When a quantum measurement can be implemented locally, and when it cannot

Scott M. Cohen[*]

Department of Physics, Duquesne University, Pittsburgh, Pennsylvania 15282, USA and
Department of Physics, Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213, USA

(Received 16 December 2009; revised manuscript received 28 September 2011; published 21 November 2011)

In the absence of quantum channels, local operations on subsystems and classical communication between parties (LOCC) constitute the most general protocols available on spatially separated quantum systems. Every LOCC protocol implements a separable quantum measurement, but it is known that there exist separable measurements that cannot be implemented by LOCC. A longstanding problem in quantum information theory is to understand the difference between LOCC and the full set of separable measurements. Toward this end, we show in this paper how to construct an LOCC protocol to implement an arbitrary separable measurement whenever such a protocol exists. In addition, given a measurement that cannot be implemented by LOCC within some fixed maximum number of rounds, the method shows explicitly that this is the case.

DOI: 10.1103/PhysRevA.84.052322 PACS number(s): 03.67.Ac

I. INTRODUCTION

Left to their own devices, quantum systems undergo unitary evolution. They may interact with other quantum systems, but considering all these systems together as a single entity, its state at any given time is related to that at any other time by a unitary transformation. We as scientists, however, often wish to know something about the systems we are studying, so we perform measurements to extract information with the aim of understanding the behavior of these systems. In order to draw conclusions, we must understand the measurements that we make, and perhaps more importantly, we will want to optimize our measurements to maximize the information we can extract given the constraints with which we either choose, or are forced by circumstances, to work. Of course, if we find a measurement that is optimal for one set of circumstances, it may well be that this measurement cannot be performed under other constraints. It is therefore crucial that we have a way to determine when a measurement is possible and when it is not.

If the system under consideration resides in a single laboratory, then it is purely an experimental question whether or not a given measurement is possible: do we have the tools and skills, or don't we? If, on the other hand, the system consists of two or more spatially separated subsystems, then it will often be the case that a given measurement simply cannot, *in principle*, be implemented. This question of the "local implementation" of a measurement is of fundamental interest for our understanding of quantum theory itself, and it also arises naturally in numerous applications considered in the quantum information sciences. Examples of such applications include distributed quantum computing [1], one-way quantum computing [2], entanglement distillation [3] and manipulation [4], local distinguishability of quantum states [5], local cloning [6], and various quantum cryptographic protocols, such as secret sharing [7].

It is not too difficult to describe in words the most general protocol possible for implementing a local measurement. Let us assume there are two subsystems, one (denote it as A) located in Alice's laboratory and the other (B) in Bob's. One of the parties, say Alice, starts by locally (on A) performing a generalized measurement [8] with outcomes corresponding to Kraus operators A_{i_1}. If the initial state of the system was $|\Psi_0\rangle$ and Alice obtained outcome i_1, the state will now be $(A_{i_1} \otimes I_B)|\Psi_0\rangle$, which is generally no longer normalized, and I_B (I_A) is the identity operator on system B (A). Alice calls Bob on the telephone and informs him her outcome was i_1, after which he performs a measurement on B, conditioned on Alice's outcome i_1 and described by Kraus operators $B_{i_2}^{(i_1)}$. He then informs Alice that his outcome was i_2, after which she performs a measurement with outcomes $A_{i_3}^{(i_1,i_2)}$, and they may continue in this way for an arbitrary number of rounds. From the fact that the probabilities of outcomes obtained at each stage must always sum to unity, one has that for each and every n,

$$\begin{aligned} I_A &= \sum_{i_n} A_{i_n}^{(\mathcal{S}_n)\dagger} A_{i_n}^{(\mathcal{S}_n)}, \\ I_B &= \sum_{i_n} B_{i_n}^{(\mathcal{S}_n)\dagger} B_{i_n}^{(\mathcal{S}_n)}, \end{aligned} \tag{1}$$

where $\mathcal{S}_n$ is a collection of indices $\{i_1,i_2,\ldots,i_{n-1}\}$ indicating all outcomes obtained in earlier measurements. The final (unnormalized) state of the system is given in terms of these Kraus operators as

$$\begin{aligned} |\Psi_f\rangle &= \big[\big(\cdots A_{i_3}^{(i_1,i_2)} A_{i_1}\big) \otimes \big(\cdots B_{i_4}^{(i_1,i_2,i_3)} B_{i_2}^{(i_1)}\big)\big]|\Psi_0\rangle \\ &= [\,\widehat{A}^{(\widehat{\mathcal{S}})} \otimes \widehat{B}^{(\widehat{\mathcal{S}})}]|\Psi_0\rangle, \end{aligned} \tag{2}$$

and $\widehat{\mathcal{S}}$ denotes the full set of local outcomes (leading to this particular overall final outcome) in the entire process just described.

This process, known as local operations and classical communication (LOCC), is quite complicated and difficult to analyze in detail. However, we see that the final outcomes are always in terms of product Kraus operators, $\widehat{A}^{(\widehat{\mathcal{S}})} \otimes \widehat{B}^{(\widehat{\mathcal{S}})}$, so we have what is known as a separable measurement [9]. Then, if one chooses to focus only on the final outcomes and not on how one actually gets to those outcomes by local measurements, the description is greatly simplified and more easily understood. For this reason, it is quite common to study separable measurements in the hope of gaining

[*]cohensm@duq.edu

a better understanding of LOCC [6,10–12]. It is known, however, that there exist separable measurements that cannot be implemented by LOCC [13–16].

Therefore before we can truly understand LOCC, we will need to know more about the difference between LOCC and the full set of separable measurements, something about which very little has been known up to the present time. In this paper, we provide an important step toward this goal by showing how to construct an LOCC protocol from an arbitrary separable measurement whenever this is possible. To be precise, by a *separable measurement* we will mean a fixed collection $\{\widehat{A}_j \otimes \widehat{B}_j\}$ of distinct product Kraus operators for which there exists a set of positive coefficients $\{\widehat{r}_j\}$ [17] such that

$$I_A \otimes I_B = \sum_j \widehat{r}_j \widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j, \tag{3}$$

where $\widehat{\mathcal{A}}_j = \widehat{A}_j^\dagger \widehat{A}_j$ and $\widehat{\mathcal{B}}_j = \widehat{B}_j^\dagger \widehat{B}_j$ (this definition of a measurement should not be confused with a separable operation, which is more general [18]). We emphasize that our definition of a measurement is in terms of the set of Kraus operators and not just the positive operators $\{\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j\}$ and that there may be more than one set of coefficients $\widehat{r}_j$ such that (3) is satisfied. Our goal in this paper is then to determine whether or not there exists an LOCC protocol for any one such set of coefficients.

In the next section, we describe a construction that accomplishes this goal and then provide a detailed algorithm for this construction. In Sec. III, several examples are discussed with the aim of giving the reader a better understanding of how the construction works. Then, in Sec. IV, a summary of the results is given. In Appendix A, we give a proof that the construction does what we have claimed it does, using two important lemmas, proved in Appendixes B and C. Appendix D discusses the complexity of the construction.

II. MAIN RESULT

Our main result is stated below.

Main theorem. Suppose Alice and Bob have a separable measurement they wish to perform. Assuming they restrict themselves to some maximum number of rounds, then the construction described below will allow them to determine whether or not the measurement they have designed can be locally implemented and, if it can, will provide them with the LOCC protocol that does so. Note that this claim applies to an extremely general situation, as it does not matter what task is accomplished by the given measurement.

In the next section, we describe the construction and explain why it accomplishes what we have just claimed. A detailed algorithm is presented in the subsequent section, and a proof of the main theorem can be found in Appendix A.

A. The construction

First note that *any LOCC protocol can be represented as a tree* consisting of nodes into each of which a single branch enters from another node on its left. Time progresses to the right, and each node at "level" n is associated with a local Kraus operator, $A_{i_n}^{(\mathcal{S}_n)}$ or $B_{i_n}^{(\mathcal{S}_n)}$. The subset of nodes at level n that are all attached via a branch to the same node on their left corresponds to a complete measurement, with the associated Kraus operators satisfying one or the other of Eqs. (1). Each node may be associated with the local Kraus operator that is performed at that point in the protocol or equally well with the ordered product of local Kraus operators that have been performed by that party up to that point. We will find it useful, however, to instead associate with each node the positive operator formed from the latter product by multiplying it on its left by its Hermitian conjugate. That is, with each (A-)node $\mathcal{S}_{n+1} = \{\mathcal{S}_n, i_n\}$, we will associate the operator

$$\mathcal{A}_{i_n}^{(\mathcal{S}_n)} = A_{i_1}^\dagger A_{i_3}^{(i_1,i_2)\dagger} \cdots A_{i_{n-2}}^{(\mathcal{S}_{n-2})\dagger} A_{i_n}^{(\mathcal{S}_n)\dagger} A_{i_n}^{(\mathcal{S}_n)} A_{i_{n-2}}^{(\mathcal{S}_{n-2})} \cdots A_{i_3}^{(i_1,i_2)} A_{i_1}. \tag{4}$$

Then, from (1), we have

$$\sum_{i_n} \mathcal{A}_{i_n}^{(\mathcal{S}_n)} = \mathcal{A}_{i_{n-2}}^{(\mathcal{S}_{n-2})}. \tag{5}$$

As simple as this equation is to obtain, it is nonetheless extremely powerful, as it *must be satisfied at each and every node in an LOCC tree*. What this means is that if we know later parts of an LOCC protocol, we can construct the earlier parts that lead to those later ones. In particular, if we know the final outcomes, we can work backward to attempt to construct a full LOCC protocol. If starting from those final outcomes we can build every tree that is compatible with (5), then we can check whether or not one of those trees corresponds to a complete LOCC protocol.

Thus, if we sum the positive operators $\mathcal{A}_{i_n}^{(\mathcal{S}_n)}$ associated with the collection of nodes emerging on the right directly from node $\mathcal{B}_{i_{n-1}}^{(\mathcal{S}_{n-1})}$, we obtain the positive operator associated with the unique node $\mathcal{A}_{i_{n-2}}^{(\mathcal{S}_{n-2})}$ from which that $\mathcal{B}_{i_{n-1}}^{(\mathcal{S}_{n-1})}$ node emerges. Furthermore, we have the very important observation that this sum is independent of the index i_{n-1}. This will serve as a useful constraint as there can be many such B nodes emerging from the node $\mathcal{A}_{i_{n-2}}^{(\mathcal{S}_{n-2})}$, and the sums in (5) for all of these B nodes must be the same,

$$\sum_{i_n} \mathcal{A}_{i_n}^{(\mathcal{S}_n)} = \sum_{i_n'} \mathcal{A}_{i_n'}^{(\mathcal{S}_n')}, \tag{6}$$

whenever $\mathcal{S}_n$ and $\mathcal{S}_n'$ differ only in their last entry, the index i_{n-1}; see Fig. 1. Obviously, there are analogous sums that must be satisfied by the $\mathcal{B}_{i_n}^{(\mathcal{S}_n)}$.

These ideas will now be used to construct a complete LOCC protocol whenever this is possible. Consider first the simplest case where each $\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j$ appears once and only once in the final set of outcomes of the protocol (the leaves of the tree), the entire collection satisfying (3) with $\widehat{r}_j = 1$. As illustrated in Fig. 2, start with two-node trees having operators $\widehat{\mathcal{A}}_j$ on the right and $\widehat{\mathcal{B}}_j$ on the left. Find all maximal subsets of $\widehat{\mathcal{B}}_j$ that are equal to each other and merge the corresponding nodes into a single node with multiple branches emerging to the corresponding A nodes. Attach a new A node to the left of each individual (merged) B node, and label these new nodes by the positive operator that is the sum of the $\widehat{\mathcal{A}}_j$ that emerge from the given B node (even if there is only a single term in that sum), as shown at the right in Fig. 2 , which will ensure that (5) is always satisfied. By merging multiple nodes into single ones only when all those nodes are equal to each other, we will also ensure that the equality is satisfied in (6).

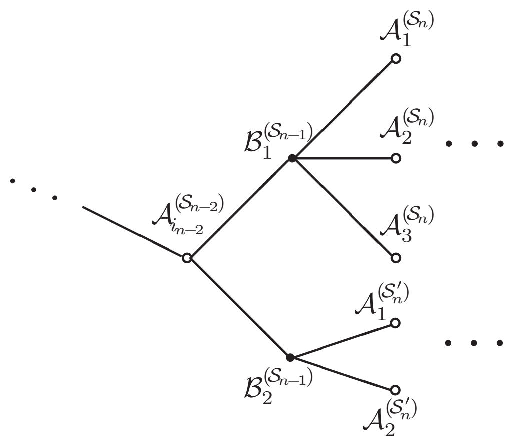

$\mathcal{A}_1^{(\mathcal{S}_n)}$
$\mathcal{A}_2^{(\mathcal{S}_n)}$
$\mathcal{A}_3^{(\mathcal{S}_n)}$
$\mathcal{B}_1^{(\mathcal{S}_{n-1})}$
$\mathcal{A}_{i_{n-2}}^{(\mathcal{S}_{n-2})}$
$\mathcal{A}_1^{(\mathcal{S}'_n)}$
$\mathcal{A}_2^{(\mathcal{S}'_n)}$
$\mathcal{B}_2^{(\mathcal{S}_{n-1})}$

FIG. 1. Illustration of Eq. (6). The sums $\mathcal{A}_1^{(\mathcal{S}_n)}+\mathcal{A}_2^{(\mathcal{S}_n)}+\mathcal{A}_3^{(\mathcal{S}_n)}$ and $\mathcal{A}_1^{(\mathcal{S}'_n)}+\mathcal{A}_2^{(\mathcal{S}'_n)}$ must be equal to each other and also equal to $\mathcal{A}_{i_{n-2}}^{(\mathcal{S}_{n-2})}$.

