A generalization of the stabilizer formalism for simulating arbitrary quantum circuits

Theodore J. Yoder
Massachusetts Institute of Technology
(Dated: December 12, 2012)

We present a new approach to simulate arbitrary quantum circuits on a classical computer. Our technique generalizes the stabilizer formalism, the underpinnings of the Gottesman-Knill theorem, to include arbitrary states and arbitrary quantum operations. The core of our approach is a novel state representation combining the density matrix and stabilizer representations. Obviously, not all simulations are efficient within our formalism, but we find special cases, wherein the input state and number of non-Clifford gates are restricted, which are. Importantly, the stabilizer circuits remain efficiently simulatable by our techniques, as a special case. The non-stabilizer quantum circuits for which we prove efficient simulations are just mildly non-stabilizer as judged by a certain heuristic, in terms of which our formalism's time complexities are naturally expressed. Interestingly, it seems that our approach is related to an interaction picture of quantum circuits, where stabilizer operations take the role of the unperturbed evolution.

I. INTRODUCTION

The divide between classical and quantum computers can be investigated in a few ways. One may find problems that the latter can do asymptotically faster than the former, Shor's factoring algorithm being the canonical example [1]. Or one might look to resources, such as entanglement [2, 3], discord [4], or negativity of the Wigner function [5, 6], to which the quantum computer has access, while the classical machine does not. In this study, we take a very direct approach: simulate quantum circuits with a classical computer and note the efficiency. It has long been known that a certain subset of quantum circuits, the stabilizer circuits, can be easily simulated by using a specific representation of a quantum state, the stabilizer formalism [7, 8]. How far does this set of "classical" circuits extend? What sort of quantum state representation might we need to simulate such circuits most efficiently?

As an attempt to answer these questions, we introduce a new representation of a quantum state, hereafter referred to as a generalized stabilizer. The generalized stabilizer representation is universal, in the sense that it has the property that (i) any quantum state, pure or mixed, may be represented and (ii) it can be simulated through arbitrary quantum circuits, including unitary gates, measurements, and quantum channels. Our representation also demonstrates that a class of circuits slightly larger than that of stabilizer circuits are efficiently classically simulatable. A central result in the theory of stabilizers is the Gottesman-Knill theorem, which states that a subset of quantum states, the stabilizer states, can be efficiently classically simulated (*i.e.* in time polynomial in the number of qubits n) through a subset of quantum circuits, the stabilizer circuits [8]. We can relax the assumptions of the Gottesman-Knill theorem in two ways, by allowing states more general than stabilizer states, and circuits more general than stabilizer circuits. Among our conclusions, we find an efficient classical simulation for stabilizer states through circuits containing $\mathcal{O}(\text{polylog}(n))$ non-stabilizer operations. Interestingly, we also find that the time to simulate an arbitrary n-qubit state τ through any stabilizer circuit of g Clifford gates and m Pauli measurements is upper bounded by $\mathcal{O}(gn + mn\Lambda(\tau) + mn^2)$. Here $\Lambda(\tau)$ is a certain property of the initial state, which is a heuristic measure of the distance of τ from the nearest stabilizer state. What is interesting is, first, the efficiency is linear in the number of measurements, as we show in section IV. Second, we have this property $\Lambda(\tau)$ whose small size is a sufficient condition for efficient simulation.

The definition of a classical simulation of a quantum circuit used throughout this paper is the strong sense. The strong simulation problem is, given a circuit and input state, determine (with some machine precision) the probability that a specified string of measurement outcomes will occur. For example, Alice may be teleporting a single qubit to Bob and we would like to know her chances of seeing the result 01 from her two measurements. This is to be distinguished from the concept of a weak simulation, which is simply required to report a string of measurement outcomes sampled from a distribution suitably close to the proper quantum distribution. Performing many weak simulations on the teleporting example, we would see a uniform distribution emerge over the set of measurement results $\{00, 01, 10, 11\}$. In each case, if we want an estimate of the entire probability distribution over m measurement outcomes an exponential number $\mathcal{O}(2^m)$ of simulations must be performed, but we judge efficiency based on the time taken for one simulation to run in terms of n, the number of qubits involved. It is known that a large gap exists between the weak and strong sampling problems. There exist circuits whose strong simulation is #P-complete, but whose weak simulation is in BPP [9].

The rest of this paper is organized as follows. In the remainder of the introduction, we describe our works' relation to previous ideas. In section II we briefly review the stabilizer formalism, citing results that will be needed in the later sections. Section III introduces the ideas of Aaronson and Gottesman [10] of a destabilizer and tableau. We also formally introduce the notion of a stabilizer basis. Section IV contains the definition of our generalized stabilizer

representation, as well as the presentation of the algorithms required for simulating circuits using it. In section V, we explain how to calculate the inner product between generalized stabilizers, which leads to an algorithm for determining the equality of two generalized stabilizer states. Finally, concluding remarks, including a compelling connection to the interaction picture of quantum mechanics, are made in section VI.

A. Related work

Since their introduction in 1997 by Gottesman, stabilizer groups have excited interest in quantum complexity and quantum coding theory [7]. For example, the insight leading to the Gottesman-Knill theorem, was to use a Heisenberg representation of a quantum state, so that rather than evolving an exponentially sized complex vector ($\sim 2^n$ numbers for an n-qubit state), the classical machine would only have to evolve n Pauli operators. However, not every state can be represented this way, and the set of Pauli operators does not map to itself under action of arbitrary quantum gates. Thus, using this formalism, one is restricted to a special subset of quantum circuits, the stabilizer circuits, which include Clifford gates and measurements [8].

This technique has been improved several times since its first introduction in the late 1990s. Aaronson and Gottesman improved the time complexity of measurement on an n-qubit stabilizer state, from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$, at the expense of a doubling in space complexity [10]. Anders and Briegel obtained a further speedup to $\mathcal{O}(n \log n)$ time and a reduction in space, assuming that the simulation begins with a *graph state*, a special type of stabilizer state [11]. The Gottesman-Knill theorem has even been broadened by Bartlett, *et. al.* to include continuous variable quantum computation [12]. Bermejo-Vega and Van den Nest have recently studied stabilizer formalisms in any finite Abelian group, generalizing from the traditional group of n-qubits $\mathbb{Z}_2^n$ [13]. Partly, the reason to study circuits that consist of many stabilizer gates and a few non-stabilizer gates or states that are slightly non-stabilizer is to extend the class of circuits to which the Gottesman-Knill theorem applies. Generalized stabilizers can provide some insights along these lines.

Similarly, among the motivations for studies of stabilizer circuits is gaining a deeper understanding of the relation between classical and quantum computation. For example, Aaronson and Gottesman also show that the problem of simulating stabilizer circuits is a complete problem for the classical complexity class $\oplus\mathrm{L} \subseteq \mathrm{P}$, implying that stabilizer circuits are unlikely to be universal even for classical computation [10]. This is somewhat surprising; after all, supplementing Clifford gates and measurements with almost any other quantum gate immediately yields universal quantum circuits, which are of course believed to be more powerful than classical circuits [3].

Finally, stabilizer states are interesting from a practical point of view. Many of the most appealing quantum error-correcting codes are naturally formalized using stabilizers. Stabilizer codes are the quantum equivalent of classical linear codes and stabilizer circuits act as their encoding and decoding routines. Famous codes from the stabilizer family are the "perfect" five-qubit code, Shor's nine-qubit code, and Calderbank-Shor-Steane (CSS) codes [7, 14, 15]. Since physical implementations of quantum computation are prone to error, simple, error-correcting schemes like those involving stabilizer codes are practically important [3, 7]. Our generalization of stabilizers fits nicely into this picture, since it allows simulation of *faults*, the arbitrary effect of a failed gate. Stabilizer codes project arbitrary faults onto a basis of Pauli operations (bit-flip, phase-flip, and bit-phase-flip errors), but there is no reason that any error correcting code must do this.

Enhancements of the stabilizer formalism have often been considered in the past. Our approach, however, differs in several ways. First, contrary to some propositions, we do not explicitly use a superposition of stabilizer states to represent an arbitrary state. Instead, we employ the tableau construction of Aaronson and Gottesman, which implicitly represents a set of orthogonal stabilizer states, a *stabilizer basis*, in which we can represent arbitrary quantum states. Importantly, updating the tableau, which takes only twice as long as updating a single stabilizer, effectively updates the entire stabilizer basis. We should point out that the concept of a stabilizer basis has been studied recently by Gay with application to improving the efficiency of model-checking [16]. Also, we allow simulation of arbitrary channels as opposed to just unitary gates and measurements. To do this, we decompose the Kraus operators of the channel into the Pauli operators from the state's tableau.

