Entanglement on mixed stabilizer states: normal forms and reduction procedures

Koenraad M R Audenaert and Martin B Plenio
Blackett Laboratory, Imperial College London, Prince Consort Road, London SW7 2BW, UK
Institute for Mathematical Sciences, Imperial College London, Exhibition Road, London SW7 2BW, UK
E-mail: k.audenaert@imperial.ac.uk

New Journal of Physics 7 (2005) 170
Received 6 May 2005
Published 11 August 2005
Online at http://www.njp.org/
doi:10.1088/1367-2630/7/1/170

Abstract. The stabilizer formalism allows the efficient description of a sizeable class of pure as well as mixed quantum states of $n$-qubit systems. That same formalism has important applications in the field of quantum error correcting codes, where mixed stabilizer states correspond to projectors on subspaces associated with stabilizer codes. In this paper, we derive efficient reduction procedures to obtain various useful normal forms for stabilizer states. We explicitly prove that these procedures will always converge to the correct result and that these procedures are efficient in that they only require a polynomial number of operations on the generators of the stabilizers. On one hand, we obtain two single-party normal forms. The first, the row-reduced echelon form, is obtained using only permutations and multiplications of generators. This form is useful to calculate partial traces of stabilizer states. The second is the fully reduced form, where the reduction procedure invokes single-qubit operations and CNOT operations as well. This normal form allows for the efficient calculation of the overlap between two stabilizer states, as well as of the Uhlmann fidelity between them, and their Bures distance. On the other hand, we also find a reduction procedure of bipartite stabilizer states, where the operations involved are restricted to be local ones. The two-party normal form thus obtained lays bare a very simple bipartite entanglement structure of stabilizer states. To wit, we prove that every bipartite mixed stabilizer state is locally equivalent to a direct product of a number of maximally entangled states and, potentially, a separable state. As a consequence, using this normal form we can efficiently calculate every reasonable bipartite entanglement measure of mixed stabilizer states.
1. Introduction

The exploration of the properties of quantum entanglement is one of the main branches of quantum information theory [1]–[3]. While a reasonably detailed understanding of two-qubit entanglement has been achieved, the entanglement properties of higher-dimensional or multi-particle systems remain largely unexplored, with only isolated results [4]. This is largely due to the complexity involved in these investigations, which, in turn, originates from the tensor product structure of the multi-particle state space. This structure leads to an exponential growth in the number of parameters that are required for the description of the state. The same problem arises when one attempts to consider the time evolution of a many-body quantum system or, say, of a quantum computation. Generally, a significant part of the Hilbert space and consequently an exponential number of parameters are required to describe the quantum system at all times.

One way of approaching this situation is to impose constraints on the set of states and/or the set of operations that one is interested in without curtailing the variety of possible qualitative entanglement structures too much. In this context an interesting class of states that arises is that of stabilizer states [5]–[10] which, via the concept of graph states [11]–[18] have some connection with graph theory. The feature of these states that allows for a more detailed study of their entanglement properties is the fact that an $n$-particle stabilizer state is determined as the
joint unique eigenvector with eigenvalue +1 of a set of only $n$-tensor products of Pauli operators. This results in a very compact description of the quantum state requiring only of order $O(n^2)$-parameters and therefore, provides hope that a more detailed understanding of their entanglement structure can be obtained. Despite this simplification, stabilizer states still exhibit multi-particle entanglement and permit, for example, the violation of local realism [15].

The stabilizer formalism not only allows for the efficient description of a certain type of quantum states, but also permits the efficient simulation of a restricted, but nevertheless interesting, class of time evolutions, namely those that respect the tensor product structure of the Pauli operators [19]–[22]. Again, these simulations can be performed in polynomial time in the number of qubits, in stark contrast to the simulation of a general time evolution of an $n$-qubit system, which requires an amount of resources that is exponential in $n$.

The stabilizer formalism uniquely specifies the quantum state of an $n$-qubit system employing only polynomial resources. This alone, however, is not sufficient. It is also important to be able to derive all relevant physical quantities, especially those relating to entanglement, directly from the stabilizer formalism. Indeed, having first to deduce the state explicitly and then computing the property from the state would generally involve an undesirable exponential overhead in resources. While one can expect a direct approach to be possible in principle, it is evident that detailed and explicit presentations of algorithms to achieve these tasks in a systematic way and whose convergence is proven are of significant interest. In the context of entanglement properties some effort has recently been expended in this direction in [23], where, employing sophisticated tools from group theory, the existence of a useful entanglement measure for multi-particle stabilizer states was demonstrated.

The present work progresses further in a similar direction. Employing elementary tools we present a number of normal forms for pure and mixed stabilizer states, together with explicit and detailed descriptions of algorithms (including proofs of convergence) that allow the reduction to these normal forms. In turn, these normal forms then permit us to compute any entanglement measure, overlaps between stabilizer states and various other quantities. Detailed descriptions of the algorithms are provided that should make it straightforward to implement these algorithms in any programming environment and we are able to provide a ($\beta$-tested) suite of MatLab programs on request.

This suite can then form the basis for more detailed studies and further applications of the stabilizer formalism to a whole range of physical questions (see also [20]). This will be reported on in a forthcoming publication.

The stabilizer formalism also plays a central role in the field of quantum error correcting codes. Mixed stabilizer states (defined in section 2) are in one-to-one correspondence with projectors on subspaces associated with stabilizer codes [5]. Although our normal forms and reduction procedures have been designed with applications to entanglement theory in mind, they might have a bearing on stabilizer codes as well.

The present paper is organized as follows: in section 2, the basic notations and conventions we use are introduced while section 3 describes the elementary operations that will form the basis of all reduction procedures.

The single-party normal forms are the topic of sections 4–6. Section 4 deals with the so-called row-reduced echelon form (algorithm RREF), the reduction to which is based on row operations only. It allows to check independence of any (putative) set of generators and to calculate partial traces (algorithm PTRACE). In section 5, we describe the full reduction procedure (algorithm CNF1) to single-party normal form, using row and column (qubit) operations. In section 6,
we present an algorithm (algorithm OVERLAP) that is based on the full reduction and allows us to calculate overlaps between stabilizer states, Uhlmann fidelity, and Bures distance.

In section 7, we turn to the bipartite case, where we prove that the bipartite entanglement structure of stabilizer states is remarkably simple. To wit, we show that mixed bipartite stabilizer states are locally equivalent to a tensor product of a certain number of maximally entangled 2-qubit states and, potentially, a fully separable mixed state. We present an algorithm (algorithm CNFP) to obtain the number of these maximally entangled pairs, allowing for the calculation of any reasonable bipartite entanglement measure.

We conclude the description of our findings in section 8.

2. Notations and conventions

A ‘stabilizer operator’ on $N$ qubits is a tensor product of operators taken from the set of Pauli operators

\[
X := \left( \begin{array}{cc} 0 & 1 \\ 1 & 0 \end{array} \right), \quad Y := \left( \begin{array}{cc} 0 & -i \\ i & 0 \end{array} \right), \quad Z := \left( \begin{array}{cc} 1 & 0 \\ 0 & -1 \end{array} \right),
\]

and the identity $\mathbb{1}$. An example for $N = 3$ would be the operator $g = X \otimes \mathbb{1} \otimes Z$. A set $G = \{g_1, \ldots, g_K\}$ of $K$ mutually commuting stabilizer operators that are independent, i.e. $\prod_{i=1}^{K} g_i^s = \mathbb{1}$ exactly if all $s_i$ are even, are called the ‘generator set’ for the ‘stabilizer group’ $S$.

This stabilizer group $S$ then consists of all distinct products of operators from the generator set. For $K = N$, a generator set $G$ uniquely determines a single state $|\psi\rangle$ that satisfies $g_k |\psi\rangle = |\psi\rangle$ for all $k = 1, \ldots, N$. Any state for which such a generator set exists is called a ‘stabilizer state’. Such a state has trivially the property that $g_k |\psi\rangle \langle \psi| = |\psi\rangle \langle \psi|$ for all $k$ so that

\[
|\psi\rangle \langle \psi| = \frac{1}{2^N} \sum_{g \in S} g.
\]

This formula depends on the complete set of stabilizers, and is, therefore, not very practical. The following formula expresses the state in terms of a generator set [8]

\[
|\psi\rangle \langle \psi| = \prod_{k=1}^{N} \frac{\mathbb{1} + g_k}{2}.
\]

The procedure presented in section 4 yields as a side result an elementary proof of this statement.