This procedure is then iterated: Consider the newest nodes that have just been created at the previous stage, merge subsets of these whose labels are all equal to each other, attach new nodes to the left of these merged nodes, and label each of these newest nodes with a sum of the positive operators that label the nodes that emerge from the corresponding node that was just merged. By induction, all labels will be sums of $\widehat{\mathcal{A}}_j$ or $\widehat{\mathcal{B}}_j$ [19].

If at some stage all these (sub)trees merge into a single connected tree, then an LOCC protocol has been identified (readers may find it helpful to study the examples given in Sec. III).

If the tree does not close, this attempt has failed, but one cannot yet conclude that no LOCC protocol exists for this set of final outcomes. This is because we must first exhaust all possible ways of merging the nodes. For example, if there are three $\widehat{\mathcal{B}}_j$ that are equal to each other, $\widehat{\mathcal{B}}_1=\widehat{\mathcal{B}}_2=\widehat{\mathcal{B}}_3$, then instead of merging all three into a single node, it may work better to merge only $\widehat{\mathcal{B}}_1$ and $\widehat{\mathcal{B}}_2$, keeping $\widehat{\mathcal{B}}_3$ aside for later use (see Example 4 of Sec. III).

Of course, Alice and Bob may also be able to accomplish their task with a protocol that ends with multiple appearances of each of the $\widehat{\mathcal{A}}_j\otimes\widehat{\mathcal{B}}_j$; allowing such replications introduces additional possible ways of constructing trees. In this case, we must multiply each copy (indexed by k) by a positive real factor $\widehat{r}_{jk}$ to ensure completeness of the measurement [see (3)]. However, since we will be using (6) to compare sums of $\widehat{\mathcal{A}}_j$'s and $\widehat{\mathcal{B}}_j$'s separately, it will be useful to instead write $\widehat{q}_{jk}\widehat{\mathcal{A}}_j\otimes\widehat{p}_{jk}\widehat{\mathcal{B}}_j$ on the final nodes (see Example 5 in Sec. III for more details of why this is useful). That is, by following the above procedure, all B nodes will be labeled by sums of $\widehat{p}_{jk}\widehat{\mathcal{B}}_j$, and two such nodes can be merged when

$$\sum_{j\in\mathcal{J}}\sum_k \widehat{p}_{jk}\widehat{\mathcal{B}}_j=\sum_{j'\in\mathcal{J}'}\sum_{k'}\widehat{p}_{j'k'}\widehat{\mathcal{B}}_{j'}, \tag{7}$$

where $\mathcal{J}$ and $\mathcal{J}'$ indicate which $\widehat{\mathcal{B}}_j$ appear in the sums labeling the two separate nodes in question and the k (k') sum is determined by which copies of each $\widehat{\mathcal{B}}_j$ are present.

If the positive quantities $\widehat{q}_{jk},\widehat{p}_{jk}$ were known, we could simply follow the previous procedure. However, since (a) these factors are unknown and (b) we do not know how many copies of each $\widehat{\mathcal{A}}_j\otimes\widehat{\mathcal{B}}_j$ to start with, we are presented with a challenge. Nonetheless, by using the $\widehat{q}_{jk},\widehat{p}_{jk}$ as free variables to be determined later by constraints of the form (7), we will see how to construct an LOCC tree whenever possible and to thereby determine whether or not an LOCC protocol exists.

Any positive operator may be thought of as a vector pointing into the "positive orthant," and positive linear combinations of a set of these vectors lie in the (convex) cone generated by the set. Then, a relationship such as (7) represents a (nontrivial [20]) intersection of the convex cone generated by the $\widehat{\mathcal{B}}_j$ for $j\in\mathcal{J}$ with that for $j\in\mathcal{J}'$. Any two or more nodes can be identified as satisfying (7) by looking for a common intersection of their cones [21], and only if their cones intersect can the two nodes be merged. We thus have a way to identify all sets of nodes that can possibly be merged.

One way to construct all possible trees [22] is as follows: imagine a toolbox initially filled with "tools" that are two-node trees, $\widehat{q}_{j1}\widehat{\mathcal{A}}_j\otimes\widehat{p}_{j1}\widehat{\mathcal{B}}_j$, as represented at the left in Fig. 2. Identify all intersections of the cones generated by vectors $\widehat{\mathcal{B}}_j$; these cones are one-dimensional at this first step. For each intersection, follow the procedure described above for merging nodes (any given tool can be used an arbitrary number of times), adding a new node to the left, and labeling this new node by sums of the associated $\widehat{q}_{jk}\widehat{\mathcal{A}}_j$ (a different k for each use of $\widehat{\mathcal{A}}_j\otimes\widehat{\mathcal{B}}_j$). Add each of these new trees to the toolbox, while keeping all trees that were already there. For multiple cones that share a common intersection, create all possible new trees, in the same way as was described above for the case that $\widehat{\mathcal{B}}_1=\widehat{\mathcal{B}}_2=\widehat{\mathcal{B}}_3$: one tree for every mutually intersecting subset. In addition, to each new tree in the toolbox, associate a set of constraints of the form (7), which will be needed at the end to determine the values of $\widehat{p}_{jk}$ and $\widehat{q}_{jk}$. Once this step is completed, we proceed to the next step by looking at all trees presently in the toolbox to identify all intersections of convex cones associated with the leftmost (now A) nodes of each of these trees, merge nodes in all possible allowed ways to create new trees, add these to the toolbox, and combine the sets of constraints associated with the merged trees along with the new constraints from the newly merged nodes to obtain a larger set of constraints.

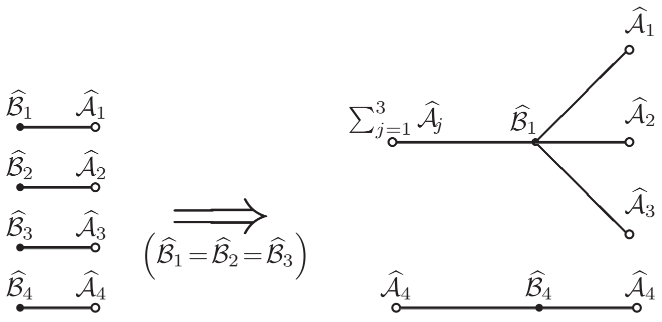

$\widehat{\mathcal{B}}_1$
$\widehat{\mathcal{A}}_1$
$\widehat{\mathcal{B}}_2$
$\widehat{\mathcal{A}}_2$
$\widehat{\mathcal{B}}_3$
$\widehat{\mathcal{A}}_3$
$\widehat{\mathcal{B}}_4$
$\widehat{\mathcal{A}}_4$
$\Longrightarrow$
$(\widehat{\mathcal{B}}_1=\widehat{\mathcal{B}}_2=\widehat{\mathcal{B}}_3)$
$\sum_{j=1}^3\widehat{\mathcal{A}}_j$
$\widehat{\mathcal{B}}_1$
$\widehat{\mathcal{A}}_1$
$\widehat{\mathcal{A}}_2$
$\widehat{\mathcal{A}}_3$
$\widehat{\mathcal{A}}_4$
$\widehat{\mathcal{B}}_4$
$\widehat{\mathcal{A}}_4$

FIG. 2. Method of construction of an LOCC tree from a set of product operators $\widehat{\mathcal{A}}_j\otimes\widehat{\mathcal{B}}_j$. When $\widehat{\mathcal{B}}_1=\widehat{\mathcal{B}}_2=\widehat{\mathcal{B}}_3$, the three B nodes corresponding to these operators can be merged into a single node, after which we attach a new node to its left, labeled by the sum $\widehat{\mathcal{A}}_1+\widehat{\mathcal{A}}_2+\widehat{\mathcal{A}}_3$.

This procedure is then iterated, creating *all* [22] *possible (connected) trees at each step* that are consistent with (5) and (6). Continue until one of these trees includes each of the $\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j$ at least once (we will refer to these as "complete" trees), there is nothing further one can do, or one has already used the maximum allowed number of rounds (it is necessary to impose a maximum number of rounds because otherwise it appears possible that the procedure could continue indefinitely even when no LOCC exists). If a complete tree is found, we must consider the full set of (linear) constraints that has been associated with that tree, checking that we can find a set of all the $\widehat{q}_{jk},\widehat{p}_{jk} \geqslant 0$ that are consistent with these constraints. Note that since in the actual protocol Alice and Bob begin by having done nothing, represented by operators I_A or I_B, the tree must close to a "double root," with each root labeled by one or the other identity operator, giving two additional constraints. If such a solution can be found, then an LOCC protocol exists. If not, continue the construction to see if another complete tree can be found. Once one has run out of rounds or come to a point that nothing further can be done, then, since we know our construction produces a closed tree whenever an LOCC protocol exists, we may conclude that no LOCC protocol exists for this separable measurement in the given maximum number of rounds. For a more detailed proof of this statement, see Appendix A.

B. Detailed algorithm

We now give a detailed step-by-step algorithm for the construction just described. At various steps along the way in this algorithm, we create multiple copies of certain trees that had previously been constructed. The point of this is to be sure we merge these trees to other ones in all ways it is possible for that tree to be merged. That is, one copy of a given tree is created for each possible way that tree can be merged to other trees, with a *different* copy of that particular tree being used for each different merging.

(1) Start with one two-node tree for each of the $\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j, j = 1,\ldots,N_0$, each with its B node on the left (the choice of B nodes rather than A nodes is arbitrary, as will become clear below), and include positive factors $\widehat{q}_{j1}(\widehat{p}_{j1})$ with each $\widehat{\mathcal{A}}_j(\widehat{\mathcal{B}}_j)$; partition all leftmost nodes into equivalence classes, within each of which all nodes are proportional to each other. Set $l = 0$, which will serve as a counter for the depth of the trees (this depth is equal to $l + 2$).

(2) WHILE $l < L$

(3) Increment l. For each equivalence class and for each subset in that class that contains at least one member, (a) CREATE one copy of each tree in the given subset; (b) MERGE the leftmost nodes of all trees within the subset into a single node and relabel the coefficients $\widehat{q}_{jk},\widehat{p}_{jk}$ with a unique value of the k index for every different appearance of $\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j$ at the leaves within the combined tree, adjusting those k indices throughout the rest of the tree to be consistent [according to (5)] with its leaf labels; (c) EXTEND the tree by attaching a new node to the left, and label that new node to obey (5) (note that this must also be done for each one-tree subset, as is illustrated for $\widehat{\mathcal{A}}_4 \otimes \widehat{\mathcal{B}}_4$ at the bottom right of Fig. 2, which is why it does not matter that we chose to start with all the B nodes on the left since we will next do the same thing with all the A nodes on the left); (d) RECORD all constraints that the merged nodes must be equal (at the first step, these will be of the form $\widehat{p}_{ik}\widehat{\mathcal{B}}_i = \widehat{p}_{jk'}\widehat{\mathcal{B}}_j$ when node $\widehat{\mathcal{B}}_i$ is merged with node $\widehat{\mathcal{B}}_j$ and are constraints on the $\widehat{p}$'s). Number the *new* trees sequentially from $N_{l-1} + 1$ to N_l, where N_l is the total number of trees at this stage (including trees of all depths constructed so far), which does not exceed $2^{N_{l-1}} - 1$. (Note that we can ignore each equivalence class that is identical to one that was previously present at an earlier pass through this algorithm, as all trees that can be constructed from that equivalence class have already been constructed.)

(4) FOR $m = N_{l-1} + 1$ to N_l

(5) IF the mth tree includes each of the $\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j$ at least once, then for the collection of all constraints recorded in (multiple passes through) step 3 for the m^{th} tree, check to see if there exists a solution for $\widehat{p}_{jk},\widehat{q}_{jk}$, including constraints that the two leftmost nodes in the tree under consideration are labeled by I_A and I_B (we start this FOR loop at $N_{l-1} + 1$ since all the earlier trees have already been examined; for $l = 1$, we know that the first N_0 of the trees have only a single $\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j$ and so cannot include each of them at least once). If such a solution exists, we are done, having identified an LOCC protocol, so exit and END this algorithm.

(6) END FOR (m)

(7) We now have an expanded set of trees all having the same "type" (A or B) of nodes on the left, labeled by sums of $\widehat{q}_{jk}\widehat{\mathcal{A}}_j$ or $\widehat{p}_{jk}\widehat{\mathcal{B}}_j$. Consider the convex cones generated by the sets $\{\widehat{\mathcal{A}}_j\}$ or $\{\widehat{\mathcal{B}}_j\}$ appearing in each such sum for the leftmost nodes, and identify an equivalence class for each subset of these nodes such that the associated convex cones are mutually intersecting (a given tree will generally be included in multiple equivalence classes). Each such intersection implies that the associated trees can be merged, which we will do next, so go back to step 2 and repeat. However, we only need to look for *new* equivalence classes, involving newly constructed trees (along with all previous ones), since we have already constructed all trees that derive from the other equivalence classes. If there are no new classes, then no new trees can be constructed, which means since no previously constructed tree has been found to be LOCC, then no LOCC protocol exists for this measurement, no matter how many rounds are allowed. Therefore, exit and END this algorithm.

(8) END WHILE (l)

Note that as we loop through the WHILE loop, we keep *all* trees for the next round, including not just those constructed in the present round but also those from all previous rounds (we also keep all constraints). This makes it possible for multiple trees of differing size and structure, such that several of them each include the same $\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j$ (or even several $\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j$ that are repeated in this way), to be merged together into a single tree. Example 4 of the next section illustrates in a detailed way how the algorithm works.