II. STABILIZERS

The stabilizer formalism is well documented in the literature (see, for some introductory accounts, [3, 7, 17]). Here we review the results relevant to this paper. Let the Pauli matrices in the computational basis be denoted

$$\sigma_0 = \mathcal{I} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},\ \sigma_1 = X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},\ \sigma_2 = Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix},\ \sigma_3 = Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{1}$$

Then we define the *Pauli group* on n-qubits as the set of all tensor products of n Pauli operators. Namely,

$$G_n = \{i^k \sigma_{i_1} \otimes \sigma_{i_2} \otimes \cdots \otimes \sigma_{i_n} \,\big|\, k, i_h \in \{0,1,2,3\}, \forall h\}. \tag{2}$$

Note that each member of this group can have one of four phases, $+1, -1, +i, -i$. The group G_n is often a convenient (and complete) basis for expressing unitary operations. For example, the three generators of the Clifford group of gates can be written

$$H = \tfrac{1}{\sqrt{2}}(X+Z)\,,\;\; S = \tfrac{1}{\sqrt{2}}(\mathcal{I} - iZ)\,,\;\; \text{CNOT} = \tfrac{1}{2}\left[\mathcal{I}\mathcal{I} + Z\mathcal{I} + \mathcal{I}X - ZX\right]. \tag{3}$$

Here H is the Hadamard gate, S is the phase gate, and CNOT is the two qubit gate controlled on the first (leftmost) qubit. Note that instead of explicitly writing the tensor product, we will often abbreviate with simple strings of Pauli operators. For example, $\mathcal{I} \otimes X = \mathcal{I}X$. The Clifford group maps G_n to itself under conjugation. Another important gate is the square root of S, also known as the $\pi/8$ gate,

$$T = \cos\frac{\pi}{8}\mathcal{I} + i \sin\frac{\pi}{8} Z. \tag{4}$$

The set of gates $\{H, T, \text{CNOT}\}$ is universal for quantum computation, while Clifford circuits, those just containing gates from (3), are probably not even universal for classical computation [10].

Stabilizer groups are subgroups of the complete Pauli group with some special properties [3, 7].

Definition 1. A *stabilizer* group S on n-qubits is a subgroup of G_n satisfying the following

- S is abelian. If $\mathcal{s}_1, \mathcal{s}_2 \in S$, then $[\mathcal{s}_1, \mathcal{s}_2] = 0$
- S does not contain the negative identity. That is, $-\mathcal{I}^{\otimes n} = -\mathcal{I} \notin S$

We also define the set of states stabilized by S as $\mathcal{H}_S = \{|\psi\rangle \in \mathbb{C}^{2^n} \,\big|\, \mathcal{s}|\psi\rangle = |\psi\rangle, \forall \mathcal{s} \in S\}$.

The fact that the elements of S commute and do not include the negative identity guarantees some nonzero state is in $\mathcal{H}_S$. We often write a stabilizer S in terms of its generators $s_1, s_2, \ldots s_r$,

$$S = \langle s_1, s_2, \ldots, s_r \rangle, \tag{5}$$

since there only r generators while S has 2^r elements in total. Any $\mathcal{s} \in S$ contains at most one factor of each generator, since Pauli operators square to the identity. Since all the generators must commute, $r \leq n$. If $r = n$, then $\mathcal{H}_S$ is one dimensional. Our convention is to call stabilizers that have as many generators as qubits *full stabilizers*.

Theorem 2. *A full stabilizer (i.e. an n-qubit stabilizer S with n generators $s_1, s_2, \ldots, s_n$) stabilizes a unique pure state, denoted $|\psi_S\rangle$ or $\rho_S = |\psi_S\rangle\langle\psi_S|$. That is, $\mathcal{H}_S = \{|\psi_S\rangle\}$. [3, 7]*

A corollary is that the density matrix $\rho_S = |\psi_S\rangle\langle\psi_S|$ is [10]

$$\rho_S = \frac{1}{2^n} \prod_{i=1}^{n} (\mathcal{I} + s_i)\,. \tag{6}$$

The simplest form of a stabilizer is exemplified by $\langle Z_2, Z_1, Z_0\rangle = \langle Z\mathcal{I}\mathcal{I}, \mathcal{I}Z\mathcal{I}, \mathcal{I}\mathcal{I}Z\rangle$ which stabilizes the three qubit state $|000\rangle$. A bit more complicated example is $\langle -Z\mathcal{I}\mathcal{I}, \mathcal{I}XX, \mathcal{I}ZZ\rangle$, which stabilizes $|1\rangle(|00\rangle + |11\rangle)$. Even in these more complicated scenarios it is often useful to think of the stabilizer generators as surrogate Pauli Z operators, so that every stabilizer becomes analogous to the simplest case. That is, for three qubits, let $S = \langle \tilde{Z}_2, \tilde{Z}_1, \tilde{Z}_0\rangle$; the $\tilde{Z}_i$ act on $|\psi_S\rangle$ just as Z_i does on $|000\rangle$. In this way, the $\tilde{Z}_i$ are the canonical Z operators on a set of virtual qubits. We will refer to those virtual qubits associated with the stabilizer as *syndrome qubits*, in reference to stabilizer coding theory.

III. DESTABILIZERS, TABLEAUS, AND STABILIZER BASES

In addition to stabilizers, we also make use of destabilizers, introduced by Aaronson and Gottesman [10]. While the stabilizer generators can be thought of as virtual Z operators, the destabilizers can be thought of as their conjugate, virtual X operators.

Definition 3. A destabilizer group D associated with a stabilizer $S = \langle s_1, s_2, \ldots, s_r\rangle$ is a stabilizer subgroup of G_n that satisfies

- D has as many generators as S. So, $D = \langle d_1, d_2, \ldots d_r \rangle$
- Each d_i anti-commutes with s_i and commutes with each s_j where $j \neq i$

The commutation algebra relating the generators of D and those of S motivate the association of generators d_i with $\tilde{X}$, the canonical X operations on the syndrome qubits. Notice that a consequence of the generator algebra is the following form for the stabilizer state ρ_S conjugated by destabilizer generators:

$$d_k \rho_S d_k = \frac{1}{2^n} (\mathcal{I} - s_k) \prod_{i \neq k} (\mathcal{I} + s_i) . \tag{7}$$

The combination of a full stabilizer and a full stabilizer is surprisingly useful. We give the pair its own name, citing Aaronson and Gottesman [10],

Definition 4. A *full tableau* is the pair $\mathcal{T} = (S, D)$, where S is a full stabilizer and D is a corresponding full destabilizer.

One useful property of a full tableau is that, using it, we can efficiently decompose any Pauli operator $P \in G_n$ in terms of stabilizer and destabilizer generators. This decomposition lemma will be extremely useful in algorithms for working with stabilizers and generalized stabilizers.

Lemma 5. Given a full tableau $\mathcal{T} = (S, D)$ and $P \in G_n$, we can find $\alpha \in \{\pm 1, \pm i\}$, $\mathfrak{d} \in D$, and $\mathfrak{s} \in S$ such that $P = \alpha \mathfrak{d} \mathfrak{s}$ in $\mathcal{O}(n^2)$ time.

Proof. A full stabilizer and full destabilizer can generate the entire Pauli group on n-qubits (up to phases). After all the generators of S are a full set of n $\tilde{Z}$ operators, while D is a full set of n $\tilde{X}$ operators. Therefore, such a decomposition exists. To find it we simply check the commutation algebra of P with each generator. In a slight abuse of notation, we will write $k \in \mathfrak{d}$ if d_k appears in the factorization of $\mathfrak{d}$ into generators. By the commutation relations between stabilizer and destabilizer, $\{P, s_j\} = 0$ if and only if $j \in \mathfrak{d}$. Likewise, $\{P, d_j\} = 0$ if and only if $j \in \mathfrak{s}$. There are $2n$ anti-commutations to check and each takes $\mathcal{O}(n)$ time. After finding $\mathfrak{d}$ and $\mathfrak{s}$, we will have to multiply them and compare the phase of their product to that of P. Doing so gives α in $\mathcal{O}(n^2)$ time. □

If a tableau is not full, but instead contains a stabilizer group and destabilizer group of r generators each, then the tableau also contains two normalizer groups N_x and N_z which each have $(n - r)$ generators and anti-commute pairwise with one another, just as the stabilizer and destabilizer do. The entire normalizer is $N = N_x \cup N_z$, the group which commutes with both the stabilizer and destabilizer. The normalizer is very important in the theory of stabilizer codes, where it represents logical X and Z operations on the encoded qubits.