Considering two parties $A$ and $B$, the reduced density matrix of the stabilizer state can be computed as

\[
\rho_A = \text{Tr}_B \rho = \frac{1}{2^N} \sum_{g \in S} \text{Tr}_B g.
\]

This obviously means that only operators $g$ contribute that have identity operators acting on all qubits belonging to $B$. Needless to say, computing $\rho_A$ from $\rho$ directly is hopelessly inefficient; as there are $2^N$ different $g$, this task requires an exponential number of operations in general.
It turns out, however, that there is a class of mixed states that can also be characterized employing stabilizers. We will call these ‘mixed stabilizer states’, and they contain the pure stabilizer states as a subset. The important feature of this class is that the reduced density matrix of a mixed stabilizer state is again a mixed stabilizer state. Furthermore, as we will show below in section 4.2, the stabilizer group of a reduction can be efficiently calculated directly from the original stabilizer group, without calculating the state and its reduction explicitly.

To characterize mixed stabilizer states, one simply considers sets $G$ that are linearly dependent. As a consequence, by multiplying stabilizers, one can achieve that some of them become identical to 11 and only $K$ linearly independent ones remain. Then the common eigenspace of these $K$ operators will have a dimension larger than 1. As in equation (3), one immediately deduces that the density operator is again just the projector onto this eigenspace, rescaled to trace 1:

$$\mathcal{P} = \prod_{k=1}^{K} \frac{1 + g_k}{2}, \quad \rho = \frac{1}{2^{N-K}} \mathcal{P}.$$ (5)

Given that $\mathcal{P}$ is a projector on to a subspace of dimension $2^{N-K}$, the entropy of $\rho$ is simply $N - K$. In analogy with matrix analysis terminology, we will call $K$ as the ‘rank’ of the stabilizer group. Stabilizer groups with $K = N$ will be called ‘full-rank’ and stabilizer groups with $K < N$ as ‘rank-deficient’.

**Note:** In the case of a rank-deficient stabilizer group, one has to distinguish between ‘stabilizer’ states and ‘stabilized’ states. The stabilizer state is the one given by (5) and the stabilizer formalism allows us to study its properties in an efficient way. On the other hand, there are many states that are stabilized by that same stabilizer group, but in general they are not stabilizer states. Indeed, most of these stabilized states cannot be described as ‘the unique state stabilized by a full-rank stabilizer group’, and hence, the stabilizer formalism cannot be used to study their properties via that group. For example, any state is stabilized by the (singleton) stabilizer group $\{1\}$, but only the maximally mixed state $1/2$ is the stabilizer state corresponding to that group.

For the following, the aim will be to derive basic entanglement properties such as the entropy of entanglement or the logarithmic negativity for stabilizer states, pure or mixed, directly from their generating set. To this end, it will be useful to find a normal form for the generator set that reveals the relevant entanglement structure.

We now introduce the concept of ‘stabilizer array’, which is quite simply a rectangular array of $K$ rows and $N$ columns, where the elements are Pauli matrices or the Identity matrix. Specifically, the element in the $k$th row and $n$th column of the stabilizer array corresponding to a generator set $G = \{g_1, \ldots, g_K\}$ on $N$ qubits is the $n$th tensor factor (corresponding to qubit $n$) of the $k$th generator $g_k$. For some applications, it will also be necessary to deal with the generator phase factors. While, in general, these phase factors can assume the values $\pm 1$ and $\pm i$, for the purpose of describing stabilizer states only the values $\pm 1$ make sense (since states are Hermitian). We will store these phase factors in a $K$-dimensional vector $s$, where $s_k$ is the phase factor of generator $g_k$.

The purpose of the various normal forms that will be presented in this paper are to structure the set of stabilizer states into certain equivalence classes. They are similar in spirit to the normal forms that have been devised for matrices. For example, the row-reduced echelon form, which exhibits the rank of a matrix, has a counterpart for stabilizer arrays. Despite this similarity, the normal forms presented here are of an entirely different nature. The rank of a stabilizer array,
Table 1. Multiplication table for Pauli operators; shown is \( \sigma_{\text{row}} \cdot \sigma_{\text{col}} \).

<table>
<thead>
<tr>
<th></th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\mathbb{I})</td>
<td>(\mathbb{I})</td>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>(\mathbb{I})</td>
<td>(iZ)</td>
</tr>
<tr>
<td>Y</td>
<td>(-iZ)</td>
<td>(\mathbb{I})</td>
<td>(iX)</td>
</tr>
<tr>
<td>Z</td>
<td>(iY)</td>
<td>(-iX)</td>
<td>(\mathbb{I})</td>
</tr>
</tbody>
</table>

which we will introduce in section 4, is akin to the rank of a matrix in that it equals the number of independent generators in a generator set, but there the similarity ends. In linear algebra, one defines both row and column rank of a matrix and one proves that these two ranks are actually equal. For stabilizer arrays, one cannot even give a meaningful definition of column rank.

These differences ultimately boil down to the fact that a stabilizer array is not really a matrix. The two foremost reasons are that its elements are not numbers but elements of the Pauli group, and secondly, that matrices represent linear operations in linear spaces, while stabilizer arrays represent sets (namely, sets of generators). As a consequence, while operations like matrix transpose, matrix multiplication, addition and inverse make perfect sense for matrices, they are utterly meaningless for stabilizer arrays. The allowed operations on stabilizer arrays are thus much more restricted than in the matrix case. For example, the only row operations that make sense for stabilizer arrays are row interchange and elementwise row multiplication (which is based on the Pauli group multiplication law). This will be discussed in more detail in the following section. This explains the need for entirely new reduction procedures for stabilizer arrays.

3. Elementary operations

In this section, we describe the allowed elementary operations that transform a stabilizer array and which we will use to reduce an array to its normal forms. As in the matrix case, these operations come in two kinds. The first kind are the row operations. It is important to realize that row operations will not alter the stabilizer state at all, but only alter the generator set it is represented by. These are the ‘row transposition’, which interchanges (transposes) two rows in the stabilizer array, and the ‘row multiplication’, which multiplies one row with another one. The latter operation changes the generators of the stabilizer group, but not the group itself and hence not the stabilizer state either. We will use the phrase ‘multiply row \(k\) with row \(l\)’ to mean ‘multiply rows \(k\) and \(l\) elementwise and set row \(l\) to the product obtained’. The multiplication table for Pauli operators is shown in table 1.

The second kind of operations are the column operations, which may alter the state. The column operations we will use are a certain class of single-qubit operations, transposing two columns and the CNOT operation between two qubits. As single-qubit operations, we take those that act on one given column of the stabilizer operators by permuting the Pauli operators (in the given column) among themselves. These operations can be constructed from combinations of Hadamard gates (\(H\)) and \(\pi/4\) gates (\(P\)) (see table 2). Note that odd permutations must involve a sign change in one of the Pauli operators in order to correspond to a unitary operation. The particular sign changes of table 2 have been chosen to make the unitaries implementing the odd permutations involutory (apart from a global phase). That is, \(UU = \exp(i\phi)\mathbb{I}\). Note also that the second and third permutation in the table are each other’s inverse.
Table 2. Truth table for the single-qubit operations employed by the CNF algorithm. Any permutation of the set of Pauli operators can be achieved.

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>Unitary</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Y</td>
<td>Z</td>
<td>I</td>
</tr>
<tr>
<td>Z</td>
<td>X</td>
<td>Y</td>
<td>PH</td>
</tr>
<tr>
<td>Y</td>
<td>Z</td>
<td>X</td>
<td>H P^†</td>
</tr>
<tr>
<td>−X</td>
<td>Z</td>
<td>Y</td>
<td>PHP^†</td>
</tr>
<tr>
<td>Y</td>
<td>X</td>
<td>−Z</td>
<td>HPPHP^†</td>
</tr>
<tr>
<td>Z</td>
<td>−Y</td>
<td>X</td>
<td>H</td>
</tr>
</tbody>
</table>

Table 3. Truth table for the CNOT gate employed by the CNF algorithm. C and T refer to control and target qubit, respectively. The primed columns give the values after the operation.

<table>
<thead>
<tr>
<th>C</th>
<th>T</th>
<th>C'</th>
<th>T'</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>I</td>
<td>I</td>
<td>I</td>
</tr>
<tr>
<td>I</td>
<td>X</td>
<td>I</td>
<td>X</td>
</tr>
<tr>
<td>I</td>
<td>Y</td>
<td>Z</td>
<td>Y</td>
</tr>
<tr>
<td>I</td>
<td>Z</td>
<td>Z</td>
<td>Z</td>
</tr>
<tr>
<td>X</td>
<td>I</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>X</td>
<td>X</td>
<td>X</td>
<td>I</td>
</tr>
<tr>
<td>X</td>
<td>Y</td>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>X</td>
<td>Z</td>
<td>−Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

For the bipartite normal form described in section 7, we will need to divide the qubits into two parties and only allow operations that are local to those parties. Transposing two columns in the bipartite case is only allowed when both columns (qubits) belong to the same party. Otherwise this would be a non-local operation, which would very likely affect the amount of entanglement.