III. EXAMPLES

Here we provide a few illustrative examples to make more concrete our method of constructing an LOCC protocol from a set of product operators, as presented in the previous section.

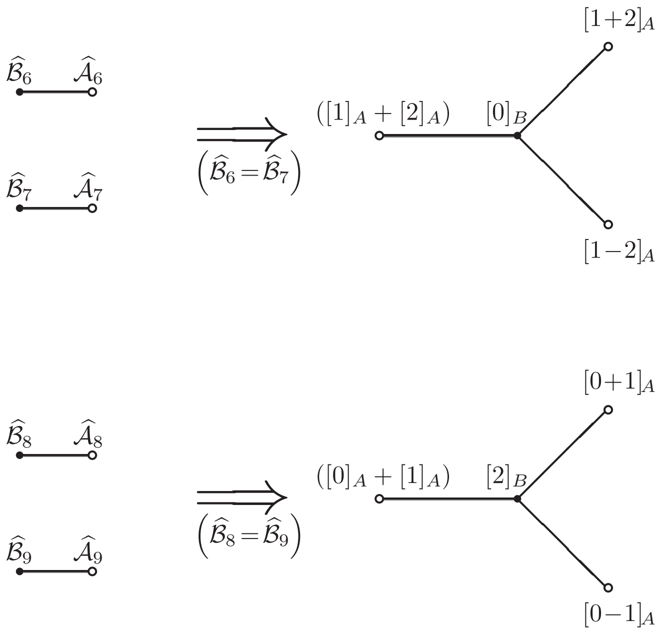
Example 1. Let us begin with a simple example, for which it will be easy to see that no LOCC protocol exists. Suppose we have a separable measurement with a corresponding set of positive operators, $\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j$, for which no two of the $\widehat{\mathcal{A}}_j$ (and no two of the $\widehat{\mathcal{B}}_j$) are equal to each other (technically, we should require that no two are proportional to each other, as we can always remove a positive factor from an $\widehat{\mathcal{A}}_j$ and place it on the $\widehat{\mathcal{B}}_j$ without changing $\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j$, but we can assume that the $\widehat{\mathcal{A}}_j$ are all normalized). Since no two of the $\widehat{\mathcal{A}}_j$ are equal, there is no possibility that any two (or more) of the $\widehat{\mathcal{B}}_j$ can emerge from the same A node and vice versa. Hence, it is not possible to merge any of the original two-node trees that we start out with and which represent the outcomes of our desired measurement, implying that there is no way to even begin to construct an LOCC tree.

Example 2. Here is another simple example, this time one for which an LOCC does exist, where Alice and Bob each do complete projective measurements on their local systems. Write $\widehat{\mathcal{A}}_{j_1} \otimes \widehat{\mathcal{B}}_{j_2} = [j_1]_A[j_2]_B$, where $[j_1]_A = |j_1\rangle_A\langle j_1|$ and similarly for $[j_2]_B$. For each projector $[j_2]_B, (j_2 = 0, \ldots, d_B - 1)$, onto one of Bob's standard basis states, j_1 runs through all values $0, \ldots, d_A - 1$, so $\{\widehat{\mathcal{A}}_{j_1} \otimes \widehat{\mathcal{B}}_{j_2}\}$ is a set of projectors onto a complete basis of Alice and Bob's full Hilbert space. Then, at the first step in our construction and for each value of j_1, connect the A nodes of all the two-node trees that are labeled by this $[j_1]_A$. Then, the B nodes that emerge to the right of each of these merged A nodes sum to the identity operator. The new B nodes that we attach to the left of the merged A nodes will then all be labeled by I_B. Since these new B nodes all have labels that are equal to each other, they can all be merged into a single node from which emerge (to their right) nodes that constitute a complete set of projectors $[j_1]_A$, which add to I_A. Hence, the new A node that we now attach to the left of the merged I_B node will be labeled by this sum, that is, by I_A, so we have a double-rooted tree with the two roots each labeled by one of the identity operators, as required for an LOCC protocol.

Note that precisely this same construction continues to work if we replace $[j_2]_B$ by $[\phi_{j_2}^{(j_1)}] = |\phi_{j_2}^{(j_1)}\rangle_B\langle\phi_{j_2}^{(j_1)}|$, with each set $\{|\phi_{j_2}^{(j_1)}\rangle_B\}$ being a complete orthonormal basis, so that Bob does a different projective measurement for each of the outcomes j_1 of Alice's measurement. The reader may find it useful to work this example out by drawing the pictures and explicitly constructing the tree, say for the case that both parties hold qubits and Bob measures either in the σ_z or σ_x basis depending on Alice's outcome.

Example 3. Now we discuss a more involved example, where we demonstrate for a 3×3 system that the separable measurement consisting of projectors onto the nine states of [16] cannot be implemented by LOCC. These states were the first example of what has been called "nonlocality without entanglement," and the projectors onto them are given by

$$\widehat{\mathcal{A}}_1 \otimes \widehat{\mathcal{B}}_1 = [1]_A[1]_B,$$
$$\widehat{\mathcal{A}}_2 \otimes \widehat{\mathcal{B}}_2 = [0]_A[0+1]_B,$$



$\widehat{\mathcal{B}}_6$
$\widehat{\mathcal{A}}_6$
$\widehat{\mathcal{B}}_7$
$\widehat{\mathcal{A}}_7$
$(\widehat{\mathcal{B}}_6 = \widehat{\mathcal{B}}_7)$
$([1]_A + [2]_A)$
$[0]_B$
$[1+2]_A$
$[1-2]_A$
$\widehat{\mathcal{B}}_8$
$\widehat{\mathcal{A}}_8$
$\widehat{\mathcal{B}}_9$
$\widehat{\mathcal{A}}_9$
$(\widehat{\mathcal{B}}_8 = \widehat{\mathcal{B}}_9)$
$([0]_A + [1]_A)$
$[2]_B$
$[0+1]_A$
$[0-1]_A$

FIG. 3. First step of our attempt at constructing an LOCC tree from projectors onto the states of [16]; see Eq. (8). Merging of the two pairs of B nodes, $\widehat{\mathcal{B}}_6 = \widehat{\mathcal{B}}_7 = [0]_B$ and $\widehat{\mathcal{B}}_8 = \widehat{\mathcal{B}}_9 = [2]_B$, is shown.

$$\begin{aligned}
\widehat{\mathcal{A}}_3 \otimes \widehat{\mathcal{B}}_3 &= [0]_A[0-1]_B,\\
\widehat{\mathcal{A}}_4 \otimes \widehat{\mathcal{B}}_4 &= [2]_A[1+2]_B,\\
\widehat{\mathcal{A}}_5 \otimes \widehat{\mathcal{B}}_5 &= [2]_A[1-2]_B,\\
\widehat{\mathcal{A}}_6 \otimes \widehat{\mathcal{B}}_6 &= [1+2]_A[0]_B,\\
\widehat{\mathcal{A}}_7 \otimes \widehat{\mathcal{B}}_7 &= [1-2]_A[0]_B,\\
\widehat{\mathcal{A}}_8 \otimes \widehat{\mathcal{B}}_8 &= [0+1]_A[2]_B,\\
\widehat{\mathcal{A}}_9 \otimes \widehat{\mathcal{B}}_9 &= [0-1]_A[2]_B,
\end{aligned} \tag{8}$$

where, for example, $[0+1] = (|0\rangle + |1\rangle)(\langle 0| + \langle 1|)/2$. These are the positive operators that we will use to label the initial two-node trees at the beginning of our construction.

Starting with Bob's nodes, we see that $\widehat{\mathcal{B}}_6 = \widehat{\mathcal{B}}_7$ and $\widehat{\mathcal{B}}_8 = \widehat{\mathcal{B}}_9$, so we may merge each of these pairs of B nodes (and no others) into single nodes. Then, since $\widehat{\mathcal{A}}_6 + \widehat{\mathcal{A}}_7 = [1]_A + [2]_A$ and $\widehat{\mathcal{A}}_8 + \widehat{\mathcal{A}}_9 = [0]_A + [1]_A$, we have the new three-level trees shown in Fig. 3. We next look at these new A nodes together with all the original ones for $j = 1, \ldots, 9$, and we notice that again there are only two pairs that can be merged, $\widehat{\mathcal{A}}_2 = \widehat{\mathcal{A}}_3$ and $\widehat{\mathcal{A}}_4 = \widehat{\mathcal{A}}_5$. Merging these looks very similar to what was just done in Fig. 3. Following this, we return to the B nodes to discover that no two labels are equal to each other, as is also the case for the A nodes. There is nothing more that can be done from this point, and since this is the only way to begin building a tree, we see that no four-level trees can be constructed for the operators of (8) (other than by trivially adding new nodes to the left of the existing trees, as is illustrated for $\widehat{\mathcal{A}}_4 \otimes \widehat{\mathcal{B}}_4$ at the bottom right of Fig. 2, but without additional merging of any trees together). Therefore, there is no LOCC tree compatible with this set of measurement operators, and by the argument in the previous section that every LOCC protocol corresponds to an LOCC tree, we may thus conclude that no LOCC protocol exists for this separable measurement. In fact, it is clear that even if we allow each of the local projectors in this measurement to be varied slightly (that is, by replacing each $\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j$ by another operator that differs from it by a small amount), there will still be nothing one can do after these two steps of merging nodes (that is, if one can even still do those two steps). Hence, this separable measurement cannot even be closely approximated by LOCC. Note also that this conclusion holds no matter what the Kraus operators $\widehat{A}_j \otimes \widehat{B}_j$ happen to be, as long as they correspond to the positive operators of (8), generalizing the discussion around Eqs. (59)– (61) in Sec. VII of [16].

Example 4. Let us now give an example to illustrate how the algorithm of Sec. II B works. First, we will describe in general terms the example and how to construct an LOCC protocol, and then we will go back and show how to do this by using the algorithm directly.

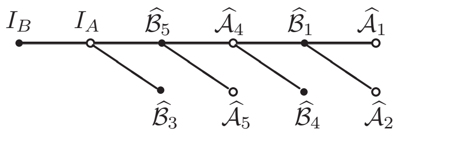
Consider a set of five product operators $\{\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j\}$ satisfying the constraints

$$\begin{aligned} \widehat{\mathcal{B}}_1 &= \widehat{\mathcal{B}}_2 = \widehat{\mathcal{B}}_3, \\ \widehat{\mathcal{B}}_5 &= \widehat{\mathcal{B}}_1 + \widehat{\mathcal{B}}_4, \\ I_B &= \widehat{\mathcal{B}}_3 + \widehat{\mathcal{B}}_5, \\ \widehat{\mathcal{A}}_4 &= \widehat{\mathcal{A}}_1 + \widehat{\mathcal{A}}_2, \\ I_A &= \widehat{\mathcal{A}}_3 = \widehat{\mathcal{A}}_4 + \widehat{\mathcal{A}}_5, \end{aligned} \tag{9}$$

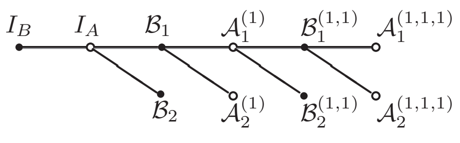
and finally that there are no other linear constraints satisfied by these operators. Since no two of the $\widehat{\mathcal{A}}_j$ are proportional to each other, we cannot merge any of the A nodes to begin the construction. We can, on the other hand, merge the three nodes $\widehat{\mathcal{B}}_1$, $\widehat{\mathcal{B}}_2$, and $\widehat{\mathcal{B}}_3$, as was done in Fig. 2. If we start with this step, however, a bit of thought will lead one to conclude that this leaves a situation where there is nothing else that can be done. No LOCC protocol can be generated by starting the construction this way.

Nonetheless, there is an LOCC protocol for this measurement, as can be seen if we start the construction by merging only nodes $\widehat{\mathcal{B}}_1$ and $\widehat{\mathcal{B}}_2$ and labeling this merged node by $\widehat{\mathcal{B}}_1$. Then, to the left of this merged node, add a new A node labeled by $\widehat{\mathcal{A}}_1 + \widehat{\mathcal{A}}_2 = \widehat{\mathcal{A}}_4$, and then merge this new node with node $\widehat{\mathcal{A}}_4$. Add a new B node labeled as $\widehat{\mathcal{B}}_1 + \widehat{\mathcal{B}}_4 = \widehat{\mathcal{B}}_5$ to the left of this, and merge it with node $\widehat{\mathcal{B}}_5$. Finally, add a new A node labeled as $\widehat{\mathcal{A}}_4 + \widehat{\mathcal{A}}_5$ to the left of this, and merge it with node $\widehat{\mathcal{A}}_3$. The reader may wish to verify that this construction leads to the tree shown in Fig. 4(a). In Fig. 4(b), we have the same tree, but it is now labeled by the quantities $\mathcal{A}_{i_n}^{(\mathcal{S}_n)}$ and $\mathcal{B}_{i_n}^{(\mathcal{S}_n)}$ introduced in the preceding section. By comparing these two trees, the latter quantities can be identified as sums of the $\widehat{\mathcal{A}}_j$ and $\widehat{\mathcal{B}}_j$, respectively, after which it is easy to see that Eqs. (5) and (6) are satisfied (as is guaranteed by our construction).