For our purposes, we focus on full tableaus. A full tableau defines what we call a *stabilizer basis*.

Definition 6. The *stabilizer basis* corresponding to the full tableau $\mathcal{T}$ is $\mathcal{B}(\mathcal{T}) = \mathcal{B}(S, D) = \{\mathfrak{d}|\psi_S\rangle \big| \ \mathfrak{d} \in D\}$.

This idea, although without use of the destabilizer or tableau, is also explored in a note by Gay [16]. The stabilizer basis can be thought of as the computational basis for the syndrome qubits, denoted with twiddled 0s and 1s, like $\{|\tilde{0}\tilde{0}\rangle, |\tilde{0}\tilde{1}\rangle, |\tilde{1}\tilde{0}\rangle, |\tilde{1}\tilde{1}\rangle\}$ for two qubits. A stabilizer basis is orthonormal and complete for the n-qubit Hilbert space.

Lemma 7. The stabilizer basis $\mathcal{B}(S, D)$ is orthonormal.

Proof. Normality follows immediately from the unitarity of any $\mathfrak{d} \in D$. For orthogonality, let $\mathfrak{d}_1 \in D$ and $\mathfrak{d}_2 \in D$ be distinct. Notice that any $\mathfrak{d} \in D$ is a product of D's generators d_i. Since $\mathfrak{d}_1 \neq \mathfrak{d}_2$, (without loss of generality) there is some $h \in \mathfrak{d}_1$, but also $h \notin \mathfrak{d}_2$. So, the generator s_h of S satisfies $\{s_h, \mathfrak{d}_1\} = 0$ while $[s_h, \mathfrak{d}_2] = 0$. Thus,

$$\begin{aligned} \langle \psi_S | \mathfrak{d}_1^\dagger \mathfrak{d}_2 | \psi_S \rangle &= \langle \psi_S | \mathfrak{d}_1 \mathfrak{d}_2 | \psi_S \rangle \\ &= \langle \psi_S | s_h \mathfrak{d}_1 \mathfrak{d}_2 | \psi_S \rangle \\ &= -\langle \psi_S | \mathfrak{d}_1 \mathfrak{d}_2 s_h | \psi_S \rangle, \end{aligned} \tag{8}$$

which implies orthogonality, $\langle \psi_S | \mathfrak{d}_1^\dagger \mathfrak{d}_2 | \psi_S \rangle = 0$, as desired. □

Although a stabilizer does not have a unique destabilizer, all stabilizer bases $\mathcal{B}(S, D)$ for a stabilizer S are the same up to phases on the basis states.

Lemma 8. Given stabilizer S, if D and D' are both destabilizers for S, then $\mathcal{B}(S, D)$ and $\mathcal{B}(S, D')$ are the same set of basis states modulo phase differences.

Proof. Since the generators of D and D' obey the same commutation properties with generators of S, if $\mathfrak{d}$ and $\mathfrak{d}'$ represent corresponding members of destabilizer groups D and D', then

$$\mathfrak{d}\rho_S\mathfrak{d} = \frac{1}{2^n}\prod_{i\in\mathfrak{d}}(\mathcal{I}-s_i)\prod_{i\notin\mathfrak{d}}(\mathcal{I}+s_i) = \frac{1}{2^n}\prod_{i\in\mathfrak{d}'}(\mathcal{I}-s_i)\prod_{i\notin\mathfrak{d}'}(\mathcal{I}+s_i) = \mathfrak{d}'\rho_S\mathfrak{d}'. \tag{9}$$

So $\mathfrak{d}|\psi_S\rangle\langle\psi_S|\mathfrak{d} = \mathfrak{d}'|\psi_S\rangle\langle\psi_S|\mathfrak{d}'$, which implies $\mathfrak{d}|\psi_S\rangle = e^{i\phi}\mathfrak{d}'|\psi_S\rangle$ for ϕ potentially dependent on the generator makeup of $\mathfrak{d}$. Note that because we are working with Pauli group operators, $e^{i\phi}$ actually can only be ± 1 or $\pm i$. □

IV. GENERALIZED STABILIZERS

A. Definition and Evolution

We now define the central object of this study, our new state representation. Afterward, we also prove how to evolve this representation through quantum circuits.

A *generalized stabilizer* uses a stabilizer basis to express an arbitrary state.

Definition 9. The generalized stabilizer representation of an arbitrary state τ is a two-tuple $(\chi, \mathcal{B}(S,D))$. Here χ is a density matrix and $\mathcal{B}(S,D) = \mathcal{B}(\mathcal{T})$ is the basis in which χ is expressed. Then τ, in the computational basis, is recovered from the representation by

$$\tau = \sum_{i=1}^{2^n}\sum_{j=1}^{2^n}\chi_{ij}\mathfrak{d}_i|\psi_S\rangle\langle\psi_S|\mathfrak{d}_j = \sum_{ij}\chi_{ij}\mathfrak{d}_i\rho_S\mathfrak{d}_j, \tag{10}$$

where we have arbitrarily indexed the elements $\mathfrak{d}\in D$. If D has r generators, we will often treat the index of $\mathfrak{d}_i$ as an r-bit binary vector such that the factorization of $\mathfrak{d}_i$ into the generators d_i of D is $\mathfrak{d}_i = \prod_{x=1}^{r}(d_x)^{i_x}$. Also, we will sometimes write $\tau = (\chi, \mathcal{B}(S,D))$.

Note that a generalized stabilizer can indeed represent a stabilizer as a special case, if χ is all zeros except for a single 1 along on the diagonal and the stabilizer basis is appropriately chosen. So anything a stabilizer representation can do is also possible with a generalized stabilizer. This includes the efficient simulation of stabilizer circuits implied by the Gottesman-Knill theorem for instance [8].

Theorem 10. (Gottesman-Knill 1997) *Any quantum computation, called a stabilizer circuit, that consists only of state preparation in the computational basis, Clifford gates, and measurement of Pauli operators (with possible, subsequent classical control) can be efficiently simulated on a classical computer. Specifically, if the circuit acts on n-qubits, then each measurement takes $\mathcal{O}(n^2)$ and each Clifford gate takes $\mathcal{O}(n)$ classical time. (Technically, these time complexities are not from the original theorem; we have incorporated the results of [10].)*

On the simplest level, the generalized stabilizer formalism is just a basis change; however, the fact that the new basis is a stabilizer basis proves useful, because we can take advantage of the structure of stabilizer groups in quantum circuit simulations. Generalized stabilizers provide an extension of Gottesman-Knill simulations to include non-Clifford gates and even general quantum channels. Though these are not generally efficiently simulated, the cases in which they are can be classified within this framework. In addition, it never becomes more difficult to update the generalized stabilizer representation through Clifford gates, no matter the quantum state being acted upon by the Clifford gate. We make this all explicit with the following definitions and theorem:

Definition 11. Let $\Lambda(A)$ be the number of nonzero elements of a matrix A. This is a measure of (inverse) sparsity. Define the Λ operator on a generalized stabilizer state $\tau = (\chi, \mathcal{B}(S,D))$ to be a minimum over all stabilizer bases

$$\Lambda(\tau) = \min_{\mathcal{B}(S,D)}\Lambda(\chi) = \min_{S}\Lambda(\chi). \tag{11}$$

It is equivalent to take the minimum over all stabilizer groups by lemma 8. In representing state τ, once the stabilizer basis $\mathcal{B}(S,D)$ is chosen, χ is fixed by the basis' orthogonality. Note the bounds, $\text{rank}(\tau) \le \Lambda(\tau) \le 4^n$.

Definition 12. Define a *Pauli channel* as a quantum operation on an arbitrary n-qubit state τ as one of the form

$$\mathcal{E}(\tau) = \sum_{ij}\phi_{ij}B_i\tau B_j^\dagger, \tag{12}$$

for complex numbers ϕ_{ij} and Pauli operators $B_i \in G_n$. We also define $\Lambda(\mathcal{E}) \equiv \Lambda(\phi)$ as a measure of the channel complexity. Note that any quantum operation can be expressed uniquely (up to phases on the basis operators B_i) as a Pauli channel.