The CNOT gate between two qubits, one being the control qubit and the other the target qubit, operates on the two corresponding columns of the stabilizer array. In the case of a bipartite CNF, we must again ensure that both qubits belong to the same party. The truth table for the CNOT is given in table 3. Note that the column pertaining to the control qubit is modified too; this is a peculiarity of the description of states by stabilizers.

4. Row-reduced echelon form

While the Clifford normal form (CNF) of a stabilizer array will be obtained below via application of both elementary row and column operations, it is possible to obtain a normal form using elementary row operations only. Due to its similarity to the matrix case, we will call this normal form the row-reduced echelon form (RREF). The benefits of the RREF are that it is very easy to obtain, the stabilizer state represented by the stabilizer array is not changed, and it is applicable to states on any number of parties. Furthermore, as we shall see below, it is an efficient way to eliminate linearly dependent rows from the stabilizer array.

The general structure of the RREF is most easily described in a recursive fashion. There are three cases:

\[
\begin{pmatrix}
\mathbb{I} & \text{RREF}' \\
\vdots & \text{RREF}' \\
\mathbb{I} & \text{RREF}'
\end{pmatrix},
\quad
\begin{pmatrix}
\sigma & \ast \cdots \ast \\
\mathbb{I} & \text{RREF}' \\
\vdots & \text{RREF}' \\
\mathbb{I} & \text{RREF}'
\end{pmatrix}
\quad\text{and}\quad
\begin{pmatrix}
\sigma_1 & \ast \cdots \ast \\
\mathbb{I} & \text{RREF}' \\
\vdots & \text{RREF}' \\
\mathbb{I} & \text{RREF}'
\end{pmatrix}.
\]

The symbols ‘\(\vdots\)’ and ‘\(\cdots\)’ denote the number of repeated rows and columns. This number may also be zero. The symbol RREF' denotes a (possibly empty) subarray that is also in RREF form. The symbol ‘\(\ast\)’ denotes either a Pauli operator or an identity \(\mathbb{I}\). Furthermore, \(\sigma, \sigma_1\) and \(\sigma_2\) are Pauli operators, and \(\sigma_1\) and \(\sigma_2\) are anticommuting. We will refer to the operators in these positions as ‘column leaders’ of their column, and ‘row leaders’ of their row.

The RREF algorithm works by applying a sequence of elementary row operations to the stabilizer array. At every step of the algorithm it is determined which elementary operation to apply based on the values contained in a certain contiguous subarray of the full array. At every step of this subarray, which we will call the ‘active region’, either stays the same or decreases in size. The algorithm terminates when the size of the active region has decreased to zero. Note that the elementary operations operate on the full stabilizer array and not just on the active region.

Let \(K\) and \(N\) be the number of rows (generators) and columns (qubits) of the stabilizer array, respectively. The variable \(K_U\) contains the index of the first row in the active region, and \(N_L\) the index of its first column. The active region thus consists of the array elements \((i, j)\) for \(K_U \leq i \leq K\) and \(N_L \leq j \leq N\). Initially, the active region comprises the full stabilizer array, hence \(K_U = 1\) and \(N_L = 1\).

In this and subsequent sections, the phase factors \(\exp(i\phi_k)\) of the various generators will not be mentioned explicitly. They are best maintained under the form of a single additional column in the stabilizer array, which is modified by row permutations and the elementary operations of tables 1–3.

**Algorithm RREF**

1. Count the number of different Pauli operators (\(X, Y\) and \(Z\)) in the first column (\(N_L\)) of the active region, i.e. restricting attention only to rows \(K_U\) up to \(K\).

2. Three cases can be considered:
   
   (a) There are no Pauli operators in column \(N_L\).
       
       (i) Increase \(N_L\) by 1.

   (b) There is only one kind of Pauli operator.
       
       Let \(k\) be the first row in the active region where column \(N_L\) contains a Pauli operator.

       (i) Make row \(k\) the top row of the active region by transposing, if necessary, row \(k\) with row \(K_U\).

       (ii) Multiply row \(K_U\) with all other rows in the active region that have the same Pauli in column \(N_L\).

       (iii) Increase \(K_U\) and \(N_L\) by 1.


Table 4. Required operations to eliminate any Pauli operator from row 3 of the stabilizer array shown above. The operators \( \sigma_1, \sigma_2 \) and \( \sigma_3 \) are a permutation of \( X, Y \) and \( Z \).

Initial stabilizer array:
\[
\begin{pmatrix}
\sigma_1 \\
\sigma_2
\end{pmatrix}
\]

Depending on the content of row 3, do the following:
- \( \mathbb{I} \): Do nothing. \\
- \( \sigma_1 \): Multiply row 1 with row 3. \\
- \( \sigma_2 \): Multiply row 2 with row 3. \\
- \( \sigma_3 \): Multiply row 1 with row 3, and then row 2 with row 3.

(c) There are at least two different kinds of Pauli operators.

Let \( k_1 \) be the first row in the active region where column \( N_L \) contains a Pauli operator and \( k_2 \) be the first row in the active region where column \( N_L \) contains a different Pauli operator.

(i) Make row \( k_1 \) the top row of the active region by transposing, if necessary, row \( k_1 \) with row \( K_U \).

(ii) Make row \( k_2 \) the second row of the active region by transposing, if necessary, row \( k_2 \) with row \( K_U + 1 \).

(iii) Multiply every other row in the active region with either row \( K_U \), row \( K_U + 1 \), both rows or none, depending on the element in column \( N_L \) (see table 4).

3. If the active region still has non-zero size (\( N_L \leq N \) and \( K_U \leq K \)), continue with step 1, else terminate.

4.1. Checking independence of a set of generators

The easiest way to check independence of a set of generators is to compute the RREF of the stabilizer array. This fact is one another property the stabilizer RREF and the matrix RREF have in common. Dependencies between generators will show up as RREF rows contain only \( \mathbb{I} \) operators. Removing all these \( \mathbb{I} \) rows leaves an independent set of generators.

Proof. From the form of the RREF one observes that there cannot be more than two rows with the same number of leading \( \mathbb{I} \) operators, and if there are two such, they have a different row leader. Consider a subset of generators \( g_k \) in the RREF, having \( n_k \) as leading \( \mathbb{I} \) operators and having \( \sigma_k \) as row leaders. Let the generators be sorted according to \( n_k \) in ascending order. When multiplying two rows that satisfy \( n_2 \geq n_1 \), the number of leading \( \mathbb{I} \) operators in the product is \( n_1 \) and the row leader is either \( \sigma_1 \) (if \( n_2 > n_1 \)) or \( \sigma_1 \sigma_2 \) (if \( n_2 = n_1 \)), which is different from either \( \sigma_1 \) or \( \sigma_2 \). In both cases this shows that the product cannot occur as another generator in the RREF. This proves that it is not possible to write one RREF generator as a product of other RREF generators.

\[ \square \]
4.2. Partial trace of a stabilizer state

A useful and important operation is the partial trace. The RREF algorithm is the central part in the following efficient partial trace algorithm:

**Algorithm PTRACE**

(i) By column permutations bring the columns of the qubits to be traced out in first position.
(ii) Bring those columns to RREF.
(iii) Remove the rows containing the column leader(s).
(iv) Finally, remove those columns themselves.

**Proof.** To prove that this algorithm indeed calculates the partial trace, consider again the three cases for the RREF:

\[
\begin{pmatrix}
\mathbb{I} \\
\vdots \\
\mathbb{I}
\end{pmatrix}
\begin{pmatrix}
\text{RREF'}
\end{pmatrix},
\begin{pmatrix}
\sigma \\
\vdots \\
\mathbb{I}
\end{pmatrix}
\begin{pmatrix}
\ast \\
\vdots \\
\ast 
\end{pmatrix}
\begin{pmatrix}
\text{RREF'}
\end{pmatrix} 
\text{and} 
\begin{pmatrix}
\sigma_1 \\
\vdots \\
\mathbb{I}
\end{pmatrix}
\begin{pmatrix}
\ast \\
\vdots \\
\ast 
\end{pmatrix}
\begin{pmatrix}
\text{RREF'}
\end{pmatrix}.
\]

We have to show that the state described by RREF', say \(\rho'\), is the state obtained from the original stabilizer state \(\rho\) by tracing out the qubit pertaining to column 1. Denote the sequences of \(\ast\) operators by \(g\), \(g_1\) and \(g_2\), respectively.