To illustrate directly how our algorithm works, let us reconsider this example. We start with five two-level trees, with B nodes on the left as shown in Fig. 5(a), one for each of the $\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j$. In step 1 of the algorithm, identify three equivalence classes, $E_1 = \{\widehat{\mathcal{B}}_1,\widehat{\mathcal{B}}_2,\widehat{\mathcal{B}}_3\}$, $E_2 = \{\widehat{\mathcal{B}}_4\}$, and $E_3 = \{\widehat{\mathcal{B}}_5\}$. For step 3(a), since E_2 and E_3 have only a single subset each, creating a copy really means simply keeping that tree; however, for E_1 we have seven distinct subsets: $s_{11} = \{\widehat{\mathcal{B}}_1\}$, $s_{12} = \{\widehat{\mathcal{B}}_2\}$, $s_{13} = \{\widehat{\mathcal{B}}_3\}$, $s_{14} = \{\widehat{\mathcal{B}}_1,\widehat{\mathcal{B}}_2\}$, $s_{15} = \{\widehat{\mathcal{B}}_1,\widehat{\mathcal{B}}_3\}$, $s_{16} = \{\widehat{\mathcal{B}}_2,\widehat{\mathcal{B}}_3\}$, and $s_{17} = \{\widehat{\mathcal{B}}_1,\widehat{\mathcal{B}}_2,\widehat{\mathcal{B}}_3\}$. The first three subsets give back the original


I_B
I_A
$\widehat{\mathcal{B}}_5$
$\widehat{\mathcal{A}}_4$
$\widehat{\mathcal{B}}_1$
$\widehat{\mathcal{A}}_1$
$\widehat{\mathcal{B}}_3$
$\widehat{\mathcal{A}}_5$
$\widehat{\mathcal{B}}_4$
$\widehat{\mathcal{A}}_2$

(a)



(b)

FIG. 4. The tree constructed for the example of Eq. (9). In (a), the nodes are labeled by the desired measurement operators, $\widehat{\mathcal{A}}_j$ and $\widehat{\mathcal{B}}_j$, while in (b) the nodes are labeled by $\mathcal{A}_{i_n}^{(\mathcal{S}_n)}$ and $\mathcal{B}_{i_n}^{(\mathcal{S}_n)}$, the quantities appearing in Eqs. (5) and (6). By comparing these two trees, the latter quantities can be identified in terms of the former.

trees for those B nodes, and then we create additional copies of each of these for each appearance in one of the other subsets. Then for each of those latter subsets we merge the corresponding B nodes [step 3(b)], extend by adding an additional A node [step 3(c)], label the latter by a sum of the $\widehat{q}_{jk}\widehat{\mathcal{A}}_j$ that appear in that tree, and record the constraints [which will be similar to the first line of (9), but with added factors $\widehat{p}_{jk}$; step 3(d)]. The result is shown in Fig. 5(b) (though $\widehat{q}_{jk},\widehat{p}_{jk}$ have been omitted to avoid too much clutter).

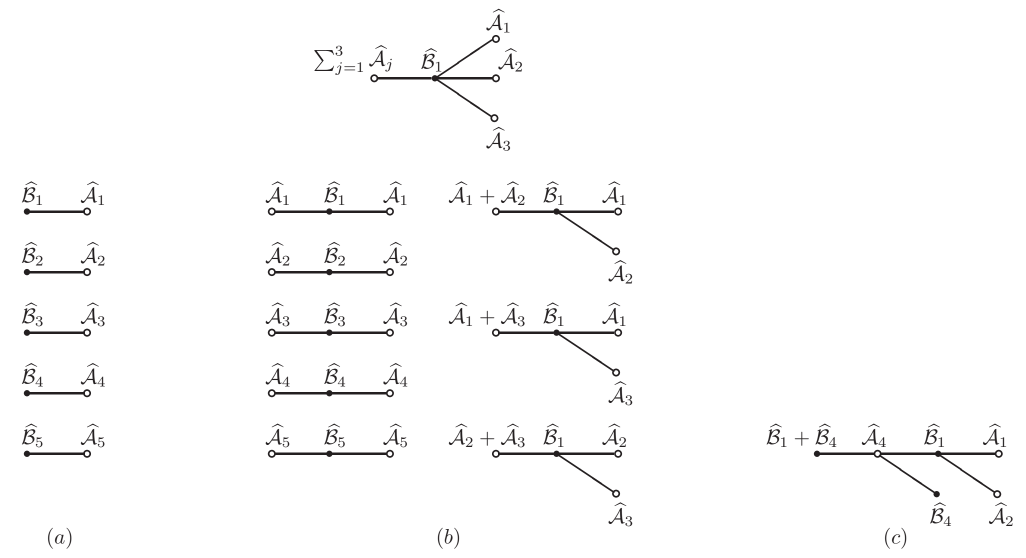
For steps 4–6 we look through these trees that were just constructed and see that none of them contains each of the $\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j$ at least once. Therefore, we continue to step 7 to identify new equivalence classes from intersections of convex cones associated with the sums of $\widehat{\mathcal{A}}_j$ labeling the leftmost nodes in the trees in Fig. 5(b). The only such equivalence class that contains more than one A node in it is $E_4 = \{\widehat{q}_{4k}\widehat{\mathcal{A}}_4,\widehat{q}_{1k}\widehat{\mathcal{A}}_1 + \widehat{q}_{2k}\widehat{\mathcal{A}}_2\}$, and the only new tree that is constructed from this class is shown in Fig. 5(c). The remainder of the construction proceeds in the same way, eventually yielding the LOCC tree shown in Fig. 4.

Example 5. Now we discuss why $\widehat{q}_{jk},\widehat{p}_{jk}$ are useful. Since we want to allow for each $\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j$ to appear on multiple different leaves, it is necessary to introduce positive factors $\widehat{r}_{jk}$ multiplying these operators, with k labeling the different appearances of $\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j$, in order that the sum over all leaves will equal $I_A \otimes I_B$, the required completeness condition on the overall measurement. That is, with $\widehat{r}_j = \sum_k \widehat{r}_{jk}$,

$$I_A \otimes I_B = \sum_j \widehat{r}_j \widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j, \tag{10}$$

which is (3). However, since the conditions (5) and (6) are conditions on $\widehat{\mathcal{A}}_j$ or $\widehat{\mathcal{B}}_j$ separately, it will be useful to write $\widehat{r}_{jk} = \widehat{q}_{jk}\widehat{p}_{jk}$ and consider product operators $\widehat{q}_{jk}\widehat{\mathcal{A}}_j \otimes \widehat{p}_{jk}\widehat{\mathcal{B}}_j$ instead of just $\widehat{r}_{jk}\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j$. Let us illustrate why this is useful, and then we will give a complete example.

Consider three of the operators, $\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j, j = 1,2,3$, and suppose $\widehat{\mathcal{B}}_1 \neq \widehat{\mathcal{B}}_2$ but $\widehat{r}_{11}\widehat{\mathcal{B}}_1 = \widehat{r}_{21}\widehat{\mathcal{B}}_2$ for some positive constants $\widehat{r}_{11}$ and $\widehat{r}_{21}$. Then, we cannot merge $\widehat{\mathcal{A}}_1 \otimes \widehat{\mathcal{B}}_1$ and



$\sum_{j=1}^{3}\widehat{\mathcal{A}}_j$
$\widehat{\mathcal{B}}_1$
$\widehat{\mathcal{A}}_1$
$\widehat{\mathcal{A}}_2$
$\widehat{\mathcal{A}}_3$
$\widehat{\mathcal{A}}_1+\widehat{\mathcal{A}}_2$
$\widehat{\mathcal{A}}_1+\widehat{\mathcal{A}}_3$
$\widehat{\mathcal{A}}_2+\widehat{\mathcal{A}}_3$
$\widehat{\mathcal{B}}_1+\widehat{\mathcal{B}}_4$
$\widehat{\mathcal{A}}_4$
$\widehat{\mathcal{B}}_4$
(a)
(b)
(c)

FIG. 5. The trees constructed by the first two passes through our algorithm for the example of Eq. (9): (a) the five two-level trees represent the five $\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j$, which are the input to the algorithm; (b) after the first pass through the algorithm, we keep all the original two-level trees, now with one additional node added to turn them into three-level trees, and add four more trees corresponding to the four different ways to merge the three proportional B nodes, $\widehat{\mathcal{B}}_1,\widehat{\mathcal{B}}_2,\widehat{\mathcal{B}}_3$; (c) after the second pass through the algorithm we still have all three-level trees that are shown in (b), but with another added node (these trees are not shown here), along with one additional tree constructed from merging two trees from (b), the one that had $\widehat{\mathcal{A}}_4$ as its leftmost node and the one that had $\widehat{\mathcal{A}}_1+\widehat{\mathcal{A}}_2$ as its leftmost node. (The reader should mentally insert the factors $\widehat{q}_{jk},\widehat{p}_{jk}$, which have been omitted for clarity, to avoid clutter.)

$\widehat{\mathcal{A}}_2 \otimes \widehat{\mathcal{B}}_2$ but can merge $\widehat{r}_{11}\widehat{\mathcal{A}}_1 \otimes \widehat{\mathcal{B}}_1$ with $\widehat{r}_{21}\widehat{\mathcal{A}}_2 \otimes \widehat{\mathcal{B}}_2$ at the B nodes, labeling this merged node as $\widehat{r}_{11}\widehat{\mathcal{B}}_1$. Following our method of construction, after merging these two B nodes, we introduce a new node to the left and label it by the sum of the A nodes that emerge to the right of these two merged B nodes, that is, by $\widehat{\mathcal{A}}_1+\widehat{\mathcal{A}}_2$. Now, it may be that this sum is not equal to $\widehat{\mathcal{A}}_3$, so the $\widehat{\mathcal{A}}_3$ node cannot be merged into the tree we have just constructed. However, if $\widehat{q}_{31}\widehat{\mathcal{A}}_3=\widehat{q}_{11}\widehat{\mathcal{A}}_1+\widehat{q}_{21}\widehat{\mathcal{A}}_2$ for some positive constants $\widehat{q}_{11}$, $\widehat{q}_{21}$, and $\widehat{q}_{31}$, then we should include a tree that has $\widehat{q}_{31}\widehat{\mathcal{A}}_3 \otimes \widehat{p}_{31}\widehat{\mathcal{B}}_3$ merged at the A node with a tree created by merging $\widehat{q}_{11}\widehat{\mathcal{A}}_1 \otimes \widehat{p}_{11}\widehat{\mathcal{B}}_1$ with $\widehat{q}_{21}\widehat{\mathcal{A}}_2 \otimes \widehat{p}_{21}\widehat{\mathcal{B}}_2$ at the B node. This can be done by using $\widehat{q}_{jk}$ and $\widehat{p}_{jk}$ instead of the single coefficient $\widehat{r}_{jk}$, leading to the two constraints,

$$
\begin{aligned}
\widehat{p}_{11}\widehat{\mathcal{B}}_1 &= \widehat{p}_{21}\widehat{\mathcal{B}}_2,\\
\widehat{q}_{31}\widehat{\mathcal{A}}_3 &= \widehat{q}_{11}\widehat{\mathcal{A}}_1+\widehat{q}_{21}\widehat{\mathcal{A}}_2,
\end{aligned}
\tag{11}
$$

and along with all other constraints, these will determine the set of coefficients $\widehat{q}_{jk},\widehat{p}_{jk}$.

Now we provide the complete example to demonstrate these ideas. This example will, in addition, show how one of the $\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j$ is used more than once, and we will also explicitly write down the constraints obtained as we merge nodes together.

Consider a measurement having seven product operators, for which the local positive operators satisfy the following conditions:

$$
\begin{aligned}
\widehat{\mathcal{B}}_1 &= 2\widehat{\mathcal{B}}_2 = 3\widehat{\mathcal{B}}_3,\\
\widehat{\mathcal{B}}_6 &= \widehat{\mathcal{B}}_1+\widehat{\mathcal{B}}_4,\\
\widehat{\mathcal{B}}_7 &= \widehat{\mathcal{B}}_1+2\widehat{\mathcal{B}}_5,\\
I_B &= \widehat{\mathcal{B}}_6+\widehat{\mathcal{B}}_7,\\
2\widehat{\mathcal{A}}_4 &= \widehat{\mathcal{A}}_1+\widehat{\mathcal{A}}_2,\\
3\widehat{\mathcal{A}}_5 &= \widehat{\mathcal{A}}_1+\widehat{\mathcal{A}}_3,\\
I_A &= \widehat{\mathcal{A}}_6+2\widehat{\mathcal{A}}_4=\widehat{\mathcal{A}}_7+3\widehat{\mathcal{A}}_5.
\end{aligned}
\tag{12}
$$

It is not necessary for us to have discovered these conditions ahead of time; it is only necessary that we know the operators themselves and are able to check for when positive linear combinations of some of them can be equal to positive linear combinations of others (intersections of convex cones); in the following we will assume that these conditions are not yet known.