Theorem 13. *Suppose we have a generalized stabilizer on n-qubits $\tau = (\chi, \mathcal{B}(S, D))$. We have the following deterministic update efficiencies for quantum operations,*

- *Clifford gates: $\mathcal{O}(n)$ time*
- *Pauli measurements: $\mathcal{O}\left(\Lambda(\chi)\, n + n^2\right)$ time*
- *Pauli channels $\mathcal{E}$: $\mathcal{O}\left(\Lambda(\chi)\,\Lambda(\mathcal{E})\, n + \sqrt{\Lambda(\mathcal{E})} n^2\right)$ time*

Note that $\Lambda(\tau) = 1$ if and only if we have a stabilizer state. In that case, the Aaronson-Gottesman [10] efficiency for measurement is recovered.

Proof. We prove each part in turn.

- Say we have Clifford circuit C. Then,

$$\begin{aligned} C\tau C^\dagger &= C\left(\sum_{ij} \chi_{ij} \mathbb{d}_i \rho_S \mathbb{d}_j\right) C^\dagger \\ &= \sum_{ij} \chi_{ij} C\mathbb{d}_i C^\dagger \left(C\rho_S C^\dagger\right) C\mathbb{d}_j C^\dagger \\ &= \sum_{ij} \chi_{ij} \mathbb{d}'_i \rho'_S \mathbb{d}'_j. \end{aligned} \tag{13}$$

  We see that we just need to update the stabilizer S and destabilizer D by conjugation by C. By the Gottesman-Knill theorem, the update takes $\mathcal{O}(n)$ time per Clifford gate in C. This update is presented in pseudocode as algorithm 1.

Algorithm 1 Updates the generalized stabilizer $\tau = (\chi, \mathcal{B}(S, D))$ by a Clifford gate G.
Takes τ and G and returns the updated stabilizer and destabilizer.

Conjugate each generator of $S = \langle s_1, s_2, \ldots, s_n\rangle$ by G.
Conjugate each generator of $D = \langle d_1, d_2, \ldots, d_n\rangle$ by G.
Return the resulting stabilizer $S' = \langle Gs_1G^\dagger, Gs_2G^\dagger, \ldots, Gs_nG^\dagger\rangle$ and destabilizer $D' = \langle Gd_1G^\dagger, Gd_2G^\dagger, \ldots, Gd_nG^\dagger\rangle$.

- Say we want to measure the (hermitian) Pauli operator $M \in G_n$. First, we decompose M in terms of the stabilizer and destabilizer,

$$M = \alpha \mathbb{d}_b \mathbb{s}_c, \tag{14}$$

  where $\alpha = \pm 1, \pm i$. By checking how M commutes with the stabilizer and destabilizer generators, we can find this decomposition in $\mathcal{O}(n^2)$ time (lemma 5). There are two major cases to consider, $b = (0, 0, \ldots, 0)$ or b is nonzero.

  In the former case, either M or $-M$ is a stabilizer group element. In what follows, all vector additions and dot products are conducted modulo two. We have

$$\begin{aligned} \tau' &= \frac{1}{4}(\mathcal{I} + M)\,\tau\,(\mathcal{I} + M) \\ &= \frac{1}{4}\sum_{ij} \chi_{ij} (\mathcal{I} + M)\, \mathbb{d}_i \rho_S \mathbb{d}_j\, (\mathcal{I} + M) \\ &= \frac{1}{4}\sum_{ij} \chi_{ij} \mathbb{d}_i \left[\left(\mathcal{I} + (-1)^{i\cdot c} M\right) \rho_S \left(\mathcal{I} + (-1)^{j\cdot c} M\right)\right] \mathbb{d}_j. \end{aligned} \tag{15}$$

  Using Eq. (6) we can evaluate the matrix in square brackets,

$$\left(\mathcal{I} + (-1)^{i\cdot c}\alpha \mathbb{s}_c\right) \rho_S \left(\mathcal{I} + (-1)^{j\cdot c}\alpha \mathbb{s}_c\right) = \begin{cases} \rho_S & , \alpha(-1)^{i\cdot c} = 1 \text{ and } \alpha(-1)^{j\cdot c} = 1 \\ 0 & , \text{otherwise} \end{cases}. \tag{16}$$

Therefore, we see that τ' can be written as a generalized stabilizer with a reduced χ-matrix, but in the same stabilizer basis,

$$\tau' = \sum_{ij} \chi'_{ij} \mathfrak{d}_i \rho_S \mathfrak{d}_j, \tag{17}$$

where $\Lambda(\chi') \leq \Lambda(\chi)$. We calculated only $\frac{1}{4}(\mathcal{I}+M)\tau(\mathcal{I}+M)$, corresponding to getting measurement result 0. However, $\frac{1}{4}(\mathcal{I}-M)\tau(\mathcal{I}-M)$ is gotten from this by changing the sign of α. The trace $\mathrm{Tr}[\tau'] = \mathrm{Tr}[\chi']$ is the probability of measuring 0 and is also the normalization required for τ'.

In the case where b is nonzero, we have more work to do. Let k be the binary vector of weight 1, whose single nonzero entry lies in the position of b's first nonzero entry. For example, if $b = (0,1,0,0,1)$, then $k = (0,1,0,0,0)$. Then we see that

$$\begin{aligned}
\tau' &= \frac{1}{4} \sum_{ij} (\mathcal{I}+M) \mathfrak{d}_i \rho_S \mathfrak{d}_j (\mathcal{I}+M) \\
&= \frac{1}{4} \sum_{ij} (\mathcal{I}+M) \mathfrak{d}_i \left(s_k^{i\cdot c} \rho_S s_k^{j\cdot c} \right) \mathfrak{d}_j (\mathcal{I}+M) \\
&= \frac{1}{4} \sum_{ij} \mathfrak{d}_i s_k^{i\cdot c} (\mathcal{I}+M) \rho_S (\mathcal{I}+M) s_k^{j\cdot c} \mathfrak{d}_j.
\end{aligned} \tag{18}$$

We have forced the projector to commute with the destabilizer elements using ρ_S to spawn a conditional factor of the stabilizer generator $s_k \equiv \mathfrak{s}_k$. Now we can perform a traditional measurement update to the stabilizer and destabilizer (in $\mathcal{O}(n^2)$ time by theorem 10), getting the new stabilizer state $\frac{1}{2}\rho''_S = \frac{1}{4}(\mathcal{I}+M)\rho_S(\mathcal{I}+M)$. This is done through four steps, [10]

1. First, replace all stabilizer generators s_j such that $b_j = 1$ with $s_j s_k$.
2. Replace all destabilizer generators d_j such that $c_j = 1$ and $j \neq k$ with $d_j s_k$. Replace d_k with 1.
3. Replace d_k with s_k.
4. Replace s_k with M.

After the first two steps we have stabilizer elements $\mathfrak{s}'$ and destabilizer elements $\mathfrak{d}'$ related to the originals by

$$\mathfrak{s}'_i = \mathfrak{s}_i s_k^{i\cdot b}, \tag{19}$$

$$\mathfrak{d}'_i = d_k^{i\cdot k} \mathfrak{d}_i s_k^{i\cdot c + (i\cdot k)(c\cdot k)}. \tag{20}$$

We note two immediate consequences,

$$M = \alpha \mathfrak{d}_b \mathfrak{s}_c = \alpha d_k \mathfrak{d}'_b \mathfrak{s}'_c s_k^{c\cdot k}. \tag{21}$$

and, from Eq. (18),

$$\tau' = \frac{1}{2} \sum_{ij} \chi_{ij} \left(d_k^{i\cdot k} \mathfrak{d}'_i s_k^{(i\cdot k)(c\cdot k)} \right) \rho''_S \left(s_k^{(j\cdot k)(c\cdot k)} \mathfrak{d}'_j d_k^{j\cdot k} \right). \tag{22}$$