Using equation (5), it is easy to see that, in the first case,

\[
\rho = \frac{\mathbb{I}}{2} \otimes \rho',
\]

and tracing out qubit 1 yields

\[
\text{Tr}_1 \rho = \rho'.
\]

In the second case,

\[
\rho = \frac{\mathbb{I} \otimes \mathbb{I} + \sigma \otimes g}{2} (\mathbb{I} \otimes \rho') = \frac{1}{2} (\mathbb{I} \otimes \rho' + \sigma \otimes g \rho'),
\]

and again, as Pauli operators have trace 0,

\[
\text{Tr}_1 \rho = \rho'.
\]

In the third and final case,

\[
\rho = \frac{\mathbb{I} \otimes \mathbb{I} + \sigma_1 \otimes g_1 \mathbb{I} \otimes \mathbb{I} + \sigma_2 \otimes g_2}{2} (\mathbb{I} \otimes 2 \rho')
\]

\[
= \frac{1}{2} (\mathbb{I} \otimes \rho' + \sigma_1 \otimes g_1 \rho' + \sigma_2 \otimes g_2 \rho' + \sigma_1 \sigma_2 \otimes g_1 g_2 \rho'),
\]

resulting yet again in

\[
\text{Tr}_1 \rho = \rho'.
\]

\(\Box\)
5. Single-party normal form

The CNF algorithm works by applying a sequence of elementary operations to the stabilizer array. At every step of the algorithm, it is determined which elementary operation to apply based on the values contained in a certain contiguous subarray of the full array. At every step this subarray, which we will call the ‘active region’, either stays the same or decreases in size. The algorithm terminates when the size of the active region has decreased to zero. Note that the elementary operations operate on the full stabilizer array and not just on the active region.

Let \( K \) and \( N \) be the number of rows (generators) and columns (qubits) of the stabilizer array, respectively. The variables \( K_U \) and \( K_L \) contain the indices of the first (uppermost) and last (lowermost) row in the active region, and \( N_L \) and \( N_R \) the indices of its first (leftmost) and last (rightmost) column. The active region thus consists of the array elements \((i, j)\) for \( K_U \leq i \leq K_L \) and \( N_L \leq j \leq N_R \). Initially, the active region comprises the full stabilizer array, hence \( K_U = 1, K_L = K, N_L = 1 \) and \( N_R = N \). We will prove below that after every iteration of the algorithm the stabilizer array has the block structure:

\[
\begin{pmatrix}
X & \ldots & \ldots & \ldots & \ldots & \ldots \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots \\
\end{pmatrix}
\]

The block containing the asterisks is the active region and has not yet been brought to normal form. The columns on the left of the active region correspond to qubits that are in an eigenstate of the \( X \) operator, the columns on its right correspond to qubits that are in a totally mixed state. The final form, after completion of the algorithm is:

\[
\begin{pmatrix}
X & \ldots & \ldots & \ldots & \ldots & \ldots \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots \\
\end{pmatrix}
\]

Here, we have left open the possibility that the rows of the initial stabilizer array might not be independent.

**Algorithm CNF1**

1. Count the number of different Pauli operators (\( X, Y \) and \( Z \)) in the first column \( (N_L) \) of the active region, i.e. restricting attention only to rows \( K_U \) up to \( K_L \).
2. Three cases can be considered:
   (a) There are no Pauli operators in column $N_L$.
       (i) If necessary, transpose column $N_L$ with column $N_R$.
       (ii) Decrease $N_R$ by 1.
   (b) There is only one kind of Pauli operator.
       Let $k$ be the first row in the active region where column $N_L$ contains a Pauli operator.
       (i) Make row $k$ the top row of the active region by transposing, if necessary, row $k$ with row $K_U$.
       (ii) Apply whatever single-qubit operation on column $N_L$ that brings that Pauli operator to an $X$.
       (iii) Multiply row $K_U$ with all other rows in the active region that have an $X$ in column $N_L$.
       (iv) Consider the elements of the first row of the active region (row $K_U$). To each of the columns beyond the first one that contains in the first row a Pauli different from $X$, apply a single-qubit operation to turn it into an $X$.
       (v) To each of these columns, which now have an $X$ in the first row, successively apply a CNOT operation with control column $N_L$.
       (vi) Increase $K_U$ and $N_L$ by 1.
   (c) There are at least two different kinds of Pauli operators.
       Let $k_1$ be the first row in the active region where column $N_L$ contains a Pauli operator, and $k_2$ be the first row in the active region where column $N_L$ contains a different Pauli operator.
       (i) Make row $k_1$ the top row of the active region by transposing, if necessary, row $k_1$ with row $K_U$.
       (ii) Make row $k_2$ the second row of the active region by transposing, if necessary, row $k_2$ with row $K_U + 1$.
       (iii) Bring the element on row $K_U$ to an $X$ and the element on row $K_U + 1$ to a $Z$ by applying, if necessary, a single-qubit operation on column $N_L$.
       (iv) Consider the first two rows of the active region (rows $K_U$ and $K_U + 1$). Find the first column beyond column $N_L$, say column $l$, that contains an anticommuting pair on those rows (i.e. two non-identical Pauli operators).
       (v) Bring the anticommuting pair to an $(X, Y)$ pair by applying, if necessary, a single-qubit operation to that column.
       (vi) Apply a CNOT operation to that column, with column $N_L$ as control.
       (vii) The extent of the active region is not changed in this case.

3. If the active region still has non-zero size ($N_L \leq N_R$ and $K_U \leq K_L$), continue with step 1, else terminate.

5.1. Proof of correctness of algorithm CNF1

We will now show that algorithm CNF1 indeed brings any stabilizer array into its normal form. We consider the three cases (a), (b) and (c) in succession.
5.1.1. Case (a). This case corresponds to column $N_L$ containing $\mathbb{I}$ only and therefore belongs to the block right of the active region. Step (a.i) does just that and step (a.ii) subsequently excludes this column from the active region.

5.1.2. Case (b). This case corresponds to a column containing $\mathbb{I}$ operators and Pauli operators of just one kind. Step (b.i) brings the first of these Pauli operators to the top row, with the ultimate goal of excluding this row from the active region. Step (b.ii) applies a single-qubit rotation to bring the Pauli operators in standard form, which in this case is an X operator.

In step (b.iii), the column is then ‘cleaned up’. Through multiplying the top row $K_U$ with other rows containing an X in column $N_L$, we obtain a stabilizer array that is still describing the same state but contains only one X in column $N_L$. So this column is already in standard form.

However, the top row is not in standard form yet. Step (b.iv) applies an appropriate single-qubit operation to every column in the active region, except for the first one, so that the first row contains either $\mathbb{I}$ or X operators. Step (b.v) then performs a ‘row cleanup’, by applying CNOTs to the columns starting with an X, the first column being the control column. The target X operators are thereby turned into $\mathbb{I}$, leaving us with a first row of the form $(X, \mathbb{I}, \ldots, \mathbb{I})$.

It is not a priori clear, however, that step (b.v) is not undoing the cleanup of column $N_L$ by step (b.iii). Nevertheless, inspection of the CNOT truth table reveals that the $\mathbb{I}$ operators in column $N_L$ can either be turned into a Z or remain $\mathbb{I}$, by any number of CNOTs. Although a Z operator can actually occur during the execution of step (b.iii), in the end all operators will be turned back into $\mathbb{I}$. This must be so because the top row of the active region is turned into $(X, \mathbb{I}, \ldots, \mathbb{I})$, which does not commute with a row starting with a Z. So the assumption of commutativity of the generators ensures that step (b.iii) is not undone by step (b.v).

Finally, we note that both the first row and the first column are now in standard form and can be removed from the active region, which is done in step (b.vi). The top left block in the normal form array hereby receives one further X operator.

5.1.3. Case (c). The most difficult case to investigate is case (c), because here it is not a priori clear that any progress is made within an iteration. Indeed, the extent of the active region is not changed and it is not clear that further iterations will eventually escape from case (c), resulting potentially in an infinite loop.

However, every execution of case (c) does result in measurable progress. As can be seen from the truth table of the CNOT operation, the end result of the CNOT in step (c.vi) is

\[
\begin{array}{c|ccc}
K_U & N_L & I & N_L' & I' \\
K_U + 1 & X & X & X & Z \\
\end{array}
\]

Hence, a $\mathbb{I}$ is introduced in row $K_U$ where there originally was none. Furthermore, no further algorithmic step in case (c) ever touches this element again, by the following reasoning.

- The only operations that do change the top row $K_U$ are the transposition in step (c.i), and the CNOT in step (c.vi).
- Step (c.i) is executed at most once before the algorithm breaks out of the (c) case, namely, at the very beginning. This is because the X brought in the top left position is not changed by the CNOT.

