{c|c} 1 & 0 \\ \hline 0 & 1 \end{array}\right), \; H \mapsto \left(\begin{array}{c|c} 0 & 1 \\ \hline 1 & 0 \end{array}\right)$$

Then, for example, the linear algebra equality

$$\left(\begin{array}{c|c} 1 & 0 \end{array}\right) \cdot \left(\begin{array}{c|c} 0 & 1 \\ \hline 1 & 0 \end{array}\right) = \left(\begin{array}{c|c} 0 & 1 \end{array}\right)$$

is the semantic representation of the Hoare triple $\{q : \mathbf{X}\}\ H(q)\ \{q : \mathbf{Z}\}$ or the typing statement $H : \mathbf{X} \to \mathbf{Z}$.

Note that there are many more binary matrices, even in reduced echelon form, than the types we have described above. Even for a single qubit, the eigenvectors of $Y = iXZ$ – with check matrix $\left(\begin{array}{c|c} 1 & 1 \end{array}\right)$ – define an additional type $\mathbf{Y}$ containing the terms

$$\tfrac{1}{\sqrt{2}}\left(|0\rangle + i\,|1\rangle\right) \text{ and } \tfrac{1}{\sqrt{2}}\left(|0\rangle - i\,|1\rangle\right)$$

While the unitary Y shares typing rules with X and Z for our existing types, adding the phase gate S would force us to expand the system, taking $\mathbf{X}$ to $\mathbf{Y}$ and $\mathbf{G}$ and $\mathbf{E}$ to new entangled states. These are easy to discover, however, by applying the S check matrix $\left(\begin{array}{c|c} 1 & 1 \\ \hline 0 & 1 \end{array}\right)$ to our existing types.

5 Applications and Future Work

This work can be extended to cover a wide range of applications, some of which we list below.

Quantum teleportation. We cannot directly verify the teleportation protocol using just the $\mathbf{Z}, \mathbf{X}$ and $\mathbf{E}$ types as the state to be teleported could be arbitrary. However, we can demonstrate *entanglement swapping*: Alice shares one ebit each with Bob and Carol using which an ebit shared between Bob and Carol can be produced. This requires us to add additional 3 and 4-qubit entangled types via the methods of §4.

Key distribution and error correction. Quantum key distribution (QKD) protocols [4, 7] where Alice and Bob share EPR pairs and measure in either the Z or X bases, are easy to typecheck using our system. Additionally, stabilizer codes [17, 5] used for error correction are a natural fit for this typesystem where information is encoded in the eigenvectors of Pauli operators and error correction is performed by a series of Clifford operations.

Ancilla correctness. Many quantum circuits introduce ancillary qubits that are used to perform some classical computation and are then discarded in a basis state. While the Quipper [8] and Q# [19] languages allow us to *assert* that ancilla are separable and can be safely discarded, and QWIRE allows us to manually verify this [13], our type system can automatically guarantee ancilla correctness by showing that the ancillae are in $\mathbf{Z}$ and (therefore) separable.

Resource tracking. Resource monotones can 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. By a suitable extension to our type system, 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. One way to include such monotones into the typesystem is by associating some numbers to the types. However, we expect that this will require significant modifications to the typesystem as discussed in this work, making this a long-term goal.

References

[1] Scott Aaronson and Daniel Gottesman. "Improved simulation of stabilizer circuits". In: *Phys. Rev. A* 70 (5 2004), p. 052328. DOI: 10.1103/PhysRevA.70.052328.

[2] Simon Anders and Hans J. Briegel. "Fast simulation of stabilizer circuits using a graph-state representation". In: *Phys. Rev. A* 73.2 (Feb. 2006), p. 022334. DOI: 10.1103/PhysRevA.73.022334.

[3] T. Baumgratz, M. Cramer, and M. B. Plenio. "Quantifying Coherence". In: *Phys. Rev. Lett.* 113 (14 2014), p. 140401. DOI: 10.1103/PhysRevLett.113.140401.

[4] Charles H. Bennett and Gilles Brassard. "Quantum cryptography: Public key distribution and coin tossing". In: *Theoretical Computer Science* 560 (2014). Theoretical Aspects of Quantum Cryptography – celebrating 30 years of BB84, pp. 7–11. DOI: https://doi.org/10.1016/j.tcs.2014.05.025.

[5] A. Robert Calderbank and Peter W. Shor. "Good quantum error-correcting codes exist". In: *Phys. Rev. A* 54.2 (1996), p. 1098. DOI: 10.1103/PhysRevA.54.1098.

[6] Eric Chitambar et al. "Everything You Always Wanted to Know About LOCC (But Were Afraid to Ask)". In: *Communications in Mathematical Physics* 328.1 (May 2014), pp. 303–326. DOI: 10.1007/s00220-014-1953-9.

[7] Artur K. Ekert. "Quantum cryptography based on Bell's theorem". In: *Phys. Rev. Lett.* 67 (6 Aug. 1991), pp. 661–663. DOI: 10.1103/PhysRevLett.67.661.

[8] Alexander S. Green et al. "Quipper: A Scalable Quantum Programming Language". In: *Proceedings of the 34th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '13)*. New York, NY, USA: ACM, 2013, pp. 333–342. DOI: 10.1145/2491956.2462177.

[9] Marc Hein, Jens Eisert, and Hans J. Briegel. "Multiparty entanglement in graph states". In: *Phys. Rev. A* 69 (6 June 2004), p. 062311. DOI: 10.1103/PhysRevA.69.062311.

[10] Ryszard Horodecki, Paweł Horodecki, Michał Horodecki, and Karol Horodecki. "Quantum entanglement". In: *Rev. Mod. Phys.* 81 (2 2009), pp. 865–942. DOI: 10.1103/RevModPhys.81.865.

[11] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010. DOI: 10.1017/CBO9780511976667.

[12] Jennifer Paykin, Robert Rand, and Steve Zdancewic. "QWIRE: A Core Language for Quantum Circuits". In: *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages (POPL '17)*. New York, NY, USA: ACM, 2017, pp. 846–858. DOI: 10.1145/3009837.3009894.

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

[14] Robert Rand, Jennifer Paykin, and Steve Zdancewic. "QWIRE Practice: Formal Verification of Quantum Circuits in Coq". In: *Proceedings 14th International Conference on Quantum Physics and Logic (QPL '17)*. 2017, pp. 119–132. DOI: 10.4204/EPTCS.266.8.

[15] A. Sabelfeld and A. C. Myers. "Language-based Information-flow Security". In: *IEEE J.Sel. A. Commun.* 21.1 (Sept. 2006), pp. 5–19. DOI: 10.1109/JSAC.2002.806121.

[16] Peter Selinger and Benoît Valiron. "A lambda calculus for quantum computation with classical control". In: *Mathematical Structures in Computer Science* 16.3 (2006), pp. 527–552. DOI: 10.1017/S0960129506005238.

[17] Andrew Steane. "Multiple-particle interference and quantum error correction". In: *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences* 452.1954 (1996), pp. 2551–2577. DOI: https://doi.org/10.1098/rspa.1996.0136.

[18] Alexander Streltsov, Gerardo Adesso, and Martin B. Plenio. "Colloquium: Quantum coherence as a resource". In: *Rev. Mod. Phys.* 89 (4 2017), p. 041003. DOI: 10.1103/RevModPhys.89.041003.

[19] Krysta Svore et al. “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)*. New York, NY, USA: ACM, 2018, 7:1–7:10. DOI: 10.1145/3183895.3183901.