We are not done, however, since this is not in generalized stabilizer form, which requires $\mathfrak{d}''$ operators on either side of ρ''_S. After the final two steps of the traditional stabilizer update, define $\mathfrak{s}''$ and $\mathfrak{d}''$ as our group elements. Then,

$$\mathfrak{s}''_i = M^{i\cdot k} \mathfrak{s}'_i, \tag{23}$$

$$\mathfrak{d}''_i = s_k^{i\cdot k} \mathfrak{d}'_i = \mathfrak{d}'_i s_k^{i\cdot k}. \tag{24}$$

Since $\mathfrak{s}''_i \rho''_S = \rho''_S$ and $M\rho''_S = \rho''_S$, we see that $\mathfrak{s}'_i \rho''_S = \rho''_S$ for all vectors i. Consider a term from the sum in Eq. (22). If $i \cdot k = 0$, then $\left(d_k^{i\cdot k} \mathfrak{d}'_i s_k^{(i\cdot k)(c\cdot k)} \right) = \mathfrak{d}''_i$ and we are done. However, now say $i \cdot k = 1$. Then we will spawn a factor of M from ρ''_S,

$$\tau' = \frac{1}{2} \sum_{ij} \chi_{ij} \left(d_k^{i\cdot k} \mathfrak{d}'_i s_k^{(i\cdot k)(c\cdot k)} \right) M^{i\cdot k} \rho''_S \left(M^\dagger \right)^{j\cdot k} \left(s_k^{(j\cdot k)(c\cdot k)} \mathfrak{d}'_j d_k^{j\cdot k} \right) \tag{25}$$

and calculate

$$\begin{aligned}\left(d_k \mathfrak{d}_i' s_k^{c\cdot k}\right) M &= \left(d_k \mathfrak{d}_i' s_k^{c\cdot k}\right)\left(\alpha d_k \mathfrak{d}_b' \mathfrak{s}_c' s_k^{c\cdot k}\right) \\ &= \alpha(-1)^{c\cdot k} d_k \mathfrak{d}_i d_k \mathfrak{d}_b' \mathfrak{s}_c' \\ &= \alpha(-1)^{i\cdot c} \mathfrak{d}_i' \mathfrak{d}_b' \mathfrak{s}_c' \\ &= \alpha(-1)^{i\cdot c} \mathfrak{d}_{i+b}'' \mathfrak{s}_c'.\end{aligned}$$

At the third equality, we made use of the fact that, if $i \cdot k = 1$, then $d_k \mathfrak{d}_i' = (-1)^{i\cdot c + c\cdot k} \mathfrak{d}_i' d_k$. Now we see that,

$$\tau' = \sum_{ij}\left[\frac{1}{2}\left(\alpha(-1)^{i\cdot c}\right)^{i\cdot k}\left(\alpha^*(-1)^{j\cdot c}\right)^{j\cdot k}\chi_{ij}\right]\left(\mathfrak{d}_{i+(i\cdot k)b}''\right)\rho_S''\left(\mathfrak{d}_{j+(j\cdot k)b}''\right). \tag{26}$$

Evidently, with an appropriate definition of the updated χ-matrix χ'_{ij}, this is in generalized stabilizer form. Notice that, in this case, the density of χ never increases and may decrease because χ'_{ij} is always zero when $(i \cdot k) = 1$ or $(j \cdot k) = 1$.

This constructive proof is summarized in Algorithm 2. Both the decomposition of M and the traditional stabilizer update take $\mathcal{O}(n^2)$ time, while the construction of the updated χ-matrix takes $\mathcal{O}(\Lambda(\chi)\, n) = \Omega(\Lambda(\tau)\, n)$ time.

Algorithm 2 Updates the generalized stabilizer $\tau = (\chi, \mathcal{B}(S, D))$ by a measurement of $M = \alpha \mathfrak{d}_b \mathfrak{s}_c$. Takes τ and α, b, c and returns χ' and so that $\tau' = \frac{1}{4}(\mathcal{I} + M)\tau(\mathcal{I} + M) = (\chi', \mathcal{B}(S', D'))$, where (S', D') is determined from (S, D) by the traditional stabilizer update described in the text.

```
Initialize sparse matrix χ'
if b = 0 then
    for all χ_ij ≠ 0 do
        if α(−1)^{i·c} = 1 and α(−1)^{j·c} = 1 then
            χ'_ij ← χ'_ij + χ_ij
else
    k ← vector of zeros with a single 1 in the location of b's first 1
    for all χ_ij ≠ 0 do
        x ← i
        y ← j
        q ← 1
        if i · k = 1 then
            q ← α(−1)^{i·c} q
            x ← i + b
        if j · k = 1 then
            q ← α*(−1)^{j·c} q
            y ← j + b
        χ'_xy ← χ'_xy + (1/2) q χ_ij
Return χ', S', D'
```

- We consider a channel of the form

$$\mathcal{E}(\tau) = \sum_{ij} \phi_{ij} B_i \tau_{ij} B_j^\dagger, \tag{27}$$

where the basis operators B_i are in G_n. Any channel, and any unitary gate, can be expressed in this fashion, though $\Lambda(\mathcal{E})$ may be exponential in general. We first decompose every B_i,

$$B_i = \alpha_i \mathfrak{d}_{b_i} \mathfrak{s}_{c_i}. \tag{28}$$

This takes $\mathcal{O}\left(n^2 s\right)$ time if there are $s \sim \sqrt{\Lambda(\mathcal{E})}$ basis operators. For each term in the channel (27), we can calculate

$$\begin{aligned}\phi_{hk} B_h \tau B_k^\dagger &= \sum_{ij} \phi_{hk}\chi_{ij} B_h \mathfrak{d}_i \rho_S \mathfrak{d}_j B_k^\dagger \\ &= \sum_{ij}\left[\phi_{hk}\alpha_h\alpha_k^*(-1)^{i\cdot c_h + j\cdot c_k}\chi_{ij}\right]\mathfrak{d}_{i+b_h}\rho_S\mathfrak{d}_{j+b_k}.\end{aligned} \tag{29}$$

With appropriate choice of χ' this is in generalized stabilizer form $(\chi', \mathcal{B}(S, D))$. For all the terms in Eq. (27), we add the χ' matrices together. This procedure is summarized in Algorithm 3, from which one can easily evaluate the time complexity.

Algorithm 3 Updates the generalized stabilizer $\tau = (\chi, \mathcal{B}(S, D))$ by a channel $\mathcal{E}(\tau) = \sum_{ij} \phi_{ij} B_i \tau_{ij} B_j^\dagger$. Takes τ and the set of decompositions $\{\alpha_h, b_h, c_h\}_h$ for each B_h and returns χ' so that $\tau' = \mathcal{E}(\tau) = (\chi', \mathcal{B}(S, D))$.

Initialize sparse matrix χ'
for all $\phi_{kl} \neq 0$ **do**
  for all $\chi_{ij} \neq 0$ **do**
    $x \leftarrow i + b_k$
    $y \leftarrow j + b_l$
    $\chi'_{xy} \leftarrow \chi'_{xy} + \left[\alpha_h \alpha_l^* (-1)^{i \cdot c_k + j \cdot c_l}\right] \phi_{kl} \chi_{ij}$
Return χ'

□

Notice that the measurement time complexity ostensibly follows from the channel time complexity; we could treat the measurement as a channel with $\Lambda(\phi) = \mathcal{O}(1)$. However, there is an important difference between the algorithms presented in each case. In the case of evolution by a channel, the sparsity of χ can generally increase. However, for the measurement routine given, χ' after measurement is *never* less sparse than χ before the measurement, $\Lambda(\chi') \leq \Lambda(\chi)$. Plus, if we were to specialize to stabilizer states, Algorithm 2 performs exactly the traditional stabilizer update, while treating the measurement as a channel and using Algorithm 3 would not. We record the increase in the complexity of χ in the following corollary.

Corollary 14. Let $\tau = (\chi, \mathcal{B}(\mathcal{T}))$ be a generalized stabilizer before action of a quantum operation and $\tau' = (\chi', \mathcal{B}(\mathcal{T}'))$ be the generalized stabilizer after the operation, with the update performed by the algorithms 1, 2, or 3. Then the following relations between Λ-complexities are true,

- Clifford gates: $\chi' = \chi$,
- Pauli measurements: $\Lambda(\chi') \leq \Lambda(\chi)$,
- Pauli channels $\mathcal{E}$: $\Lambda(\chi) \leq \Lambda(\chi') \leq \Lambda(\mathcal{E})\Lambda(\chi)$.