The first CNOT that operates on target column $l$ introduces the $\mathbb{I}$ there. In further iterations, the CNOT will not operate on column $l$ a second time, because step (c.iv) sets the target column to a column containing an anticommuting pair in the top two rows, and the $\mathbb{I}$ created in the top of column $l$ does not form part of an anticommuting pair.

It is now easy to see why the algorithm must eventually break out of the (c) case. Every iteration through this case increases the number of $\mathbb{I}$ operators in the top row by 1, but there are only a limited number of places (columns) available to do this. Hence, the number of successive iterations through case (c) must be limited too.

The algorithm breaks out of the loop through the (c) case when there are no further anticommuting pairs in column $N_L$. As a consequence, the algorithm will then either execute case (a) or case (b), thereby again reducing the extent of the active region.

5.2. Alternative proof of projection formulas (3) and (5)

In this subsection, we present a new proof of the equivalence of the expressions (2) and (3) for a pure stabilizer state, and of (4) and (5) for a mixed stabilizer state.

By the proof of the CNF1 algorithm, a state described by a certain stabilizer array is unitarily equivalent to the state described by the normal form of that array. Let the initial stabilizer group $S$ be given by a stabilizer array. Let $S'$ be the stabilizer group described by the normal form of that array. The $K$ generators of $S'$ are of the form

$$g'_k = \mathbb{I} \otimes \cdots \otimes X \otimes \cdots \otimes \mathbb{I},$$

with the $X$ operator in the $k$th tensor factor. The stabilizer state corresponding to the normal form is therefore

$$\rho' = \frac{1}{2^N} \sum_{i_1, \ldots, i_K \in \{0,1\}} X^{i_1} \otimes \cdots \otimes X^{i_K} \otimes \mathbb{I}^{\otimes N-K}$$

$$= \left( \frac{\mathbb{I} + X}{2} \right)^{\otimes K} \otimes (\mathbb{I}/2)^{\otimes N-K}$$

$$= \frac{1}{2^{N-K}} \prod_{k=1}^{K} \frac{\mathbb{I} + g'_k}{2}.$$ 

Let $U$ be the unitary corresponding to the sequence of elementary operations that brought the stabilizer array to its normal form. To wit, $S$ consists of the elements $g = Ug'U^\dagger$, $g' \in S'$, and can be generated by the generators $g_k := Ug'_kU^\dagger$. Then the stabilizer state corresponding to $S$ is given by

$$\rho = \frac{1}{2^N} \sum_{g \in S} g$$

$$= \frac{1}{2^N} \sum_{g' \in S'} Ug'U^\dagger$$

$$= U \rho' U^\dagger.$$
6. Fidelity between stabilizer states

The topic of this section is an algorithm to calculate the overlap $F = \text{Tr}[\rho_1 \rho_2]$ between two mixed stabilizer states $\rho_1$ and $\rho_2$ directly from their $K_1 \times N$ and $K_2 \times N$ stabilizer arrays $A_1$ and $A_2$.

While the overlap between two states is certainly an interesting quantity, the Bures distance $D(\rho_1, \rho_2) := 2\sqrt{1 - F_u(\rho_1, \rho_2)}$, where $F_u$ is the Uhlmann fidelity $F_u(\rho_1, \rho_2) := \text{Tr} \left[ \sqrt{\sqrt{\rho_1} \rho_2 \sqrt{\rho_1}} \right]$, is a much more desirable quantity, as it is an actual distance measure and has a much nicer interpretation. It is well-known that for pure states the Uhlmann fidelity between two states is just the square root of their overlap, while for general mixed states there is no such relation. It will turn out that with just a minor modification, the algorithm is also able to calculate the Uhlmann fidelity. This allows us to calculate the overlap, the Uhlmann fidelity and Bures distance for stabilizer states in one go.

For the calculation of the overlap (and fidelity), it is imperative to take the generator 'phases' into account. We will use the vectors $S_1$ and $S_2$ for that purpose. The elementary row operations of multiplication and permutation of stabilizer rows are understood to treat the phase vector as an additional column of the stabilizer array. Furthermore, row multiplication, single-qubit rotation and CNOT operation have to multiply the appropriate generator phase with the phase factor mentioned in their truth tables.

Algorithm OVERLAP

1. Construct the $(K_1 + K_2) \times N$ composite array $A$ and the composite vector $S$ of generator phases:

\[
A = \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}, \quad S = \begin{pmatrix} S_1 \\ S_2 \end{pmatrix}.
\]

1 This composite array is no longer a stabilizer array because generators from $A_1$ need not commute with those of $A_2$. Even worse, $A_1$ and $A_2$ may generate conflicting stabilizers, i.e. with opposite phase factors. One thus cannot just apply reduction algorithms, say RREF, to $A$ as a whole, and claim to have calculated the rank of $A$; in fact, the rank of $A$ is just not defined. There are, however, some situations where one can treat the composite array, with due care, as if it were a stabilizer array. This is especially useful when one needs to apply the same column operations to $A_1$ and $A_2$. 
2. By applying the CNF1 algorithm to the composite array $A$ (and its vector $S$ of generator phases), with initial active region set to the full $A_1$ part (excluding $A_2$!), the $A_1$ part is brought to CNF form, while automatically applying the same sequence of column operations to the $A_2$ part. Let $R_1$ be the number of $X$ operators in this CNF.

3. Set the active region to all the rows of the $A_2$ part and all the columns for which $A_1$ contains $X$ operators. That is, $K_U = K_1 + 1$, $K_L = K_1 + K_2$, $N_L = 1$ and $N_R = R$. Set $T = 1$.

4. Count the number of different Pauli operators ($X$, $Y$ and $Z$) in the first column ($N_L$) of the active region, i.e. restricting attention only to rows $K_U$ up to $K_L$.

5. Three cases can be considered:

   (a) There are no Pauli operators in column $N_L$.

      (i) Do nothing.

   (b) There is only one kind of Pauli operator.

      Let $k$ be the first row in the active region where column $N_L$ contains a Pauli operator.

      (i) Make row $k$ the top row of the active region by transposing, if necessary, row $k$ with row $K_U$.

      (ii) Multiply row $K_U$ with all other rows in the active region that have a Pauli operator (necessarily equal to the one on row $K_U$) in column $N_L$.

      (iii) Let $P$ be the element in column $N_L$ on row $K_U$ (the column leader).

         • If $P$ is not an $X$, divide $T$ by 2 and increase $K_U$ by 1.

         • If $P$ is an $X$, multiply row $N_L$ (that is, the row containing an $X$ in column $N_L$ of subarray $A_1$) to row $K_U$.

   (c) There are at least two different kinds of Pauli operators.

      Let $k_1$ be the first row in the active region where column $N_L$ contains a Pauli operator, and $k_2$ be the first row in the active region where column $N_L$ contains a different Pauli operator.

      (i) Make row $k_1$ the top row of the active region by transposing, if necessary, row $k_1$ with row $K_U$.

      (ii) Make row $k_2$ the second row of the active region by transposing, if necessary, row $k_2$ with row $K_U + 1$.

      (iii) Multiply every other row in the active region with either row $K_U$, row $K_U + 1$, both rows or none, depending on the element in column $N_L$ (according to table 4).

      (iv) Let $P_1$ be the element in column $N_L$ on row $K_U$ and $P_2$ the one on row $K_U + 1$ (the column leaders). Turn $P_2$ into an $X$, as follows:

         • If neither $P_1$ nor $P_2$ is an $X$, multiply row $K_U$ to row $K_U + 1$, effectively turning $P_2$ into an $X$.

         • If $P_1$ is an $X$, transpose row $K_U$ with $K_U + 1$.

      (v) Multiply row $N_L$ (that is, the row containing an $X$ in column $N_L$ of subarray $A_1$) to row $K_U + 1$.

      (vi) Divide $T$ by 2, and increase $K_U$ by 1.

7. If the active region is still non-empty ($N_L \leq N_R$ and $K_U \leq K_R$), continue with step 4.

8. (End game) Here we calculate a correction factor $C$ for the overlap and the fidelity. Set $C = 1$ as default value. If $K_U \leq K_L$ do the following:

(a) Case $N_L \leq N$: consider the bottom right block of stabilizer array $A_2$ consisting of rows $K_U$ to $K_L$ and columns $N_R + 1$ to $N$. Calculate the rank $R_2$ of that block using, e.g. the RREF algorithm. From $R_2$ calculate the correction factor as $C = \frac{2N - R_1 - R_2}{2}$. 