Begin by checking for sets of $\widehat{\mathcal{B}}_j$ that are proportional to each other. We find that the only such cases are for $\widehat{\mathcal{B}}_1,\widehat{\mathcal{B}}_2$, and $\widehat{\mathcal{B}}_3$, which are all proportional. We could merge all three together or any of the three pairs, and in general we will want to check all these possibilities, but because of (12) we will see below that only the pairs $j=1,2$ and $j=1,3$ are needed. As we are using $j=1$ twice, multiply each use of $\widehat{\mathcal{A}}_1$ and $\widehat{\mathcal{B}}_1$ by different coefficients $\widehat{q}_{11},\widehat{p}_{11}$ for the first use and $\widehat{q}_{12},\widehat{p}_{12}$ for the second one. Then, merging these pairs of B nodes leads to the constraints

$$
\begin{aligned}
\widehat{p}_{11}\widehat{\mathcal{B}}_1 &= \widehat{p}_2\widehat{\mathcal{B}}_2,\\
\widehat{p}_{12}\widehat{\mathcal{B}}_1 &= \widehat{p}_3\widehat{\mathcal{B}}_3
\end{aligned}
\tag{13}
$$

(one could replace p_2 by p_{21} and similarly for p_3, but since we will end up using only one copy of each of these operators in building a tree, there will be no need for the extra index in this particular example). Attach a new node to the left of each of these merged trees, one of which will be labeled by $\widehat{q}_{11}\widehat{\mathcal{A}}_1+\widehat{q}_2\widehat{\mathcal{A}}_2$ and the other by $\widehat{q}_{12}\widehat{\mathcal{A}}_1+\widehat{q}_3\widehat{\mathcal{A}}_3$.

We next check the A nodes to see which ones can be merged. From (12), we will find that $\widehat{\mathcal{A}}_4$ lies within the convex cone

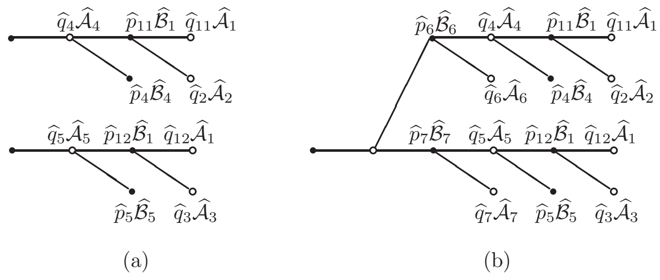

$\widehat{q}_4\widehat{\mathcal{A}}_4$
$\widehat{p}_{11}\widehat{\mathcal{B}}_1$
$\widehat{q}_{11}\widehat{\mathcal{A}}_1$
$\widehat{p}_4\widehat{\mathcal{B}}_4$
$\widehat{q}_2\widehat{\mathcal{A}}_2$
$\widehat{q}_5\widehat{\mathcal{A}}_5$
$\widehat{p}_{12}\widehat{\mathcal{B}}_1$
$\widehat{q}_{12}\widehat{\mathcal{A}}_1$
$\widehat{p}_5\widehat{\mathcal{B}}_5$
$\widehat{q}_3\widehat{\mathcal{A}}_3$
(a)
$\widehat{p}_6\widehat{\mathcal{B}}_6$
$\widehat{q}_4\widehat{\mathcal{A}}_4$
$\widehat{p}_{11}\widehat{\mathcal{B}}_1$
$\widehat{q}_{11}\widehat{\mathcal{A}}_1$
$\widehat{q}_6\widehat{\mathcal{A}}_6$
$\widehat{p}_4\widehat{\mathcal{B}}_4$
$\widehat{q}_2\widehat{\mathcal{A}}_2$
$\widehat{p}_7\widehat{\mathcal{B}}_7$
$\widehat{q}_5\widehat{\mathcal{A}}_5$
$\widehat{p}_{12}\widehat{\mathcal{B}}_1$
$\widehat{q}_{12}\widehat{\mathcal{A}}_1$
$\widehat{q}_7\widehat{\mathcal{A}}_7$
$\widehat{p}_5\widehat{\mathcal{B}}_5$
$\widehat{q}_3\widehat{\mathcal{A}}_3$
(b)

FIG. 6. The tree constructed for Example 5: (a) midway through the construction and (b) the final tree.

formed by $\widehat{\mathcal{A}}_1$ and $\widehat{\mathcal{A}}_2$, while $\widehat{\mathcal{A}}_5$ lies within the convex cone formed by $\widehat{\mathcal{A}}_1$ and $\widehat{\mathcal{A}}_3$, so we can merge the $\widehat{\mathcal{A}}_4$ node and the $\widehat{\mathcal{A}}_5$ node into the appropriate tree created at the previous step where B nodes were merged to start things off. Doing this and attaching new nodes on the left leave us with Fig. 6(a). The constraints obtained for these mergings are

$$\begin{aligned}\widehat{q}_4\widehat{\mathcal{A}}_4 &= \widehat{q}_{11}\widehat{\mathcal{A}}_1 + \widehat{q}_2\widehat{\mathcal{A}}_2,\\ \widehat{q}_5\widehat{\mathcal{A}}_5 &= \widehat{q}_{12}\widehat{\mathcal{A}}_1 + \widehat{q}_3\widehat{\mathcal{A}}_3.\end{aligned} \tag{14}$$

The leftmost nodes of the two trees we have constructed so far will next be labeled by $\widehat{p}_{11}\widehat{\mathcal{B}}_1 + \widehat{p}_4\widehat{\mathcal{B}}_4$ and $\widehat{p}_{12}\widehat{\mathcal{B}}_1 + \widehat{p}_5\widehat{\mathcal{B}}_5$, respectively. The next step of our construction is to now check again for which B nodes can be merged, and we will find [according to (12)] that $\widehat{\mathcal{B}}_6$ lies in the cone of $\widehat{\mathcal{B}}_1$ and $\widehat{\mathcal{B}}_4$, while $\widehat{\mathcal{B}}_7$ lies within that of $\widehat{\mathcal{B}}_1$ and $\widehat{\mathcal{B}}_5$. Hence with the constraints

$$\begin{aligned}\widehat{p}_6\widehat{\mathcal{B}}_6 &= \widehat{p}_{11}\widehat{\mathcal{B}}_1 + \widehat{p}_4\widehat{\mathcal{B}}_4,\\ \widehat{p}_7\widehat{\mathcal{B}}_7 &= \widehat{p}_{12}\widehat{\mathcal{B}}_1 + \widehat{p}_5\widehat{\mathcal{B}}_5,\end{aligned} \tag{15}$$

we can merge $\widehat{q}_6\widehat{\mathcal{A}}_6 \otimes \widehat{p}_6\widehat{\mathcal{B}}_6$ and $\widehat{q}_7\widehat{\mathcal{A}}_7 \otimes \widehat{p}_7\widehat{\mathcal{B}}_7$, one to each of the leftmost B nodes in Fig. 6(a), and those two B nodes can then be relabeled in agreement with how they are labeled in Fig. 6(b).

Next, attach new A nodes to the left of each of these trees and label them as $\widehat{q}_4\widehat{\mathcal{A}}_4 + \widehat{q}_6\widehat{\mathcal{A}}_6$ and $\widehat{q}_5\widehat{\mathcal{A}}_5 + \widehat{q}_7\widehat{\mathcal{A}}_7$, respectively. When next checking for intersections of convex cones for A nodes, we will find that these two nodes do indeed share an intersection. Therefore, merge them into one node, labeled by either of these sums, and attach a new B node to the left, with the label $\widehat{p}_6\widehat{\mathcal{B}}_6 + \widehat{p}_7\widehat{\mathcal{B}}_7$. We obtain one additional constraint,

$$\widehat{q}_4\widehat{\mathcal{A}}_4 + \widehat{q}_6\widehat{\mathcal{A}}_6 = \widehat{q}_5\widehat{\mathcal{A}}_5 + \widehat{q}_7\widehat{\mathcal{A}}_7. \tag{16}$$

We now have a single tree in which every $\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j$ appears at least once, so we can include two additional constraints that the two roots of this tree are I_A, I_B,

$$\begin{aligned}I_A &= \widehat{q}_4\widehat{\mathcal{A}}_4 + \widehat{q}_6\widehat{\mathcal{A}}_6,\\ I_B &= \widehat{p}_6\widehat{\mathcal{B}}_6 + \widehat{p}_7\widehat{\mathcal{B}}_7.\end{aligned} \tag{17}$$

Thus, we have constructed a closed, double-rooted tree for this set of measurement outcomes. Equations (13)–(17) constitute the complete set of constraints that must be solved to determine if this tree represents a valid LOCC protocol. One solution to these constraints is indicated by (12), which does indeed yield such a protocol. Notice that even though there are only seven outcomes defining the measurement, there are actually eight final outcomes of the protocol, an obvious consequence of the fact that $\widehat{\mathcal{A}}_1 \otimes \widehat{\mathcal{B}}_1$ appears twice.

IV. CONCLUSIONS

To summarize, we have seen that every LOCC protocol corresponds to a tree graph with nodes labeled in a way that must satisfy (5) and (6), which constrain the way that any such tree may be constructed from a separable measurement. Given a separable measurement, then, we know how to construct an LOCC protocol if one exists. Furthermore, we see that by constructing all [22] possible trees of depth L working backward from the leaves, we will then also determine when no LOCC consisting of L or fewer rounds exists for this measurement. We note, finally, that it is not difficult to generalize the construction to the case of more than two parties.

ACKNOWLEDGMENTS

I am very grateful to Li Yu, Jonathan Walgate, Ronald de Wolf, Vlad Gheorghiu, and Shiang-Yong Looi for helpful discussions and to the KITP at the University of California, Santa Barbara, where this paper was begun. This work has been supported in part by a grant from the Research Corporation and by the National Science Foundation through Grants No. PHY-0757251 and No. PHY-0551164.

APPENDIX A: PROOF OF MAIN THEOREM

Here we will see that the construction described in the main text produces an LOCC protocol for a given separable measurement corresponding to positive operators $\{\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j\}$, whenever one exists. Every LOCC protocol can be described as a sequence of complete local measurements, each conditioned on previous outcomes. Each local measurement is a branching of possibilities to one of the outcomes of this measurement, and the collection of those outcomes that follow any given outcome of the previous measurement satisfies a completeness relation, as expressed by (1). This branching lends itself directly to representation as a tree graph (without closed loops), which we have illustrated in various figures. As has been described in the main text, the nodes of the tree can be labeled by the positive operators corresponding to the product of Kraus operators implemented by that party up to that point in the protocol, and we have shown that these positive operators must satisfy (5) and (6) at each and every node of the tree [(6) is a direct consequence of (5), along with the branching structure of the tree]. Hence, if we can construct all trees (if any) that (i) are compatible with these two equations at every node, (ii) end with one of the $\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j$ on each of the leaves, and (iii) have each $\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j$ appearing on at least one of the leaves, then we will have found all LOCC protocols for this separable measurement. Since every LOCC protocol corresponds to a tree, if there is no such tree, then there is no LOCC protocol.

We will not be able to construct all LOCC trees, however. The first reason is that we do not presently have an argument that our construction will always terminate on its own, so it is possible the algorithm in Sec. II B could continue indefinitely.

We therefore impose an upper limit L, which is, in principle, arbitrary, on the number of levels (depth) of the trees we construct, equivalent to restricting the number of rounds in the LOCC protocols that are considered.

To understand the second reason we will not construct all LOCC trees, even those of restricted depth, divide the entire collection of trees of depth no greater than L into two classes. Class $\mathcal{C}$ (for congruent) contains all such trees that include at least one node from which emerges (to the right) two or more subtrees that are *congruent*, by which we mean they are identical to each other apart from differing coefficients $\widehat{q}_{jk},\widehat{p}_{jk}$ (that is, differing k indices) multiplying the operators on their leaves (see Fig. 7 in Appendix C for an example). Since *any* number of such congruent subtrees can emerge from any given node and since this number can be changed without altering the fact that the complete tree is a valid LOCC (again, see Appendix C), the number of trees in this class is infinite. Constructing all trees in this class is therefore a difficult, perhaps intractable, problem. The second class of trees is finite and is the complement of the first within the entire set of LOCC trees of depth not exceeding L. This latter class, denoted $\mathcal{D}$ (for distinct), contains all trees in which every node has *only* distinct subtrees emerging from it to the right, no two of which are congruent (note that classes $\mathcal{C}$ and $\mathcal{D}$ are entirely unrelated to the equivalence classes that were introduced in step *1* of the detailed algorithm of Sec. II B).

We will construct all trees in $\mathcal{D}$, and then we will show in Appendix C that if no LOCC tree can be constructed in $\mathcal{D}$, then there are no LOCC trees within $\mathcal{C}$ either. Therefore, if our construction fails, then no LOCC protocol with L or fewer rounds exists for this separable measurement.

We will next give a proof of our main theorem that the construction will yield an LOCC protocol whenever one exists in L or fewer rounds.

1. The construction yields an LOCC tree whenever one exists

We here give a restatement of our main theorem.

Main theorem (restatement). The construction given in algorithmic form in Sec. II B, when restricted to L rounds, builds an LOCC tree for a given set of product operators $\{\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j\}$ whenever an LOCC protocol in L rounds exists for this separable measurement (L is a finite, but otherwise arbitrary, integer).

The main idea of the proof is to imagine that an (arbitrary) LOCC protocol is provided to us and then to show that the algorithm will construct an LOCC protocol for the same measurement. Since the algorithm starts from a known set of $\{\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j\}$, we identify these operators by examining the leaves of the tree associated with the given protocol. Then, from this measurement and the known structure of its LOCC tree, the proof proceeds to show how the algorithm constructs that tree. Since the tree we started with was arbitrary, this proves that the algorithm will construct an LOCC protocol whenever one exists.

Proof. Suppose an LOCC protocol exists and consists of L rounds; then there is an LOCC tree associated with that protocol. If this tree includes direct merging of congruent subtrees, then according to Lemma 2 (proved in Appendix C), there is also an LOCC tree for this measurement that has no such direct merging. From Lemma 1 of Appendix B, the latter tree yields an LOCC protocol. Therefore, we need only show that the construction builds the latter tree, the one without any direct merging of congruent subtrees, and so consider the latter (from here on referred to as the "original" tree) in the following. We may always assume that every branch has the same number of nodes along it by extending shorter ones with additional nodes along a single line (no further branching), with each such node representing a round where that party did nothing (performed the identity operator). For the same reason, we can (arbitrarily) choose to have every leaf be an A node emerging from a B node on its left.