B. Comments and Complexity

It is important to note what freedoms are available in representing a state τ as a generalized stabilizer $(\chi, \mathcal{B}(S, D))$. Obviously, there are many stabilizer bases we could choose. Since there are roughly $2^{\frac{1}{2}n^2}$ pure stabilizer states on n qubits [10], there are $2^{\frac{1}{2}n^2 - n}$ stabilizer bases for the n-qubit Hilbert space. Of course, the best choice of stabilizer basis would be that which minimizes the density of the χ-matrix, so that $\Lambda(\chi) = \Lambda(\tau)$ from definition 11. We do not, however, know of a fast way of finding such an optimal basis.

We would now like to draw some conclusions about the complexity of simulating certain quantum circuits. Using theorem 13, one can consider several special cases of quantum circuits, by restricting initial states or the allowed gates, and see how efficiently the simulation can proceed using a generalized stabilizer.

- One case is simulation of arbitrary states through stabilizer circuits. From theorem 13 this takes time $\mathcal{O}(gn + m\left(\Lambda(\tau)n + n^2\right))$, if there are g Clifford gates, m measurements, and the initial state is $\tau = (\chi, \mathcal{B}(\mathcal{T}))$. We assume that the optimal stabilizer basis with which to represent τ can be found offline, before the simulation starts. Certainly if $\Lambda(\tau) = \mathcal{O}\left(\text{poly}(n)\right)$, this simulation would be polynomial time. However, even if we restrict our initial state to be a tensor product of k blocks of b qubits each (so $\Lambda(\tau) \leq k2^{2b}$), we improve upon the similar result of Aaronson and Gottesman, whose algorithm is exponential in the number of measurements [10]. This improvement can be attributed to our measurement algorithm, Algorithm 2, which never decreases the sparsity of the χ matrix.

  An algorithm that uses just a density matrix representation could also use sparsity to improve its runtime for simulation of such non-stabilizer states through stabilizer circuits. However, simulating g Clifford gates will generally result in an exponential (*i.e.* $2^{\mathcal{O}(g)}$) decrease in the sparsity of the density matrix representation, such that, when it comes time to measure the state at the conclusion of the simulation, the calculation will

be intractable. Our method avoids this by updating the stabilizer basis instead of the density matrix directly. The price we pay is having to interpret the concluding measurements in terms of our modified basis, but this is polynomial in n by the decomposition lemma.

- Another special case is simulation of stabilizer states through non-stabilizer circuits. To simplify the discussion, we will first add just one non-Clifford gate to the set of allowed gates, say the $\pi/8$ gate T, giving us quantum universality. We can simulate the action of T on a generalized stabilizer by Algorithm 3, since

$$T\tau T^\dagger = \left(\cos\frac{\pi}{8}\mathcal{I} + i\sin\frac{\pi}{8}Z\right)\tau\left(\cos\frac{\pi}{8}\mathcal{I} - i\sin\frac{\pi}{8}Z\right). \tag{30}$$

  can be expanded into the form of a Pauli channel. Every action of a T-channel will decrease the sparsity of χ by a factor of four, $\Lambda(\chi) \leq \Lambda(\chi') \leq 4\Lambda(\chi)$. Therefore, simulation of a circuit involving t $\pi/8$ gates, m measurements, and g Clifford gates acting on a stabilizer state takes $\mathcal{O}\left(gn + m\left(4^t n + n^2\right) + \left(4^t n + tn^2\right)\right)$ time. If $t = \mathcal{O}(\log n)$ this is polynomial. The time remains sub-exponential for $t = \mathcal{O}(\text{polylog}(n))$.

  In general, k-qubit channels will not increase $\Lambda(\chi)$ by more than a factor of 16^k. Therefore, t such k-qubit channels, acting on a stabilizer initial state, can be efficiently handled by a generalized stabilizer if $tk = \mathcal{O}(\log(n))$. In other words, acting upon a stabilizer state, circuits in which the number of qubits that undergo non-stabilizer quantum evolution is of order $\log(n)$ are efficiently simulatable.

- What if we just restricted our circuit to Clifford gates, say g of them? In this case, we could take an arbitrary initial quantum state on n-qubits and simulate its evolution through the circuit in $O(gn)$ time, returning a generalized stabilizer at the output. We stress that any circuit that has no measurements is not a complete quantum computation, in which the output must be reported, through measurement, to the user. However, such a Clifford circuit could be a useful subroutine; for example, the encoding and decoding routines of stabilizer codes are Clifford circuits.

We summarize these observations here,

Corollary 15. Upper bounds on quantum circuit simulations

- Any quantum state τ can be simulated through a stabilizer circuit of g Clifford gates and m measurements in time $\mathcal{O}\left(gn + mn\left(\Lambda(\tau) + n\right)\right)$.
- Any quantum state τ can can be simulated through an quantum circuit of g Clifford gates, m measurements, and t, k-qubit channels in time at most $\mathcal{O}\left(gn + mn\left(16^{kt}\Lambda(\tau) + n\right) + n\left(16^{kt}\Lambda(\tau) + 4^k nt\right)\right)$.
- In particular, a stabilizer state can be simulated through an quantum circuit of g Clifford gates, m measurements, and t, k-qubit channels in time $\mathcal{O}\left(gn + mn\left(16^{kt} + n\right) + n\left(16^{kt} + 4^k nt\right)\right)$.

V. INNER PRODUCTS

In this section, we show how to calculate the product of the density matrices of two stabilizer states. This, in turn, leads to an algorithm for calculating the inner product of two stabilizer states, essentially equivalent to that of Aaronson and Gottesman [10]. Using the inner product algorithm for stabilizers, we present an algorithm for doing the same with generalized stabilizers. By the Cauchy-Schwarz inequality, this provides an equality test for generalized stabilizers.

A. Inner products for stabilizer states

By the inner product of two quantum states, we mean the Hilbert-Schmidt inner product, which is the number $\text{Tr}\left[\rho^\dagger\sigma\right] = \text{Tr}\left[\rho\sigma\right]$, for density matrices ρ and σ. This is a true inner product, in that it satisfies the three requisite properties, conjugate symmetry, linearity, and positivity [3]. We first consider calculating the product $\rho_{S'}\rho_S$ for n-qubit stabilizer states. We assume that S and S' are full stabilizers, since it suffices for application to generalized stabilizers, but the procedure can be straightforwardly extended to tableaus with a normalizer, which then represent mixed stabilizer states [10]. By equation (6), we certainly have

$$\rho_{S'}\rho_S = \prod_{s'\in S'}\frac{\mathcal{I}+s'}{2}\prod_{s\in S}\frac{\mathcal{I}+s}{2}, \tag{31}$$

where the products are over the generators of S' and S. We would like to reduce this product to

$$\rho_{S'}\rho_S = \prod_{u\in U} \frac{\mathcal{I}+u}{2} \tag{32}$$

for as small a set of Pauli operators U as possible. We call U the stabilizer union of S and S'. Note, U must be ordered, since elements of U may not commute, and the product (32) must therefore be ordered left to right. We will want to refer to the set of terms in the expansion of equation (32),

$$\mathcal{G}(U) \equiv \left\{ \prod_{x=1}^{|U|} u_x^{i_x} \,\middle|\, i \in \{0,1\}^{|U|} \right\}, \tag{33}$$

where again the products are ordered left to right. The size of U satisfies $|U| \geq n$. Also, we will see that ρ_S and $\rho_{S'}$ are orthogonal if and only if $-\mathcal{I} \in \mathcal{G}(U)$.

Once we have U, the inner product is calculated by expanding (32) and taking the trace of each term. Only the trace of the identity is nonzero and no product of u's from U can be the identity (or else U could be made smaller). Thus, matching the conclusion of Aaronson and Gottesman [10], we have

$$\mathrm{Tr}\left[\rho_{S'}\rho_S\right] = 2^{n-|U|}, \tag{34}$$

if ρ_S is not orthogonal to $\rho_{S'}$.