(b) Case $N_L > N$: let $t_k$ be the generator phase of row $k$. If at least one of the $t_k$ for $K_U \leq k \leq K_L$ is $-1$, set $T = 0$.

9. Terminate with return values $F = CT/2^{N-K_1+N-K_2}$ for the overlap and $F_u = C\sqrt{T}/2^{N-K_1+N-K_2}$ for the Uhlmann fidelity.

6.1. Proof of correctness of algorithm OVERLAP

The overlap $F = \text{Tr}[\rho_1 \rho_2]$ can be calculated iteratively by performing the trace as a succession of partial traces over single qubits: $F = \text{Tr}_{\{1\}}[\text{Tr}_{\{1\}}[\rho_1 \rho_2]]$, where $\text{Tr}_{\{1\}}$ denotes the partial trace over the first qubit. What we need to show is that one iteration of steps 4–7 indeed performs this single-qubit partial trace. It will be convenient to express the overlap in terms of the projectors $P_1$ and $P_2$, with $\rho_1 = P_1/2^{N-K_1}$ and $\rho_2 = P_2/2^{N-K_2}$. Then

$$F = \frac{1}{2^{N-K_1+N-K_2}} \text{Tr}[P_1 P_2].$$

Keeping in mind that we also want to calculate $F_u$, we will proceed by first calculating $P_1 P_2 P_1$. The overlap is just the trace of this quantity, which is the same as $\text{Tr}[P_1 P_2]$ by virtue of $P_1$ being a projector.

Step 2 of the algorithm applies the same sequence of unitaries to both states, hence the overlap between them does not change (and neither does the Uhlmann fidelity). Let $P_1$ thus be specified by a CNF stabilizer array, containing $R_1 \leq N X$-operators:

$$P_1 = \bigotimes_{i=1}^{R_1} \frac{1}{2} \left( \mathbb{I} + s_i X \right) \otimes \mathbb{I}^{\otimes N-R_1}.$$

If all generators in array $A_1$ are independent, we obviously have $R_1 = K_1$. In the above expression, $s_i$ is the generator phase of the $i$th generator of $P_1$. Likewise, we will denote by $t_i$, the generator phase of the $i$th generator of $P_2$. Furthermore, let $P'_1$ be the stabilizer projector of the array obtained by deleting row 1 and column 1 from $A_1$.

In the following, we will calculate $P_1P_2P_1$ and show that it is equal to a certain scalar value $T$ times a tensor product of rank-1 projectors and identity operators. We will proceed in an iterative fashion, by showing that $P_1P_2P_1$ decomposes as a scalar $T_1$ times either a rank-1 projector or an identity tensored with a product $P'_1P'_2P'_1$ of projectors over qubits 2 to $N$. To calculate $T$, we start off with the initial value $T = 1$ and update $T$ by multiplying it with the value of $T_1$ found at each iteration.

We will assume first that $A_1$ and $A_2$ have more than 1 column. The case that they only have 1 column, which is what can happen in the final iteration of the algorithm, will be considered in subsection 6.1.4. We will also assume that the first tensor factor of $P_1$ is a rank-1 projector.
(i.e. $R_1 > 0$). The case that $P_1$ equals the identity (which will again typically happen at the end of the iterations) will also be covered in subsection 6.1.4

Let us now take on the main case, where there are at least two tensor factors to consider, and the first factor of $P_1$ is $\frac{I + sX}{2}$. Thus we can write $P_1 = \frac{I + sX}{2} \otimes P'_1$. As in algorithm CNF1 there are three cases to consider, depending on the number of different Pauli operators contained in the first column of the second array $A_2$. We will investigate these three possibilities in succession.

It is useful to note that $\frac{I + sX}{2} \otimes \frac{I + sX}{2} = \begin{cases} (I + sX)/2, & \sigma = I, \\ s(I + sX)/2, & \sigma = X, \\ 0, & \sigma = Y, Z. \end{cases}$

### 6.1.1. Case (a).

If the first column of $A_2$ contains no Pauli operators, this corresponds to $P_2$ being of the form

$$P_2 = I \otimes P'_2,$$

where $P'_2$ is the stabilizer projector of the array obtained by deleting column 1 from $A_2$. Hence

$$P_1P_2P_1 = \left( \frac{I + s_1X}{2} \otimes P'_1 \right) \left( I \otimes P'_2 \right) \left( \frac{I + s_1X}{2} \otimes P'_1 \right)$$

$$= \frac{I + s_1X}{2} \otimes P'_1 P'_2 P'_1.$$

This is indeed of the form claimed above, with scalar value $T_1 = 1$. Hence, nothing needs to be done in this iteration except for deleting column 1.

### 6.1.2. Case (b).

Steps (b.i) and (b.ii) bring column 1 of $P_2$ to RREF form. In this case, column 1 will contain a single Pauli operator, $\sigma$, in row 1. Denote the remaining operators on row 1 by $g'$. Let $P'_2$ be the stabilizer projector of the array obtained by deleting row 1 and column 1 from $A_2$. Thus $P_2$ is of the form

$$P_2 = \frac{I + t_1\sigma \otimes g'}{2} (I \otimes P'_2).$$

We then have

$$P_1P_2P_1 = \left( \frac{I + s_1X}{2} \otimes P'_1 \right) \frac{I + t_1\sigma \otimes g'}{2} (I \otimes P'_2) \left( \frac{I + s_1X}{2} \otimes P'_1 \right)$$

$$= \frac{I + s_1X}{2} \otimes P'_1 P'_2 P'_1 + \frac{t_1 I + s_1X}{2} \sigma \frac{I + s_1X}{2} \otimes P'_1 g' \otimes P'_2 P'_1.$$

We can therefore distinguish two cases. If $\sigma$ is not an $X$, we find

$$P_1P_2P_1 = \frac{1}{2} \frac{I + s_1X}{2} \otimes P'_1 P'_2 P'_1.$$

This corresponds to a value of $T_1 = 1/2$. This is implemented in step (b.iii, first case) by dividing the running $T$ by 2, and deleting row 1 and column 1 from $A_2$. 

If, on the other hand, \( \sigma = X \), we have
\[
\mathcal{P}_1 \mathcal{P}_2 \mathcal{P}_1 = \frac{1}{2} \left( \mathbb{1} + s_1 X \right) \otimes \mathcal{P}_1' \mathcal{P}_2' \mathcal{P}_1' + \frac{1}{2} \mathcal{P}_1' g' \mathcal{P}_2' \mathcal{P}_1' + \frac{1}{2} \mathcal{P}_1' \mathcal{P}_2' \mathcal{P}_1'.
\]
\[
= \frac{1}{2} \left( \mathbb{1} + s_1 X \right) \otimes \mathcal{P}_1' \mathcal{P}_2' \mathcal{P}_1' + \frac{1}{2} \mathcal{P}_1' \mathcal{P}_2' \mathcal{P}_1',
\]
where \( \mathcal{P}_2'' = [(\mathbb{1} + s_1 t_1 g')/2] \mathcal{P}_2' \) is a projector. This corresponds to a scalar value of \( T_1 = 1 \). This is accomplished in step (b.iii, second case) by multiplying row 1 of \( A_1 \) to row 1 of \( A_2 \), and deleting column 1 of \( A_2 \) (leaving row 1).