Imagine cutting all the edges joining the latter B nodes to the A nodes on their left (not to the leaf A nodes, which are on their right). This separates all those B nodes from each other, leaving multiple "two-level" trees, each with a single B node and one or more (leaf) A nodes attached to it. No two of those A nodes emerging from a single, given B node will be the same $\widehat{\mathcal{A}}_j$, as this would entail a direct merging of congruent subtrees: two $\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j$ with the same j will not be merged directly to each other in this tree. Each of these individual trees is a merging of several different $\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j$ together at the B node, where those B nodes must all be proportional to each other. That is,

$$\mathcal{B}_{i_{L-1}}^{(S_{L-1})} = p_{jk}\widehat{\mathcal{B}}_j = p_{j'k'}\widehat{\mathcal{B}}_{j'} = \cdots , \qquad \text{(A1)}$$

where $\mathcal{B}_{i_{L-1}}^{(S_{L-1})}$ is the label on that particular B node in the original tree. [Note that we have drawn a distinction between the coefficients q_{jk},p_{jk} from the original tree, which are *known* (by assumption the original tree is given in advance and so is fully known), and the $\widehat{q}_{jk},\widehat{p}_{jk}$ to be used in our construction, which are as yet *unknown*.] Since $\widehat{\mathcal{B}}_j,\widehat{\mathcal{B}}_{j'},\cdots$ are all proportional to each other, then at the first pass through step 3 of the algorithm our construction will build each of the two-level trees that result from this cutting of the original tree, introducing unknown coefficients $\widehat{q}_{jk},\widehat{p}_{jk}$ on the A and B nodes. The algorithm also adds a new A node on the left to each of these and labels them with sums of those $\widehat{q}_{jk}\widehat{\mathcal{A}}_j$.

The cut we made in the original tree may have produced multiple copies of a given (sub)tree. To this point in the construction, we will only have built a single copy because we will not yet have (fore)seen the need for more than one, but the additional copies will arise later. As discussed above, the multiple copies will not be directly merged to each other in the original tree, so they must merge to other subtrees before the resulting composite subtrees merge to each other, where those other subtrees differ from one copy of this subtree to another. When multiple trees that can be merged with a given (fixed) one are present at some stage in our construction, then the algorithm makes the necessary multiple copies of the fixed one so that it can be merged with each of those others. This is done by identifying multiple classes and/or multiple subsets of those classes within which that fixed tree is included. Even at this first stage, this will generally already have occurred if any single $\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j$ appears more than once on the leaves of the original tree.

Returning to the entire tree, cut edges one level further into the tree from the leaves, isolating some number of three-level trees (rather than the two-level ones obtained by the cut made in the previous paragraph), all having A nodes on their left with one or more B nodes emerging to the right. The leftmost A node in each of these trees must satisfy (5), which says that it is equal to a sum of the $q_{jk}\widehat{\mathcal{A}}_j$ that label all the leaf A nodes that emerge from any one of the B nodes emerging from that leftmost A node. If there are multiple such B nodes emerging, then the multiple sums of those individual sets of $q_{jk}\widehat{\mathcal{A}}_j$ necessarily satisfy (6) since by assumption the original tree is a valid LOCC. But satisfying (6) implies that the convex cones generated by those sets of $\widehat{\mathcal{A}}_j$ are mutually intersecting, and from the previous paragraph, those same sets of $\widehat{\mathcal{A}}_j$ appear in the sums labeling the new nodes that have just been added in our construction. Therefore in our construction, a tree will appear that merges those same A nodes that are merged at this level in the original tree (with $\widehat{q}_{jk},\widehat{p}_{jk}$ in place of the original q_{jk},p_{jk}). Therefore, our construction will build all these three-level trees from the two-level ones previously obtained.

Now we can use an inductive argument. Assume that our construction builds all l-level trees that appear in the original tree and labels them with the $\widehat{\mathcal{A}}_j,\widehat{\mathcal{B}}_j$ in exactly the same way as they are labeled in that tree, apart from replacing $q_{jk}\to\widehat{q}_{jk}$ and $p_{jk}\to\widehat{p}_{jk}$. We have just shown this holds for $l=2,3$, so assuming it holds for all levels up to l, if we can show that it then also holds for $l+1$, the proof will be complete. Since all the l-level trees are labeled by sums involving the same set of $\widehat{\mathcal{A}}_j,\widehat{\mathcal{B}}_j$ as in the original tree, we know that when we look for intersections of convex cones generated by these sets, we will always find intersections that will lead us to merge these l-level trees to create every one of the $(l+1)$-level trees that appear from that cut of the original tree [the cut that produces $(l+1)$-level trees]. The reason is that these intersections are simply solutions to equations of the form (6), and we know from the original tree that at least one solution will always exist since $\widehat{q}_{jk}=q_{jk},\widehat{p}_{jk}=p_{jk}$ is such a solution, and this completes the proof. ■

Note that for a given set of $\{\widehat{\mathcal{A}}_j\otimes\widehat{\mathcal{B}}_j\}$, if an LOCC protocol exists in any finite number L' of rounds, then by setting $L=L'$ in the above arguments, we have the immediate corollary that our construction will build an LOCC protocol whenever one exists in *any* finite number of rounds.

Let us illustrate the procedure of the proof by an example. Consider the first cut isolating two-level trees that lie at the leaf end of the original tree, nodes labeled by $\mathcal{A}_{i_L}^{(\mathcal{S}_L)}$ and $\mathcal{B}_{i_{L-1}}^{(\mathcal{S}_{L-1})}$, respectively. Suppose node $\mathcal{B}_1^{(\mathcal{S}_{L-1})}$ branches to three $\mathcal{A}_{i_L}^{(\mathcal{S}_L)}$ ($\mathcal{S}_L=\{\mathcal{S}_{L-1},1\}$) with $i_L=1,2,3$. Since this LOCC implements the $\{\widehat{\mathcal{A}}_j\otimes\widehat{\mathcal{B}}_j\}$, it must be that (for some ordering of the latter and for some set of coefficients q_{j1},p_{j1})

$$\mathcal{B}_1^{(\mathcal{S}_{L-1})}=p_{11}\widehat{\mathcal{B}}_1=p_{21}\widehat{\mathcal{B}}_2=p_{31}\widehat{\mathcal{B}}_3 \tag{A2}$$

and

$$\begin{aligned}\mathcal{A}_1^{(\mathcal{S}_L)}&=q_{11}\widehat{\mathcal{A}}_1,\\ \mathcal{A}_2^{(\mathcal{S}_L)}&=q_{21}\widehat{\mathcal{A}}_2,\\ \mathcal{A}_3^{(\mathcal{S}_L)}&=q_{31}\widehat{\mathcal{A}}_3.\end{aligned} \tag{A3}$$

What this means is that $\widehat{\mathcal{B}}_1,\widehat{\mathcal{B}}_2$, and $\widehat{\mathcal{B}}_3$ are all proportional to each other (each generates a one-dimensional convex cone, and all three of these cones share a mutual intersection as they are identical to each other), from which it immediately follows that the three two-node trees, $\widehat{q}_{j1}\widehat{\mathcal{A}}_j\otimes\widehat{p}_{j1}\widehat{\mathcal{B}}_j, j=1,2,3$, will be merged at the B nodes in our construction, forming a tree having the exact same structure as the two-level tree in which $\mathcal{B}_1^{(\mathcal{S}_{L-1})}$ branches to the three nodes $\mathcal{A}_{i_L}^{(\mathcal{S}_L)}, i_L=1,2,3$, in the original tree. Not only is the structure exactly the same, but in fact the two trees are identical in every way except that the one obtained from our construction has coefficients that are as yet unknown. Although this is a specific example of a B node branching to three A nodes, it should be clear that the ideas are completely general, and for each and every final two-level subtree arising from this cut, $\mathcal{B}_{i_{L-1}}^{(\mathcal{S}'_{L-1})}$ with (possibly) multiple edges emerging to A nodes $\mathcal{A}_{i_L}^{(\mathcal{S}'_L)}$ on the leaves, a tree will be formed in our construction that is identical to it apart from the unknown coefficients.

The next thing is to consider the cut done one level further into the tree from the leaves, isolating sets of three-level subtrees. In our example, if node $\mathcal{B}_1^{(\mathcal{S}_{L-1})}$ emerges from node $\mathcal{A}_1^{(\mathcal{S}_{L-2})}$, then according to (5), it must be that

$$\mathcal{A}_1^{(\mathcal{S}_{L-2})}=\mathcal{A}_1^{(\mathcal{S}_L)}+\mathcal{A}_2^{(\mathcal{S}_L)}+\mathcal{A}_3^{(\mathcal{S}_L)}=q_{11}\widehat{\mathcal{A}}_1+q_{21}\widehat{\mathcal{A}}_2+q_{31}\widehat{\mathcal{A}}_3. \tag{A4}$$

Recall that in our construction, an A node is attached at the left of any merged B node. In the specific example we are here discussing, $\widehat{\mathcal{B}}_1,\widehat{\mathcal{B}}_2$, and $\widehat{\mathcal{B}}_3$ are all merged into a single node, and the newly attached A node will be labeled in our construction as $\widehat{q}_{11}\widehat{\mathcal{A}}_1+\widehat{q}_{21}\widehat{\mathcal{A}}_2+\widehat{q}_{31}\widehat{\mathcal{A}}_3$ (for the choice $\widehat{q}_{j1}=q_{j1}$, this is indeed equal to $\mathcal{A}_1^{(\mathcal{S}_{L-2})}$, demonstrating that, at least to this point in the construction, there exists a solution for the coefficients).

Suppose node $\mathcal{B}_2^{(\mathcal{S}_{L-1})}$ also emerges from node $\mathcal{A}_1^{(\mathcal{S}_{L-2})}$ and emerging from that $\mathcal{B}_2^{(\mathcal{S}_{L-1})}$ node are two $\mathcal{A}_{i_L}^{(\mathcal{S}'_L)}, i_L=1,2$ with $\mathcal{S}'_L=\{\mathcal{S}_{L-1},2\}$. Then, again from (5), it must be that $\mathcal{A}_1^{(\mathcal{S}'_L)}+\mathcal{A}_2^{(\mathcal{S}'_L)}=\mathcal{A}_1^{(\mathcal{S}_{L-2})}$. From the same argument as was used above, we will have

$$\mathcal{B}_2^{(\mathcal{S}_{L-1})}=p_{41}\widehat{\mathcal{B}}_4=p_{51}\widehat{\mathcal{B}}_5 \tag{A5}$$

and

$$\begin{aligned}\mathcal{A}_1^{(\mathcal{S}'_L)}&=q_{41}\widehat{\mathcal{A}}_4,\\ \mathcal{A}_2^{(\mathcal{S}'_L)}&=q_{51}\widehat{\mathcal{A}}_5,\end{aligned} \tag{A6}$$

so that

$$\mathcal{A}_1^{(\mathcal{S}_{L-2})}=q_{11}\widehat{\mathcal{A}}_1+q_{21}\widehat{\mathcal{A}}_2+q_{31}\widehat{\mathcal{A}}_3=q_{41}\widehat{\mathcal{A}}_4+q_{51}\widehat{\mathcal{A}}_5, \tag{A7}$$

and we have assumed for illustration purposes that $j=1,2,3$ are not repeated in the latter set of two $\widehat{\mathcal{A}}_j\otimes\widehat{\mathcal{B}}_j$, though that could certainly also occur. This demonstrates that the two convex cones generated by $\widehat{\mathcal{A}}_1,\widehat{\mathcal{A}}_2,\widehat{\mathcal{A}}_3$ and by $\widehat{\mathcal{A}}_4,\widehat{\mathcal{A}}_5$ intersect with each other (and furthermore $\mathcal{A}_1^{(\mathcal{S}_{L-2})}$ lies within that intersection). Therefore, at the second step of our construction, a tree will be created that includes all these nodes and has the same structure as the corresponding subtree within the original tree: an A node branching to these two B nodes, one of which branches to three A nodes and the other to two. Again we can see that, even though this is only a specific example, the ideas can be applied quite generally, and every subtree within the original tree having a depth of three nodes will also be created in our construction. Furthermore, just as $\mathcal{A}_1^{(\mathcal{S}_{L-2})}$ lies within that intersection of the two convex cones mentioned above, all trees created in our construction to this level will have leftmost nodes labeled in a way that the corresponding $\mathcal{A}_{i_{L-2}}^{(\mathcal{S}_{L-2})}$ will lie in the convex cone generated by the sum of the $\widehat{\mathcal{A}}_j$ that labels the node in that tree we have created.

This sort of analysis can be continued back to the roots of the tree, and from similar arguments one will find that one of the trees created by our construction has a structure identical to the entire original tree. In addition, all nodes of this tree are labeled by positive linear combinations of the $\widehat{\mathcal{A}}_j$ (or $\widehat{\mathcal{B}}_j$), and the $\mathcal{A}_{i_n}^{(\mathcal{S}_n)}$ (or $\mathcal{B}_{i_m}^{(\mathcal{S}_m)}$) on a given node of that original tree will always lie within the convex cone generated by the $\widehat{\mathcal{A}}_j$ (or $\widehat{\mathcal{B}}_j$) labeling the corresponding node in this tree, precisely as was found above for the final few levels of the tree. Therefore, there will always be at least one solution to the constraints on the $\widehat{q}_{jk},\widehat{p}_{jk}$, with that solution being $\widehat{q}_{jk} = q_{jk}$ and $\widehat{p}_{jk} = p_{jk}$, which yields precisely the original LOCC tree. Thus, we see that given any set $\{\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j\}$ for which there exists an LOCC protocol, our construction always builds an LOCC tree, providing an associated LOCC protocol.