Equation (31) is not in the desired form, in general, since a generator $s' \in S'$ may be composed of generators from S, $s' = \mathcal{S}_i \in S$. If that is the case, then a portion of the product from (31) can be simplified

$$\begin{aligned}\left(\frac{\mathcal{I}+s'}{2}\right)\prod_{x=1}^{n}\left(\frac{\mathcal{I}+s_x}{2}\right)^{i_x} &= \left(\frac{\mathcal{I}+s'}{2}\right)\left(\frac{\mathcal{I}+\mathcal{S}_i}{2}\right)\prod_{x=1}^{n}\left(\frac{\mathcal{I}+s_x}{2}\right)^{i_x} \\ &= \left(\frac{\mathcal{I}+\mathcal{S}_i}{2}\right)\prod_{x=1}^{n}\left(\frac{\mathcal{I}+s_x}{2}\right)^{i_x} \\ &= \prod_{x=1}^{n}\left(\frac{\mathcal{I}+s_x}{2}\right)^{i_x},\end{aligned} \tag{35}$$

using the idempotence of the projectors $(\mathcal{I}+s)/2$. Note that if $-s' \in S$, then the same manipulations lead to zero, and thus ρ_S and $\rho_{S'}$ are orthogonal. We must also take into account cases where the product of two or more generators of S' is in S. For example, $s'_1, s'_2 \in S'$ and $s'_1, s'_2 \notin S$, but $s'_1 s'_2 \in S$, which results in simplification of the product (31) as well. We give an algorithm for calculating U given tableaus for the stabilizer states.

Theorem 16. *The stabilizer union U of two tableaus $\mathcal{T}' = (S', D')$ and $\mathcal{T} = (S, D)$ can be calculated in $\mathcal{O}(n^3)$ time.*

Proof. The idea is to keep and update a third tableau $\mathcal{U}$. This tableau contains "marked" generators, which, in the end, will be exactly the members of the stabilizer union U. First, initialize $\mathcal{U}$ to $\mathcal{T}$ and mark all the stabilizer generators from S. For some generator s'_i of S', decompose it in terms of $\mathcal{U}$ using the decomposition lemma.

If the decomposition of s'_i contains only marked items, then either s'_i or $-s'_i$ is in $\mathcal{G}(U)$ (the cases $\pm i s'_i \in \mathcal{G}(U)$ can be ruled out, because, as we will see, s'_i cannot anti commute with any marked generators from the destabilizer of $\mathcal{U}$). If $-s_i \in \mathcal{G}(U)$, then $\mathcal{G}(U)$ contains $-\mathcal{I}$ and we should report that $\mathcal{T}$ and $\mathcal{T}'$ represent orthogonal states. Otherwise, s'_i is in $\mathcal{G}(U)$ and no further action needs taken.

If the decomposition of s'_i contains unmarked generators, then replace one of these generators, say δ_k, from (the destabilizer of) $\mathcal{U}$ with s'_i. Mark s'_i; it is now part of the stabilizer union. However, we must preserve the commutation algebra of the tableau $\mathcal{U}$ so it can be used for further decompositions. Fix up $\mathcal{U}$ by multiplying any other generators from $\mathcal{U}$ that anticommute with s'_i by σ_k, the stabilizer generator from $\mathcal{U}$ conjugate to δ_k. This is comparable to how the measurement update is done on a tableau. No generator of S' can anti-commute with these marked destabilizers, because they are themselves members of S'.

Repeat this procedure for all generators s'_i of S'. Then, the marked generators of $\mathcal{U}$ form the stabilizer union U, since at each step we added a new generator to U if and only if adding it would increase the size of $\mathcal{G}(U)$. This procedure takes $\mathcal{O}(n^3)$ time, since we must perform tableau decomposition exactly n times. We impose the order on U that marked destabilizers from $\mathcal{U}$ should be listed before marked stabilizers. □

Two corollaries follow from theorem 16.

Corollary 17. Stabilizer states ρ_S and $\rho_{S'}$ are orthogonal if and only if there exists a $s' \in S'$ such that $-s' \in S$.

Corollary 18. Stabilizer bases $\mathcal{B}(S, D)$ and $\mathcal{B}(S', D')$ are mutually unbiased if and only if S' is a destabilizer for S.

The second references mutually unbiased bases. If $\{v_i\}$ and $\{u_i\}$ are both bases for the same vector space of dimension N, they are mutually unbiased if $v_i \cdot u_j = 1/\sqrt{N}$ for all i and j. We see that the stabilizer union of a stabilizer and its destabilizer has size $2n$, and the converse is true by the construction in theorem 16. Thus, $\mathrm{Tr}\left[\rho_S \rho_{S'}\right] = \left|\langle\psi_S|\psi_{S'}\rangle\right|^2 = 2^{-n}$ and the same is true for any other basis states, $\mathrm{d}_i|\psi_S\rangle$ and $\mathrm{d}'_j|\psi_{S'}\rangle$ for all i and j.

B. Inner products for generalized stabilizers

Armed with stabilizer inner products, generalized stabilizer inner products are fairly straightforward. As we will show, the inner product between $\tau = (\chi, \mathcal{B}(\mathcal{T}))$ and $\tau' = (\chi', \mathcal{B}(\mathcal{T}'))$ takes $\mathcal{O}(\Lambda(\chi)\Lambda(\chi')n^3)$ time. More importantly, the inner product algorithm, which we will present shortly, allows us to determine whether two generalized stabilizer states are equal. This is not always a trivial thing to do, because two generalized stabilizers with different stabilizer bases and different χ-matrices may represent the same state.

The Cauchy-Schwarz inequality on the Hilbert space of n-qubit density matrices states that

$$\left|\mathrm{Tr}\left[\tau'\tau\right]\right| \leq \sqrt{\mathrm{Tr}\left[\tau'^2\right]\mathrm{Tr}\left[\tau^2\right]}, \tag{36}$$

with equality if and only if $\tau = \tau'$. Thus, we will calculate both sides of the Cauchy-Schwarz inequality and check for equality. If we let $\tau = (\chi, \mathcal{B}(\mathcal{T}))$ and $\tau' = (\chi', \mathcal{B}(\mathcal{T}'))$, calculating the left side takes $\mathcal{O}\left(\Lambda(\chi)\Lambda(\chi')n^3\right)$ time, while the right side can be done in $\mathcal{O}\left(\Lambda(\chi) + \Lambda(\chi')\right)$ time, because $\mathrm{Tr}\left[\tau^2\right] = \sum_{ij}\left|\chi_{ij}\right|^2$.

The following theorem and constructive proof show how the inner product between generalized stabilizers can be calculated.

Theorem 19. *The inner product between two stabilizer states $\tau = (\chi, \mathcal{B}(S, D))$ and $\tau' = (\chi', \mathcal{B}(S', D'))$ can be calculated in time $\mathcal{O}(\Lambda(\chi)\Lambda(\chi')n^3)$.*

Proof. Write

$$\begin{aligned}\mathrm{Tr}\left[\tau'\tau\right] &= \sum_{ijhk} \chi'_{ij}\chi_{hk}\,\mathrm{Tr}\left[\mathrm{d}'_i\rho_{S'}\mathrm{d}'_j\mathrm{d}_h\rho_S\mathrm{d}_k\right] \\ &= \sum_{ijhk} \chi'_{ij}\chi_{hk}\,\mathrm{Tr}\left[\mathrm{d}'_i\mathrm{d}'_j\left(\mathrm{d}'_j\rho_{S'}\mathrm{d}'_j\right)\left(\mathrm{d}_h\rho_S\mathrm{d}_h\right)\mathrm{d}_h\mathrm{d}_k\right].\end{aligned} \tag{37}$$

Now we can certainly conjugate tableaus $\mathcal{T}$ and $\mathcal{T}'$ by the indicated destabilizers. Then find the stabilizer union and product tableau $\mathcal{U} = (S_\mathcal{U}, D_\mathcal{U})$ of the conjugated tableaus in $\mathcal{O}(n^3)$ time. Finally, decompose $x = \mathrm{d}_h\mathrm{d}_k\mathrm{d}'_i\mathrm{d}'_j$ in terms of the tableau $\mathcal{U}$, taking $\mathcal{O}(n^2)$ time for both the multiplication required to calculate x and the decomposition. If x is part of the stabilizer union (those generators in $\mathcal{U}$ that are marked), then say $x = \alpha\sigma_c\delta_b$ for destabilizer $\delta_b \in D_\mathcal{U}$ and $\sigma_c \in S_\mathcal{U}$. In this case, the $(ijhk)$ term of equation (37) equals $\alpha\chi'_{ij}\chi_{hk}$. Otherwise, if x cannot be decomposed into only marked generators from $\mathcal{U}$, the trace in the $(ijhk)$ term will vanish.