6.1.3. Case (c). Steps (c.i), (c.ii) and (c.iii) bring column 1 of \( A_2 \) in RREF form. Column 1 will contain two Pauli operators, \( \sigma_1 \) in row 1 and \( \sigma_2 \neq \sigma_1 \) in row 2. Step (c.iv) ensures, by suitable row multiplication or transposition, that \( \sigma_2 \) is an \( X \) operator, so \( \sigma_1 \) is not. Let the remaining operators on rows 1 and 2 be denoted by \( g_1 \) and \( g_2 \), respectively. Let \( \mathcal{P}_2' \) be the stabilizer projector of the array \( A_2 \), obtained by deleting rows 1 and 2 and column 1 from \( A_2 \). Then \( \mathcal{P}_2 \) is given by
\[
\mathcal{P}_2 = \frac{\mathbb{1} + t_1 \sigma_1 \otimes g_1' \mathbb{1} + t_2 X \otimes g_2' (\mathbb{1} \otimes \mathcal{P}_2')}{2}.
\]
Thus
\[
\mathcal{P}_1 \mathcal{P}_2 \mathcal{P}_1 = \left( \frac{\mathbb{1} + s_1 X}{2} \otimes \mathcal{P}_1' \right) \frac{\mathbb{1} + t_1 \sigma_1 \otimes g_1' \mathbb{1} + t_2 X \otimes g_2' (\mathbb{1} \otimes \mathcal{P}_2')}{2} \left( \frac{\mathbb{1} + s_1 X}{2} \otimes \mathcal{P}_1' \right)
\]
\[
= \frac{1}{4} \left[ \frac{\mathbb{1} + s_1 X}{2} \otimes \mathcal{P}_1' \mathcal{P}_2' \mathcal{P}_1' + \frac{\mathbb{1} + s_1 X}{2} \sigma_1 \frac{\mathbb{1} + s_1 X}{2} \otimes \mathcal{P}_1' g_1' \mathcal{P}_2' \mathcal{P}_1' 
\]
\[
+ \frac{t_2}{2} \frac{\mathbb{1} + s_1 X}{2} X \frac{\mathbb{1} + s_1 X}{2} \otimes \mathcal{P}_1' g_2' \mathcal{P}_2' \mathcal{P}_1' + \frac{\mathbb{1} + s_1 X}{2} \sigma_1 \frac{\mathbb{1} + s_1 X}{2} \otimes \mathcal{P}_1' g_1' g_2' \mathcal{P}_2' \mathcal{P}_1' \right],
\]
giving
\[
\mathcal{P}_1 \mathcal{P}_2 \mathcal{P}_1 = \frac{1}{2} \left( \frac{\mathbb{1} + s_1 X}{2} \otimes \mathcal{P}_1' \frac{\mathbb{1} + s_1 t_2 g_2'}{2} \mathcal{P}_2' \mathcal{P}_1' \right)
\]
\[
= \frac{1}{2} \left( \frac{\mathbb{1} + s_1 X}{2} \otimes \mathcal{P}_1' \mathcal{P}_2' \mathcal{P}_1' \right)
\]
with \( \mathcal{P}_2'' = [(\mathbb{1} + s_1 t_2 g_2')/2] \mathcal{P}_2' \) a projector. This corresponds to \( T_1 = 1/2 \). This is implemented in steps (c.v) and (c.vi) through multiplying row 2 in \( A_2 \) by row 1 of \( A_1 \), dividing \( T \) by 2, and subsequently deleting row 1 and column 1 in \( A_2 \).

6.1.4. End game. We still have to consider the situation where there is only one column left and the one where \( \mathcal{P}_1 \) is a tensor product of identity operators.
The first situation is when \( N_L = N \). In that case the symbols \( \mathcal{P}_1', g' \) and \( \mathcal{P}_2' \) used in the previous subsections are meaningless. However, we can still make sense out of the calculations if we replace these symbols formally by the scalar 1. Inspection of the relevant calculations then shows that at the very end of the algorithm, if \( N_L = N \), we have to check whether one of the remaining generator phases \( t_i \) is \(-1\), in which case the states under consideration are orthogonal. That means both the overlap and the Uhlmann fidelity are 0, which we impose by setting \( T = 0 \).

If \( N_L < N \) but \( \mathcal{P}_1 \) acts as the identity on columns \( N_L \) to \( N \), \( \mathcal{P}_1 \mathcal{P}_2 \mathcal{P}_1 \) reduces to \( \mathcal{P}_2 \). This can easily be decomposed as a tensor product by calculating its rank \( R_2 \) (the easiest way to do this is by using the RREF algorithm). Thus the remaining part on columns \( N_L \) to \( N \) of \( \mathcal{P}_1 \mathcal{P}_2 \mathcal{P}_1 = \mathcal{P}_2 \) is unitarily equivalent to

\[
\bigotimes_{i=1}^{R_2} \frac{\mathbb{1} + X}{2} \otimes \mathbb{1}^{\otimes N - R_1 - R_2}.
\]

6.1.5. Overlap and Uhlmann fidelity. In the previous subsection, we have shown that \( \mathcal{P}_1 \mathcal{P}_2 \mathcal{P}_1 \) is equal to \( T \) times a tensor product of \( R_1 + R_2 \) rank-1 projectors and \( N - R_1 - R_2 \) identity operators. Calculating the overlap and the Uhlmann fidelity is now easy. Assuming that \( R_1 = K_1 \), we have for the overlap

\[
F = \frac{1}{2^{N-K_1+N-K_2}} \text{Tr}[\mathcal{P}_1 \mathcal{P}_2 \mathcal{P}_1],
\]

\[
= \frac{1}{2^{N-K_1+N-K_2}} T 2^{N-R_1-R_2},
\]

\[
= 2^{-(N-K_2+R_2)} T,
\]

where it has to be noted that \( R_2 \leq K_2 \). Since \( T \) is also a negative power of 2, one sees that the overlap takes values of either 0 or \( 2^{-j} \), where \( j \) is an integer between 0 and \( N \).

Similarly, the Uhlmann fidelity between states \( \rho_1 \) and \( \rho_2 \) is given by

\[
F_u = \text{Tr} \left[ \sqrt{\mathcal{P}_1} \sqrt{\mathcal{P}_2} \sqrt{\mathcal{P}_1} \right].
\]

Again we substitute the stabilizer states for their appropriately scaled projectors. Noting that the square root of a projector is that same projector which gives

\[
F_u = (2^{N-K_1} 2^{N-K_2})^{-1/2} \text{Tr} \left[ \sqrt{\mathcal{P}_1} \mathcal{P}_2 \mathcal{P}_1 \right]
\]

\[
= (2^{N-K_1} 2^{N-K_2})^{-1/2} \sqrt{T} 2^{N-R_1-R_2}
\]

\[
= 2^{(K_2-K_1)/2-R_2} \sqrt{T}.
\]

7. Bipartite normal form

In this section, we will modify the single-party algorithm CNF1 so that it can be used to reduce a stabilizer array of a bipartite system to a certain normal form. This algorithm will allow us to deduce the exact structure of this normal form, which is the content of theorem 1. This theorem basically tells us that a bipartite mixed stabilizer state is locally equivalent to a tensor product.
of some number of pure EPR pairs and a separable mixed state. The main benefit of this normal form is that the entanglement of the state can immediately be read off from the normal form. Because of the theorem, it turns out that in order to calculate the state’s entanglement it is not necessary to actually compute the normal form completely. Instead, a simplified algorithm to calculate entanglement will be presented.

Let us start with the statement of the normal form.

**Theorem 1.** Consider a system of \( N \) qubits, separated into two parties, \( A \) and \( B \), containing \( N_A \) and \( N_B \) qubits, respectively. Consider a stabilizer state described by an array of \( K \) independent, commuting generators.

(i) By applying a suitable sequence of elementary row operations and local elementary column (qubit) operations, the stabilizer array can be brought into the following normal form:

\[
\begin{pmatrix}
\mathbb{1} & \cdots & \mathbb{1} & \mathbb{1} & \cdots & \mathbb{1} & \mathbb{1} & \cdots & \mathbb{1} \\
Z & \mathbb{1} & \cdots & \mathbb{1} & \mathbb{1} & \cdots & \mathbb{1} & \mathbb{1} & \cdots & \mathbb{1} \\
\mathbb{1} & X & \cdots & \mathbb{1} & \mathbb{1} & \cdots & \mathbb{1} & \mathbb{1} & \cdots & \mathbb{1} \\
\mathbb{1} & Z & \cdots & \mathbb{1} & \mathbb{1} & \cdots & \mathbb{1} & \mathbb{1} & \cdots & \mathbb{1} \\
\vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\
\mathbb{1} & \mathbb{1} & \cdots & X & \mathbb{1} & \cdots & \mathbb{1} & \mathbb{1} & \cdots & \mathbb{1} \\
\mathbb{1} & \mathbb{1} & \cdots & Z & \mathbb{1} & \cdots & \mathbb{1} & \mathbb{1} & \cdots & \mathbb{1} \\
\mathbb{1} & \mathbb{1} & \cdots & \mathbb{1} & \ast & \cdots & \ast & \mathbb{1} & \cdots & \mathbb{1} \\
\vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots & \vdots & \cdots & \vdots \\
\mathbb{1} & \mathbb{1} & \cdots & \mathbb{1} & \ast & \cdots & \ast & \mathbb{1} & \cdots & \mathbb{1} \\
\end{pmatrix},
\]  

(ii) Every pair of rows containing the \( XZ \) combinations corresponds to two qubits (one from each party) being in a pure maximally entangled EPR state and completely disentangled from the other qubits. The rows in the lower blocks of the normal form, containing only \( \mathbb{1} \) and \( X \) operators, correspond to the remaining qubits being in a (in general, mixed) separable state.

(iii) The stabilizer state described by the stabilizer array is locally equivalent to a tensor product of a certain number \( p \) of EPR pairs \( \Psi \) with a separable state. For any additive entanglement measure \( E \), the entanglement of the stabilizer state is \( pE(\Psi) \). An upper bound on \( p \) is given by

\[
p \leq \min\left(\left\lfloor \frac{K}{2} \right\rfloor, N_A, N_B\right).
\]

Equality is obtained if and only \( K = 2N_A = 2N_B \).