APPENDIX B: LOCC TREES AND LOCC PROTOCOLS

In this section, we show that there is an equivalence between LOCC trees and LOCC protocols, which will be useful in that it tells us not only that every LOCC protocol corresponds to such a tree but also that if we can construct a tree of this type, then we can also construct an LOCC protocol.

Lemma 1. A double-rooted tree represents an LOCC protocol if its roots are labeled by I_A,I_B and every one of its nodes (i) has a single edge entering it from the left (excepting the left-most root) and (ii) satisfies (5) for every branch emerging from it toward the right.

The reverse implication, that every LOCC protocol corresponds to such a tree, was shown in the main text. Therefore, this lemma completes the demonstration of an equivalence between these trees and LOCC protocols.

Proof. We will construct an LOCC protocol from any tree satisfying the conditions of the lemma. Each (nonroot) node of the tree is labeled by $\mathcal{A}_{i_n}^{(\mathcal{S}_n)}$ or $\mathcal{B}_{i_n}^{(\mathcal{S}_n)}$, which satisfy (5), reproduced here for convenience,

$$\sum_{i_n} \mathcal{A}_{i_n}^{(\mathcal{S}_n)} = \mathcal{A}_{i_{n-2}}^{(\mathcal{S}_{n-2})} \tag{B1}$$

and similarly for Bob's operators. The phrase "for every branch emerging from it toward the right" in (ii) of the statement of the lemma should be understood in (B1) to mean "for every i_{n-1}," as each of those branches from node $(\mathcal{S}_{n-2},i_{n-2})$ are labeled by one of the integers i_{n-1}.

Suppose the first nodes to the right of the roots are A nodes (the arguments are the same if they are B nodes). Define Kraus operators

$$A_{i_1} = \sqrt{\mathcal{A}_{i_1}}, \tag{B2}$$

and choose the unique positive square root. Then, we have that

$$\sum_{i_1} A_{i_1}^\dagger A_{i_1} = \sum_{i_1} \mathcal{A}_{i_1} = I_A, \tag{B3}$$

which follows from (B1) with $n = 1$ along with the fact that the node that is two steps to the left of the $n = 1$ nodes is the leftmost root, equal to I_A. Therefore, this set of Kraus operators is a complete measurement, and we choose it as Alice's first one. By the same argument the Kraus operators

$$B_{i_2}^{(i_1)} = \sqrt{\mathcal{B}_{i_2}^{(i_1)}} \tag{B4}$$

for each fixed i_1 are a complete first measurement for Bob, conditioned on the outcome of Alice's first measurement.

Let us now show how to choose each of Alice's subsequent measurements; Bob's can be chosen by the same approach. For Alice's second measurement, choose

$$A_{i_3}^{(\mathcal{S}_3)} = U_{i_3}^{(\mathcal{S}_3)} \sqrt{\mathcal{A}_{i_3}^{(\mathcal{S}_3)}} A_{i_1}^{-1}, \tag{B5}$$

where $\mathcal{S}_3 = (i_1,i_2)$; we choose unitary $U_{i_3}^{(\mathcal{S}_3)}$ for the convenience of having $A_{i_3}^{(\mathcal{S}_3)}$ be a positive operator (we will mean "positive semidefinite" whenever we write "positive"), and $A_{i_1}^{-1}$ should be understood to be the inverse on its support. Then $A_{i_1}^{-1}A_{i_1} = P_{i_1}$, where P_{i_1} is the projector onto the support of A_{i_1} and is also equal to $A_{i_1}A_{i_1}^{-1}$ since A_{i_1} is a positive operator. Include one additional Kraus operator equal to $I_A - P_{i_1}$. Now,

$$\begin{aligned}\sum_{i_1} A_{i_3}^{(\mathcal{S}_3)\dagger} A_{i_3}^{(\mathcal{S}_3)} &= \sum_{i_3} (A_{i_1}^\dagger)^{-1} \mathcal{A}_{i_3}^{(\mathcal{S}_3)} A_{i_1}^{-1} \\ &= (\sqrt{\mathcal{A}_{i_1}})^{-1} \mathcal{A}_{i_1} (\sqrt{\mathcal{A}_{i_1}})^{-1} = P_{i_1},\end{aligned} \tag{B6}$$

where the second equality follows from (B1). Adding in that last Kraus operator shows this is a complete measurement. The overall Kraus operator (product of Kraus operators implemented by Alice up to any given point in the protocol) corresponding to this added one is $(I_A - P_{i_1})A_{i_1} = 0$, which is why it need not appear as a node in our tree. (If one wishes, an additional branch and node can be added for this "outcome" of the measurement, with no further nodes emerging from it. Since it vanishes identically, this is really unnecessary.)

Assume we have managed to construct a complete set of (positive) Kraus operators for each of Alice's local measurements up to level m. If we can then construct a complete set of Kraus operators for her next measurement $(m + 2)$, we will have shown by induction that the tree yields an LOCC protocol, completing the proof. Thus, by assumption, we have positive $A_{i_m}^{(\mathcal{S}_m)}$ such that

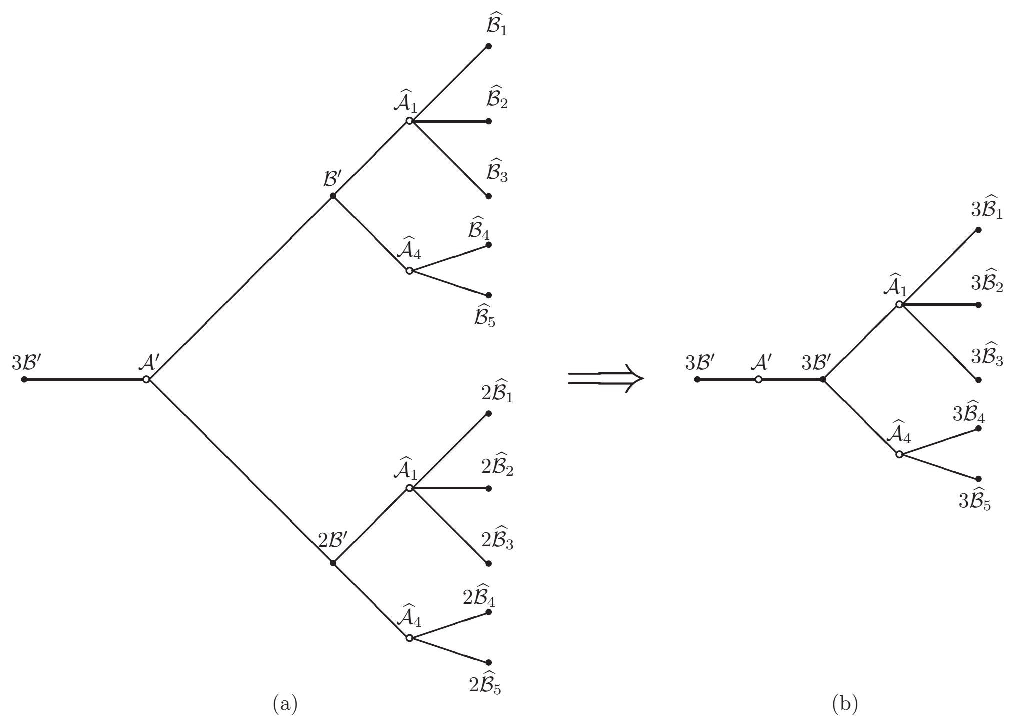
$$\sum_{i_m} A_{i_m}^{(\mathcal{S}_m)\dagger} A_{i_m}^{(\mathcal{S}_m)} = \sum_{i_m} \mathcal{A}_{i_m}^{(\mathcal{S}_m)} = I_A, \tag{B7}$$

where we have included an extra Kraus operator like the one introduced in the previous paragraph. Define

$$A_{i_{m+2}}^{(\mathcal{S}_{m+2})} = U_{i_{m+2}}^{(\mathcal{S}_{m+2})} \sqrt{\mathcal{A}_{i_{m+2}}^{(\mathcal{S}_{m+2})}} \left(A_{i_m}^{(\mathcal{S}_m)}\right)^{-1}, \tag{B8}$$

choosing $U_{i_{m+2}}^{(\mathcal{S}_{m+2})}$ so that each of these is positive. Then,

$$\begin{aligned}\sum_{i_{m+2}} A_{i_{m+2}}^{(\mathcal{S}_{m+2})\dagger} A_{i_{m+2}}^{(\mathcal{S}_{m+2})} &= \sum_{i_{m+2}} \left(A_{i_m}^{(\mathcal{S}_m)\dagger}\right)^{-1} \mathcal{A}_{i_{m+2}}^{(\mathcal{S}_{m+2})} \left(A_{i_m}^{(\mathcal{S}_m)}\right)^{-1} \\ &= \left(A_{i_m}^{(\mathcal{S}_m)\dagger}\right)^{-1} \mathcal{A}_{i_m}^{(\mathcal{S}_m)} \left(A_{i_m}^{(\mathcal{S}_m)}\right)^{-1} = P_{i_m}^{(\mathcal{S}_m)},\end{aligned} \tag{B9}$$



$3\mathcal{B}'$
$\mathcal{A}'$
$\mathcal{B}'$
$\widehat{\mathcal{A}}_1$
$\widehat{\mathcal{A}}_4$
$\widehat{\mathcal{B}}_1$
$\widehat{\mathcal{B}}_2$
$\widehat{\mathcal{B}}_3$
$\widehat{\mathcal{B}}_4$
$\widehat{\mathcal{B}}_5$
$2\mathcal{B}'$
$2\widehat{\mathcal{B}}_1$
$2\widehat{\mathcal{B}}_2$
$2\widehat{\mathcal{B}}_3$
$2\widehat{\mathcal{B}}_4$
$2\widehat{\mathcal{B}}_5$
(a)
$3\mathcal{B}'$
$\mathcal{A}'$
$3\mathcal{B}'$
$3\widehat{\mathcal{B}}_1$
$3\widehat{\mathcal{B}}_2$
$3\widehat{\mathcal{B}}_3$
$3\widehat{\mathcal{B}}_4$
$3\widehat{\mathcal{B}}_5$
(b)

FIG. 7. Illustration of why it is not necessary to consider merging multiple copies of congruent subtrees. To simplify the notation, all coefficients ($\widehat{q}_{jk},\widehat{p}_{jk}$) have been taken to be integers for illustration purposes. We have that $\mathcal{A}' = \widehat{\mathcal{A}}_1 + \widehat{\mathcal{A}}_4$ and $\mathcal{B}' = \widehat{\mathcal{B}}_1 + \widehat{\mathcal{B}}_2 + \widehat{\mathcal{B}}_3 = \widehat{\mathcal{B}}_4 + \widehat{\mathcal{B}}_5$. Any valid LOCC tree in which the subtree in (a) appears can be replaced by a valid LOCC tree that has this subtree replaced by the simpler one in (b). The reason is that the leftmost nodes (one A and one B) are the same in these two subtrees, so whatever else (a) connects to can just as well be connected to (b).

and after including that one additional Kraus operator equal to $I_A - P_{i_m}^{(\mathcal{S}_m)}$ with $P_{i_m}^{(\mathcal{S}_m)} = A_{i_m}^{(\mathcal{S}_m)}(A_{i_m}^{(\mathcal{S}_m)})^{-1} = (A_{i_m}^{(\mathcal{S}_m)})^{-1}A_{i_m}^{(\mathcal{S}_m)}$, we see that this is a complete set for Alice's next measurement. Note once again that $(I_A - P_{i_m}^{(\mathcal{S}_m)})A_{i_m}^{(\mathcal{S}_m)} = 0$, so this extra Kraus operator has zero probability of occurring in the protocol, completing the proof. ■

If one wishes to obtain a specific set of Kraus operators at the end of the protocol (levels $m = L-1, L$) and if these Kraus operators are compatible with the positive operators $\mathcal{A}_{i_m}^{(\mathcal{S}_m)}, \mathcal{B}_{i_m}^{(\mathcal{S}_m)}$ (for those same values of m), this can always be done by adjusting $U_{i_m}^{(\mathcal{S}_m)}$ at these levels.

APPENDIX C: MERGING CONGRUENT SUBTREES DIRECTLY TO ONE ANOTHER

In this section, we provide the final piece of the puzzle by proving the following result, which was assumed in the proof of the main theorem.

Lemma 2. If an LOCC tree exists for a given set of product operators $\{\widehat{\mathcal{A}}_j \otimes \widehat{\mathcal{B}}_j\}$ and includes direct merging of congruent subtrees, then there also exists an LOCC tree for the same set of operators that does not include any such direct merging.

The reason it is not necessary to consider trees with congruent subtrees merged directly to one another is that those congruent subtrees can easily be combined into a single subtree. Note that, being congruent, every node in each of these subtrees has a counterpart in the other subtrees, where all these counterparts are labeled by linear combinations of the exact same set of operators, the only difference being that the coefficients, $\widehat{q}_{jk},\widehat{p}_{jk}$, are different. This follows from (5) and the fact that the subtrees are congruent, which implies that the set of final outcomes (and the number of copies of each) at the leaves of these subtrees is the same.