This series of steps must be done for each nonzero product $\chi'_{ij}\chi_{hk}$, for a total of $\mathcal{O}\left(\Lambda(\chi)\Lambda(\chi')\left(n^3 + n^2\right)\right) = \mathcal{O}\left(\Lambda(\chi)\Lambda(\chi')n^3\right)$ time. □

We note that there is high redundancy here, since a tableau $\mathcal{T}$ conjugated by destabilizer elements differs from the original only by negative signs on stabilizer generators. So, with more clever accounting of these signs, this calculation of the inner product could proceed with just one stabilizer product calculation, taking time $\mathcal{O}\left(\Lambda(\chi)\Lambda(\chi')n^2 + n^3\right)$.

VI. CONCLUSION

We have presented a representation of a quantum state for a classical computer that combines the fast updates of stabilizer states with the generality of density matrices. We also found update efficiencies of our representation through unitary gates, measurements, and quantum channels, and we noted their dependence on the sparsity of the density matrix or, in our notation, $\Lambda(\chi)$, the number of nonzero elements in matrix χ. It seems unlikely that $\Lambda(\tau)$ defines a metric for distance of a given state τ from the set of stabilizer states, though it is a heuristic measure for this property. We also noted that our simulations are linear in the number of measurements in the circuit, because

our state representation is never made more complicated by a measurement. This seems natural; measuring simplifies quantum states through collapse.

An interesting question that we have touched on is, exactly what set of quantum states can be simulated efficiently through every stabilizer circuit? It is certainly more than just the set of stabilizer states; as we have observed, states τ such that $\Lambda(\tau)$ is polynomial in the number of qubits have efficiently simulated stabilizer circuit evolution. This is a sufficient property for states that answer our question. Unfortunately, what conditions are both necessary *and* sufficient is unresolved. The set of states that are stabilizer circuit efficient is related to the set of magic states, those states that, when combined with stabilizer circuits, allow universal quantum computation [18]. The problems are essentially complementary; no magic state will be simulatable through every stabilizer circuit (assuming BQP is larger than BPP, that is). With qudits of odd prime dimension, it was recently found by Veitch *et. al.* that a sufficient condition for a state to be efficiently (weakly) simulated through stabilizer circuits is that a certain quasi-probability representation of the state be positive. They also conjecture this condition is necessary [5].

There are some possible improvements to our representation. In our current simulation scheme, as laid out by theorem 13, we would have to create two updated generalized stabilizers upon encountering a measurement. However, it is also possible to simply treat the measurement results as symbolic bits, say $a_1, a_2, \ldots, a_m$ for m measurements. These symbols would then appear in the tableau and χ-matrix, so that, by the end of simulation, we would have a generalized stabilizer representing the state after any sequence of measurement results; just plug in the appropriate values for each a_i. Conceivably, the use of symbolic manipulation will slow the calculation down, especially if elements of the χ-matrix accumulate many terms. However, this method of simulation may be beneficial in some instances. If very complicated channels $\mathcal{E}$ with large $\Lambda(\mathcal{E})$ are involved, performing algorithm 3 just once could be worth the overhead of symbolic manipulation.

A generalized stabilizer $(\chi, \mathcal{B}(\mathcal{T}))$, in some sense, separates the "classical" part of the quantum state from the quantum. The quasi-classical tableau $\mathcal{T}$ updates through Clifford gates and measurements, while the the χ-matrix is updated by non-Clifford operations. It is, in fact, possible to never update the χ-matrix until the very end of the circuit. In such a divided simulation, when we encounter Clifford gates or measurements we perform the appropriate update to the tableau, as usual. In the case of measurements, we also store the decomposition of the measurement operator, (α, b, c), such that $M = \alpha \mathfrak{d}_b \mathfrak{s}_c$ using the notation from theorem 13. Because b and c are just binary vectors marking positions in the tableau, and the order of stabilizers and destabilizers in the tableau does not change through any update routine, we can use this decomposition at *any* time to run algorithm 2. Likewise, upon encountering a non-Clifford gate or channel, we store the Kraus operator decompositions $\{(\alpha_h, b_h, c_h)\}$ and the coefficients from the operator-sum, to be used at any time with algorithm 3. Notice that the updates to the χ-matrix, when they are eventually done, must be done in chronological order. Thus, simulating the classically difficult part of the circuit, the evolution of the χ-matrix, can be separated from the classically easy part of the circuit, updating the tableau. From a practical point of view, the algorithms 2 and 3 are simply $2^n \times 2^n$ matrices, or superoperators, that are applied to χ. Multiplying these matrices with one another before application to χ might very well be faster, if χ is dense, than multiplying χ by each in turn. Collecting all such superoperators at the end of the circuit, then determining the optimal multiplication order, makes such a simplification possible.

We close by connecting our representation to quantum physics, motivated by the observation that the evolutions of the tableau and χ-matrix can be separated as described in the previous paragraph. Each circuit element can be viewed as a Hamiltonian that is turned on for some period of time, leading to evolution by the operator $C_k = e^{-iH_k T}$. To implement channels (or measurement) this way, we have to include an environment (*e.g.* measuring apparatus), but let's assume we have done so. Now, part, called H_k^S, of that Hamiltonian will correspond to a stabilizer evolution, a Clifford gate or Pauli measurement. The rest can be treated as a perturbation, so that $H_k = H_k^S + H_k'$. We can simulate such a perturbation using the interaction picture of quantum mechanics. In this framework, operators evolve in time by the unperturbed Hamiltonian H_k^S, leading to an update to the tableau of $\mathcal{T}' = C_k^S \mathcal{T} \left(C_k^S\right)^\dagger$ where $C_k^S = e^{-iH_k^S T}$. The interaction picture state χ_I, evolves according to the interaction Hamiltonian $H_k^I(t) = e^{iH_k^S t} H_k' e^{-iH_k^S t}$. When we decompose measurement and channel operators in terms of the tableau, we are actually doing this conversion, from H_k' to H_k^I. Then algorithms 2 and 3 evolve χ_I as dictated by the Schwinger-Tomonaga interaction equation, $i\hbar\partial_t \chi_I = \left[H_k^I, \chi_I\right]$. The interaction state χ_I is related to the traditional representation of a state in the computational basis by exactly the unitary C_k^S. A generalized stabilizer keeps two parts, a stabilizer basis which undergoes evolution by the unperturbed Hamiltonian, and the χ matrix which undergoes evolution due to the interaction Hamiltonian. This interpretation might lead to ways to view some unitary operations as perturbations on a Clifford gate.

References

[1] P. W. Shor, SIAM J. Comput., 26(5), 1484 (1994).
[2] R. Josza and N. Linden, Proc. R. Soc. London A **459**, 2011 (2003).
[3] M. Nielson and I. Chuang, *Quantum Computation and Quantum Information*, Cambridge University Press, 2000.
[4] G. Passante, *et. al.*, Phys. Rev. A **84**, 044302 (2011).
[5] V. Veitch, C. Ferrie, D. Gross, J. Emerson, New J. Phys. 14, 113011 (2012).
[6] A. Mari, J. Eisert, Phys. Rev. Lett. **109**, 230503 (2012).
[7] D. Gottesman, CalTech Ph.D. thesis, arXiv:quant-ph/9705052 (1997).
[8] D. Gottesman, Proceedings of the XXII International Colloquium on Group Theoretical Methods in Physics, edited by S.P. Corney *et al.* (International Press, Cambridge, MA, 1999), p. 32.
[9] M. Van den Nest, Quant. Inf. Comp. 10, 0258 (2010).
[10] S. Aaronson and D. Gottesman, Phys. Rev. A **70**, 052328 (2004).
[11] S. Anders and H. Briegel, P, Phys. Rev. A **73**, 022334 (2006).
[12] S. Bartlett and B. Sanders, Phys. Rev. Lett. **88**, 097904 (2002).
[13] J. Bermejo-Vega and M. Van den Nest, arXiv:1210.3637 (2012).
[14] A. Calderbank and P. Shor, Phys. Rev. A, **54**(2), pp. 1098-1106, (1996).
[15] A. Steane, Phys. Rev. Lett. **77**, 003190 (1996).
[16] S. Gay, arXiv:1112.2156 (2011).
[17] S. Anders, University of Innsbruck Ph.D. thesis, http://www.ebi.ac.uk/systems-srv/mp/file-exchange/, (2007).
[18] S. Bravyi and A. Kitaev, Phys. Rev. A **71**, 022316 (2005).