We will postpone the proof of part (i) of the theorem, the normal form, to the end of this section. The proof of part (iii), the entanglement properties of the normal form, is elementary and is left to the reader. The proof of part (ii) is presented next.

**Proof of part (ii).** For convenience of notation, we first permute the qubits in such a way that the pairs of columns having an \( XZ \) pair in the same rows are adjacent. By equation (5), the stabilizer state corresponding to the normal form of the theorem is

\[
\rho = \frac{1}{2^{N-K}} \prod_{k=1}^{2p} \frac{\mathbb{1} + g_k}{2} \prod_{l=2p+1}^{K} \frac{\mathbb{1} + g_l}{2}.
\]
From the specific form of the generators one sees that $\rho$ can be written as tensor product

$$\rho = \left(\frac{\mathbb{I} + X \otimes X}{2} \otimes p \otimes \rho'\right)^{\otimes p} \otimes \rho',$$

where $\rho'$ corresponds to the factor $\prod_{i=2, p+1}^{K} (\mathbb{I} + g_{i})/2$ containing $\mathbb{I}$ and $X$ operators only. It is a simple matter to verify that the factor $(\mathbb{I} + X \otimes X) (\mathbb{I} + Z \otimes Z)/4$ is identical to the EPR state $\Psi = |\psi\rangle\langle\psi|$, with $\psi = (1, 0, 0, 1)^{T}/\sqrt{2}$.

It is also simple to see that $\rho'$ is a separable state. As it only contains $\mathbb{I}$ and $X$ operators, it is diagonal in any basis where $X$ is diagonal, and it is well-known and easy to see that diagonal states are separable.

Algorithm CNFP for calculating the entanglement

We now present an algorithm to calculate the number of EPR pairs in the normal form, without actually reducing the stabilizer array completely to that normal form. This algorithm is almost identical to algorithm CNF1, the reduction algorithm for the single-party case.

To calculate the entanglement, the initial active region is set to comprise the block of elements pertaining to party A only, rather than the full stabilizer array, and algorithm CNFP (CNF for a single Party) is run on this active region.

Algorithm CNFP is identical to CNF1, apart from the following two differences:

Step (b.vi). While in CNF1 step (b.iii) is never undone by step (b.v) due to commutativity of the generators, this need no longer be the case here. Indeed, here we restrict attention to one party only, and the parts of the generators local to party A need not commute. Hence, step (b.v) might leave $Z$ operators in the leftmost column of the active region. We thus need a modification here: firstly, we must check whether this has happened and only if there are no $Z$ operators in this column may $K_{U}$ and $N_{L}$ be increased by 1. Otherwise, the extent of the active region must stay the same. The additional $Z$'s will then be treated in the next iteration of the algorithm.

Step (c.iv). In step (c.iv), the original algorithm looked for an anticommuting pair in the top two rows, the presence of which having been guaranteed by generator commutativity. Here, again, this is no longer true, because the pair might be located in party B, which we are not allowed to touch here. We, therefore, need a second modification, to deal with the case that there is no such anticommuting pair. In that case, instead of steps (c.v)–(c.vii), the following operations must be executed. Recall that the first column has an $XZ$ pair in its first two rows. This pair can now be used to eliminate all other Pauli operators in both the first two rows (by suitable single-qubit operations and CNOTs) and in the first column (by suitable row multiplications). Tables 5 and 6 contain the details. After that, the $XZ$ pair can be split off from the active region to form part of the normal form, by increasing $K_{U}$ by 2 and $N_{L}$ by 1.

Algorithm CNFP brings only that part of the stabilizer array in normal form that belongs to party A. Nevertheless, this is enough to read off the number of EPR pairs in the full reduction. This will be proven below. The number of EPR pairs $p$ is simply given by the number of $XZ$ pairs in the normal form of party A.

Proof of part (i). By suitable modification of the proof of algorithm CNF1, it can be shown that algorithm CNFP brings that part of the stabilizer array belonging to party A to the form as shown in (8), the columns left of the double vertical line.
Table 5. Required operations to eliminate all Pauli operators from column 2 of the stabilizer array shown above, in the various cases encountered.

<table>
<thead>
<tr>
<th>Column 2 Content</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Z</td>
</tr>
<tr>
<td>Y</td>
<td>Z</td>
</tr>
<tr>
<td>Z</td>
<td>Z</td>
</tr>
</tbody>
</table>

Table 6. Required operations to eliminate any Pauli operator from row 3 of the stabilizer array shown above.

<table>
<thead>
<tr>
<th>Row 3 Content</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Multiply row 1 with row 3.</td>
</tr>
<tr>
<td>Y</td>
<td>Multiply row 1 with row 3 and then row 2 with row 3.</td>
</tr>
<tr>
<td>Z</td>
<td>Multiply row 2 with row 3.</td>
</tr>
</tbody>
</table>

We next show that by further applying suitable column operations on the columns of party B, the complete normal form of (8) can be obtained.

Consider the first $XZ$ pair in party A. By commutativity of the generators, there must at least be 1 anticommuting pair on the same rows in party B. By a column permutation and a suitable single-qubit rotation, this anticommuting pair can be moved to the first column of party B and be brought in $XZ$ form. Using suitable CNOTs (see table 5) the operators right of the $XZ$ pair can all be brought to an $\mathbb{I}$ operator. Again by commutativity, the operators below the $XZ$ pair must then automatically be all $\mathbb{I}$ operators. Indeed, if a row (below the second) contained a Pauli operator in the first column of party B, it would not commute with either the first row, the second, or both.

One can proceed in a similar fashion with the second of party A’s $XZ$ pairs and party B’s second column and third and fourth row, and so forth until all of A’s $XZ$ pairs have been treated in this way.
What remains then are the rows below the horizontal line in (8). To show that the lower right block of party $B$ in (8) can be brought to the form as advertized (i.e. containing only $X$ and $\mathbb{I}$ operators, as denoted by the asterisks), we note that party $A$ contains no anticommuting pairs in those rows. Hence, the subarray consisting of party $B$’s lower right block (restricted to that block’s columns) consists of mutually commuting generators. By applying algorithm CNF1 to that subarray it can be brought in single-party normal form, consisting of $X$ and $\mathbb{I}$ operators only. Evidently, the row operations performed by the CNF1 algorithm (row permutation and multiplication) will also affect the corresponding rows in party $A$. However, as party $A$ has only $X$ and $\mathbb{I}$ operators in those rows, no $Y$ or $Z$ operators will be introduced and the end result will also contain only $X$ and $\mathbb{I}$ operators. 

\[ \square \]

8. Conclusions

The stabilizer formalism is a convenient tool for the study of entanglement properties of large quantum many-body systems. While the stabilizer formalism provides an efficient description of the quantum state in terms of eigenvalue equations, it is not immediately obvious how to obtain physical properties directly from these eigenvalue equations without explicitly having to write out the corresponding quantum state. In this paper, we have presented, employing elementary tools, a number of normal forms for pure and mixed stabilizer states. We have furthermore provided explicit, detailed descriptions of algorithms, whose convergence we have proven, that allow the generation of these normal forms. Using these normal forms, we can compute any entanglement measure, overlaps between stabilizer states and various other quantities. Detailed descriptions of the algorithms are provided that should make it straightforward to implement them in any programming language and we are able to provide MatLab suite of programs on request.

These algorithms provide a firm basis for the exploration entanglement properties of stabilizer states and suitable generalizations in a great variety of contexts. For example, it is readily seen that our approach is suitable for the efficient simulation of systems where the initial state is a linear combination of a polynomial number of stabilizer states. This and other applications will be explored in forthcoming publications.

Acknowledgments

We thank H J Briegel, O Dahlsten and J Oppenheim for discussions. This work is part of the QIP-IRC (www.qipirc.org) supported by EPSRC (GR/S82176/0) and is also supported by the EU Thematic Network QUPRODIS (IST-2001-38877) and the Leverhulme Trust grant F/07 058/U. Finally, we thank the anonymous referees for comments that substantially improved the presentation and also for pointing out a serious omission in our first treatment of overlap and Uhlmann fidelity.

References

[12] Schlingemann D 2004 Quant. Inf. Comp. 4 289
[22] Schlingemann D 2003 Quant. Inf. Comp. 3 431
[23] After completion of the scientific work reported in this paper, we became aware of Fattal D, Cubitt T S, Yamamoto Y, Bravyi S and Chuang I L 2004 Preprint quant-ph/0406168