Before proceeding to the proof of Lemma 2, let us first illustrate how, from a tree that has several congruent subtrees that are merged directly, we can construct a tree that has only one of those subtrees. The node at which they are merged corresponds to one of the parties, say A. Keep everything in the original tree unchanged except for these congruent subtrees. Then, do the following.

(1) Erase all but one of those congruent subtrees in their entirety.

(2) Leave all the A-node labels unchanged in the remaining subtree (because it is an A node where the congruent subtrees are merged).

(3) Replace each B-node label in the remaining subtree by a sum, over all counterpart nodes for that particular node, including the one in the subtree that remains, of the linear combination of operators that label those counterparts (this sums the $\widehat{p}_{jk}$ appearing on all those counterpart nodes for each fixed j).

This is illustrated in Fig. 7. If the original tree is valid for LOCC, then so is the new one. The reason this works is that the leftmost nodes (one A and one B) are the same in the two subtrees in Figs. 7(a) and 7(b), so whatever else the subtree in Fig. 7(a) connects to can just as well be connected to the subtree in Fig. 7(b). If Fig. 7(a) is connected to other subtrees at the $3\mathcal{B}'$ node (note that this node is not part of the congruent subtrees), then those other subtrees can also be connected to the leftmost $3\mathcal{B}'$ node in Fig. 7(b). Everything else in the entire tree will remain unchanged. If, on the other hand, other subtrees are connected to the $\mathcal{A}'$ node in Fig. 7(a), they can also be connected to the $\mathcal{A}'$ node in Fig. 7(b), though then the label $3\mathcal{B}'$ on the leftmost node in both Figs. 7(a) and 7(b) would need to be altered, but this alteration will be the same in Fig. 7(b) as it is in Fig. 7(a). It should be noted that, while the new tree yields a valid LOCC protocol if the original one did, the new tree may correspond to a different set of weights $\widehat{r}_j = \sum_k \widehat{q}_{jk}\widehat{p}_{jk}$ as opposed to those in the original tree. This may be important to Alice and Bob depending on the context in which they wish to use the LOCC protocol. We leave for future work the question of when it will be possible to construct a larger tree from a smaller one by introducing direct merging of congruent nodes, with the aim of obtaining a specified set of weights.

Now we turn to the proof of the lemma.

Proof of Lemma 2. If we can combine two congruent subtrees into one, then we can combine any number of congruent subtrees into one simply by combining them two at a time. Therefore, consider that subtrees T_1 and T_2 are combined to become $\widehat{T}$ according to the prescription described above: erase T_2 and in T_1 leave the A nodes unchanged and replace each B node by the sum of that node with its counterpart from T_2. Since the A nodes are unchanged throughout the entire tree, then the sums in (5) and (6) for $\mathcal{A}_{i_m}^{(\mathcal{S}_m)}$ are also unchanged, so these required conditions will still be satisfied, assuming they were satisfied in the original tree. Thus, we need only demonstrate that these conditions will also still be satisfied for the B nodes.

Label each B node in T_1 as $\mathcal{B}_{i_m}^{(\mathcal{S}_m)}$, the corresponding node in T_2 as $\mathcal{B}_{i_m}^{(\mathcal{S}'_m)}$, and that in $\widehat{T}$ as $\widehat{\mathcal{B}}_{i_m}^{(\mathcal{S}_m)}$. The indices denoting position in T_1 and in $\widehat{T}$ are exactly the same $(\mathcal{S}_m,i_m)$ because these two subtrees have the exact same structure and lie in the same position within their respective overall trees. The indices denoting position within T_2 are not quite the same as those, differing only at the node where T_1 and T_2 are merged to each other, so that $\mathcal{S}'_m$ differs from $\mathcal{S}_m$ only in the index corresponding to the different branch the two follow from that particular node. From the procedure described above, we have

$$\widehat{\mathcal{B}}_{i_m}^{(\mathcal{S}_m)} = \mathcal{B}_{i_m}^{(\mathcal{S}_m)} + \mathcal{B}_{i_m}^{(\mathcal{S}'_m)} \tag{C1}$$

for each m that labels a node in T_1 (and therefore also in T_2), and from (5), we know that

$$\begin{aligned}\mathcal{B}_{i_m}^{(\mathcal{S}_m)} &= \sum_{i_{m+2}} \mathcal{B}_{i_{m+2}}^{(\mathcal{S}_{m+2})},\\ \mathcal{B}_{i_m}^{(\mathcal{S}'_m)} &= \sum_{i_{m+2}} \mathcal{B}_{i_{m+2}}^{(\mathcal{S}'_{m+2})}.\end{aligned} \tag{C2}$$

Adding the last two equations together, we obtain

$$\widehat{\mathcal{B}}_{i_m}^{(\mathcal{S}_m)} = \sum_{i_{m+2}} \left(\mathcal{B}_{i_{m+2}}^{(\mathcal{S}_{m+2})} + \mathcal{B}_{i_{m+2}}^{(\mathcal{S}'_{m+2})}\right) = \sum_{i_{m+2}} \widehat{\mathcal{B}}_{i_{m+2}}^{(\mathcal{S}_{m+2})}, \tag{C3}$$

showing that within $\widehat{T}$, (5) is satisfied at every B node.

The rest of the original tree outside $\widehat{T}$ has been unaltered by this procedure for merging T_1 with T_2. We need to check that (5) will continue to be satisfied at every B node outside of $\widehat{T}$ in the newly formed tree. Suppose the A node at which T_1 and T_2 were merged is $\mathcal{A}_{i_{M+1}}^{(\mathcal{S}_{M+1})}$. From (5), the B node $\mathcal{B}_{i_M}^{(\mathcal{S}_M)}$ from which $\mathcal{A}_{i_{M+1}}^{(\mathcal{S}_{M+1})}$ emerges is, in the original tree, equal to

$$\mathcal{B}_{i_M}^{(\mathcal{S}_M)} = \sum_{i_{M+2}} \left(\mathcal{B}_{i_{M+2}}^{(\mathcal{S}_{M+2})} + \mathcal{B}_{i_{M+2}}^{(\mathcal{S}'_{M+2})}\right) + \Delta, \tag{C4}$$

where the terms in the sum are from T_1 and T_2, respectively. The quantity Δ includes everything that contributes from outside T_1 and T_2 and is unchanged in the new tree, in which $\mathcal{B}_{i_M}^{(\mathcal{S}_M)}$ has also not changed from what it was in the original tree (it is to the left of $\mathcal{A}_{i_{M+1}}^{(\mathcal{S}_{M+1})}$ and therefore not a part of T_1,T_2). Therefore, $\mathcal{B}_{i_M}^{(\mathcal{S}_M)}$ remains equal to (C4), whereas from (5), it should be equal to

$$\mathcal{B}_{i_M}^{(\mathcal{S}_M)} = \sum_{i_{M+2}} \widehat{\mathcal{B}}_{i_{M+2}}^{(\mathcal{S}_{M+2})} + \Delta \tag{C5}$$

in the new tree, which it clearly is.

We now argue that all other B nodes in the new tree also satisfy (5). Those $\mathcal{B}_{i_m}^{(\mathcal{S}_m)}$ that are not in $\widehat{T}$ and not downstream (to the left) of it are unchanged from the original tree, as their upstream B neighbors $\mathcal{B}_{i_{m+2}}^{(\mathcal{S}_{m+2})}$ are, so it holds for these nodes. Those nodes downstream from $\widehat{T}$ are $\mathcal{B}_{i_M}^{(\mathcal{S}_M)}$, and those downstream from it are $\mathcal{B}_{i_{M-2}}^{(\mathcal{S}_{M-2})}$, $\mathcal{B}_{i_{M-4}}^{(\mathcal{S}_{M-4})}$, etc. The latter are each, from (5), equal to sums of the B nodes that lie immediately to the right in the original tree, and those B nodes to the right are the same in the new tree as they were in the original one, so they still satisfy (5), which is therefore seen to be satisfied at every node in the new tree. Since the roots are also unchanged in the new tree, they remain equal to I_A,I_B (even if these roots are the previously mentioned nodes $\mathcal{A}_{i_{M+1}}^{(\mathcal{S}_{M+1})}$ and $\mathcal{B}_{i_M}^{(\mathcal{S}_M)}$). Thus, from Lemma 1, this tree is LOCC and yields a valid LOCC protocol, completing the proof. ∎

APPENDIX D: COMPLEXITY OF THE CONSTRUCTION

As indicated in the algorithm presented above for our construction of LOCC trees, the maximum number of trees generated at the lth pass through the algorithm is $N_l = 2^{N_{l-1}} - 1$, where N_{l-1} is the maximum number of trees that could have been created at the previous pass through. Therefore, the complexity of this method of searching for an LOCC tree could be enormous, apparently as much as "multiply-exponential," with the space required to store all the trees as one proceeds being $\mathcal{O}(2^{N_L})$ for L rounds. Nonetheless, this is a finite upper bound on the number of trees that can be generated in constructing a protocol having no more than L rounds. It should also be pointed out that this complexity is very much a "worst-case scenario" and is likely to be a rather loose upper bound. We believe, though we have no proof, that almost all cases (and perhaps even *all* cases) will require much less in the way of computational resources than this bound represents.

The total number of final outcomes (leaves) in any protocol constructed in this way is finite but potentially quite large. For example, one might generate N_l different trees after l passes through the algorithm, and each of these large number of trees will generally have a large number of leaves on it. Then one could imagine at the next pass it might be possible to combine all these trees into a single tree, so that all the leaves of those trees become leaves of this one combined tree. Clearly, the total number of final outcomes would then be enormous. There is nonetheless a finite upper bound, and one can always devise a protocol that has more final outcomes than that bound. Does this then mean that there exist measurements for which an LOCC protocol exists but for which our construction fails to find one? The answer is no for the reason that if one allows direct merging of congruent subtrees (see Appendix C), then one can end up with any number of final outcomes in such a tree. Our construction will not build this tree, but from Lemma 2, it will still yield an LOCC protocol for the same measurement, just one that has a smaller number of final outcomes. The direct implication is that in order for a protocol to have a number of final outcomes exceeding that finite upper bound, the corresponding tree *must* include direct merging of congruent subtrees.

[1] J. I. Cirac, A. K. Ekert, S. F. Huelga, and C. Macchiavello, Phys. Rev. A **59**, 4249 (1999).

[2] R. Raussendorf and H. J. Briegel, Phys. Rev. Lett. **86**, 5188 (2001).

[3] C. H. Bennett, D. P. DiVincenzo, J. A. Smolin, and W. K. Wootters, Phys. Rev. A **54**, 3824 (1996).

[4] M. A. Nielsen, Phys. Rev. Lett. **83**, 436 (1999).

[5] J. Walgate, A. J. Short, L. Hardy, and V. Vedral, Phys. Rev. Lett. **85**, 4972 (2000).

[6] F. Anselmi, A. Chefles, and M. B. Plenio, New J. Phys. **6**, 164 (2004).

[7] M. Hillery, V. Buzek, and A. Berthiaume, Phys. Rev. A **59**, 1829 (1999).

[8] K. Kraus, *States, Effects, and Operations* (Springer, Berlin, 1983).

[9] E. M. Rains, Phys. Rev. A **60**, 173 (1999).

[10] V. Gheorghiu and R. B. Griffiths, Phys. Rev. A **78**, 020304 (2008).

[11] A. Chefles, Phys. Rev. A **69**, 050307 (2004).

[12] J. I. Cirac, W. Dür, B. Kraus, and M. Lewenstein, Phys. Rev. Lett. **86**, 544 (2001).

[13] D. P. DiVincenzo, T. Mor, P. W. Shor, J. A. Smolin, and B. M. Terhal, Commun. Math. Phys. **238**, 379 (2003).

[14] C. H. Bennett, D. P. DiVincenzo, T. Mor, P. W. Shor, J. A. Smolin, and B. M. Terhal, Phys. Rev. Lett. **82**, 5385 (1999).

[15] E. Chitambar, Phys. Rev. Lett. **107**, 190502 (2011).

[16] C. H. Bennett, D. P. DiVincenzo, C. A. Fuchs, T. Mor, E. Rains, P. W. Shor, J. A. Smolin, and W. K. Wootters, Phys. Rev. A **59**, 1070 (1999).

[17] This definition fixes the outcomes of the measurement, defined by the Kraus operators, but not the probabilities of obtaining those outcomes, which also depend on $\widehat{r}_j$.

[18] The difference between our definition of a separable measurement and the more general concept of a separable operation is that in the latter, represented by $S(\rho) = \sum_j (\widehat{A}_j \otimes \widehat{B}_j)\rho(\widehat{A}_j^\dagger \otimes \widehat{B}_j^\dagger)$, the choice of Kraus operators is not uniquely determined.

[19] This method will label every node by sums of $\widehat{\mathcal{A}}_j$ or $\widehat{\mathcal{B}}_j$. When a complete tree is constructed, one can then (if one wishes, though there is no particular need to do so) directly read off the values of $\mathcal{A}_{i_n}^{(\mathcal{S}_n)}$ and $\mathcal{B}_{i_n}^{(\mathcal{S}_n)}$.

[20] Every pair of convex cones in a given space intersect at the origin. However, since such "trivial" intersections are not of interest to us here, the term "intersection" should be understood to mean "nontrivial intersection" throughout this paper.

[21] M. Boyer and L. Paquette, BIT Numer. Math. **16**, 459 (1976).

[22] This approach does not actually construct all possible trees, only those outside of a specific class of exceptions. For a given separable measurement, it is shown in Appendix C that if no LOCC tree exists outside that class, then none exist within that class either. Therefore, to determine the existence of an LOCC protocol, we only need to consider trees outside this class of exceptions.